COMIZOA

Motion Software Development Kit

TEST & MEASUREMENT & AUTOMATION

COMIZOA INTEGRATED MOTION SYSTEM

JANUARY 2011

P/N 0715-2009-02

© 2007 COMIZOA Inc. All rights reserved

API Reference Manual

TEST & MEASUREMENT & AUTOMATION / COMIZOA

CMMSDK Manual

Copyright © 2011 by COMIZOA, Inc. All rights reserved.

COMIZOA owns all right, title and interest in the property and products described herein, unless otherwise indicated. No part of this document may be translated to another language or produced or transmitted in any form or by any information storage and retrieval system without written permission from COMIZOA. COMIZOA reserves the right to change products and specifications without written notice. Customers are advised to obtain the latest versions of any product specifications.

COMIZOA MAKES NO WARRANTIES, EXPRESSED OR IMPLIED, OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE, OTHER THAN COMPLIANCE WITH THE APPLICABLE COMIZOA SPECIFICATION SHEET FOR THE PRODUCT AT THE TIME OF DELIVERY. IN NO EVENT SHALL COMIZOA BE LIABLE FOR ANY INDIRECT, INCIDENTAL OR CONSEQUENTIAL DAMAGES AS A RESULT OF THE PRODUCT'S PERFORMANCE OR FAILURE TO MEET ANY ASPECT OF SUCH SPECIFICATION. COMIZOA PRODUCTS ARE NOT DESIGNED OR INTENDED FOR USE IN LIFE SUPPORT APPLIANCES, DEVICES OR SYSTEMS WHERE A MALFUNCTION OF A COMIZOA DEVICE COULD RESULT IN A PERSONAL INJURY OR LOSS OF LIFE. CUSTOMERS USING OR SELLING COMIZOA DEVICES FOR USE IN SUCH APPLICATIONS DO SO AT THEIR OWN RISK AND AGREE TO FULLY INDEMNIFY COMIZOA FOR ANY DAMAGES RESULTING FROM SUCH IMPROPER USE OR SALE.

Information contained herein is presented only as a guide for the applications of our products. COMIZOA does not warrant this product to be free of claims of patent infringement by any third party and disclaims any warranty or indemnification against patent infringement. No responsibility is assumed by COMIZOA for any patent infringement resulting from use of its products by themselves or in combination with any other products. No license is hereby granted by implication or otherwise under any patent or patent rights of COMIZOA or others.
COMIZOA software and its documentation are available only under the terms of a Master Software Use and Support Agreement.

Trademarks

The COMIZOA logo is a registered trademark. All other brand names, product names, trademarks, and registered trademarks are the property of their respective owners.

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

고객(顧客) 기술 지원

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

본사 안내
대전 광역시 유성구 탑립동 914 번지

TEL : 042-936-6500 ~ 6506
FAX : 042-936-6507

COMIZOA Motion System Integrated Control Library Reference

© 2011 COMIZOA
All Rights Reserved. No Part of this publication may be reproduced, stored in retrieval system or transmitted, in any form, or by any means, electronic, mechanical, photocopying, recording or otherwise, without the prior permission, in writing, from the publisher.

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 ㈜커미조아는 고객(顧客)님들을 위해 언제나 최선과 정성을 다하는 자세로 지금 이 시간에도 보다 나은 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 설명 드리는 내용은 모션 라이브러리 운용과는 별개의 내용이오나, 제품의 보증과 규정 그리고 저작권에 대한 내용을 설명 드리고 있으므로, 반드시 습득하여 주시기 바랍니다.

1 CMMSDK 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

이 매뉴얼의 일부 혹은 전체를 무단복사, 복제, 전재하는 것은 대한민국 저작권법에 저촉됩니다.

1.1.4 상표안내

Windows 는 Microsoft Corp. 의 등록상표입니다.

Visual C++ 는 Microsoft Corp. 의 등록상표입니다.

Visual Basic 은 Microsoft Corp. 의 등록상표입니다.

C++ Builder 는 Borland Software Corp. 의 등록상표입니다.

Delphi 는 Borland Software Corp. 의 등록상표입니다.

이외의 상표는 각 회사의 등록상표입니다.

1.1.5 주의사항

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

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

최신 내용 및 기술되지 않은 사항 혹은 누락된 사항은 ㈜ 커미조아 제품 군 구입시에 고객(顧客) 등록을 해주신 고객(顧客)님의 정보를 통해 안내해 드릴 예정이며, 기술 지원 요청 시에 보다 자세하고 정확한 방법과 내용을 통해 도움 드릴 것을 약속 드립니다.

본 매뉴얼에 시작하기 앞서, 본 장(Chapter) 에서는 본 매뉴얼을 보다 정확하게 빠르게 이해(理解)하실 수 있도록 CMMSDK 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.

CMMSDK 는 최신 커미조아 라이브러리를 의미하며, 다음과 같은 CMMSDK 와 CMMSDK v2 의 라이브러리 명칭은 매뉴얼의 전체 내용에서 모두 같은 의미를 가집니다.

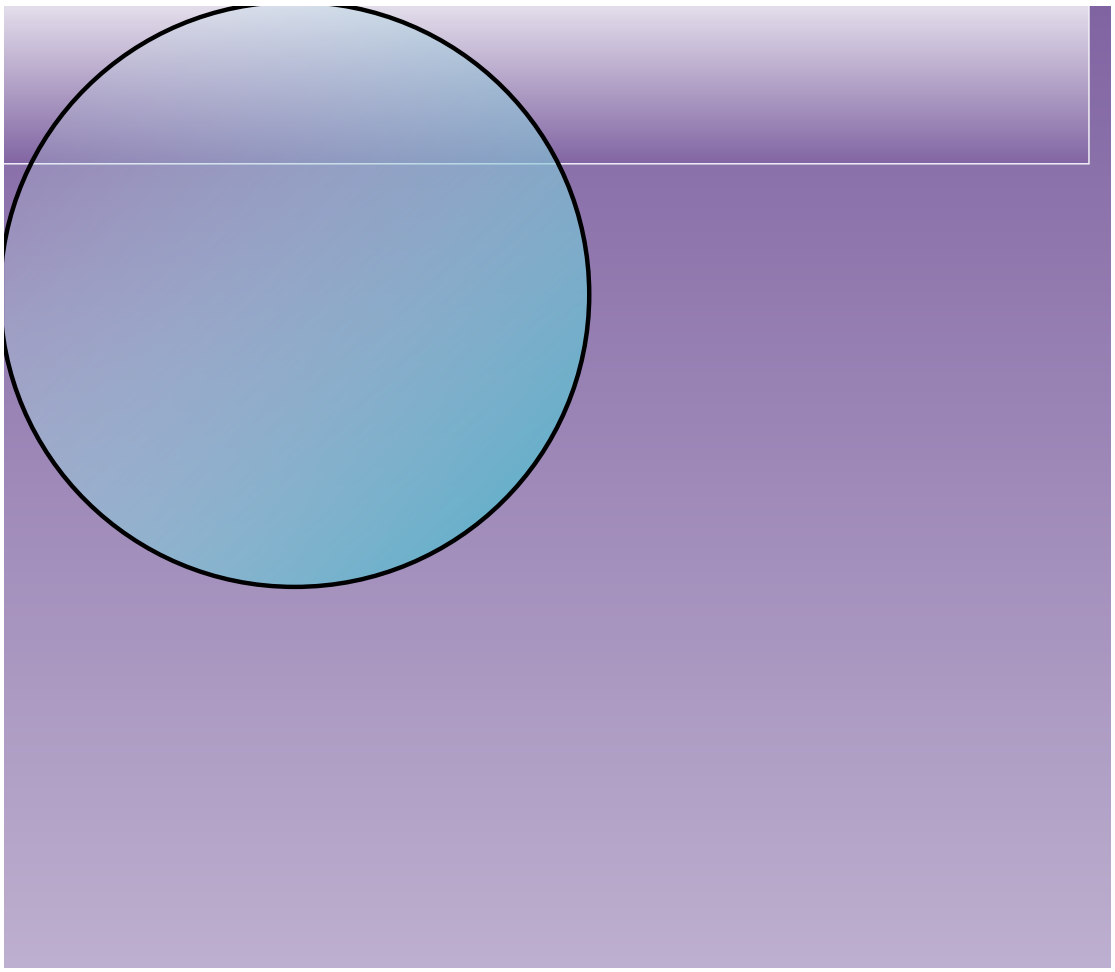


그림 1-1 본 매뉴얼에서 다루는 CMMSDK 라이브러리 명칭

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

명칭	의미
모터 (Motor)	서보모터를 비롯한 스텝 모터,
서보팩 (Servo Pack)	서보 앰프
서보 드라이브 (Servo Drive)	서보모터와 서보앰프의 편성
스텝 드라이브 (Step Drive)	스텝모터와 스텝 모터 제어회로의 편성
서보 시스템 (Servo System)	서보 드라이브와 ㈜ 커미조아 상위 제어기를 조합한, 일련의 완성된 시스템
지령 위치 출력 신호 (Command Pulse)	㈜ 커미조아 모션 제어 제품에서 서보 팩 혹은 스텝 드라이브 측으로 출력하는 전기적 펄스(Pulse) 열 신호
실제 위치 입력 신호 (Feedback Pulse)	㈜ 커미조아 모션 제어 제품에서 위치 검출기(Encoder) 등으로 입력 받는 전기적 펄스(Pulse) 열 신호
모션 보드 (Motion Board)	㈜ 커미조아의 모션 제어를 위한 제품으로서 LX502, LX504, LX508, LX504a 제품의 일부 혹은 종합적으로 일컫는 명칭
디지털 입출력 보드 (Digital Input Output Board)	㈜ 커미조아의 범용 디지털 입출력 제어를 위한 제품으로서 SD4xx Series 제품의 일부 혹은 종합적으로 일컫는 말

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 모션 및 범용 디지털 입출력 통합 라이브러리인 CMMSDK 의 장점은 다음과 같습니다.

1.2.1 독립성

CMMSDK 는 Microsoft 社의 표준라이브러리인 DLL(Dynamic Link Library) 형태의 독립된 라이브러리 인터페이스를 제공하며, 라이브러리 자체의 독립된 장치 관리 및 "COMIZOA Motion Environment" 를 이용한 장치 관리의 편의성을 제공하고 있으며, DLL 형태의 라이브러리 장점을 통해 유지 보수와 귀사의 제품 구현에 보다 간편하게 하고 신뢰성 있는 독립 형 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 CMMSDK 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 시간 내에 숙련된 라이브러리 사용자로 만들어드립니다. 특히, CMMSDK 의 모든 함수 명에는 의미적 명명 규칙을 내포하였으며, 이것은 분명 실무 개발 환경에서 많은 부분 이점으로 작용할 것입니다.

1.2.4 확장성

㈜ 커미조아 통합 라이브러리인 CMMSDK 는 커미조아의 기존 제품 군과 함께 사용할 수 있습니다. 기존 COMI-AX Pro 를 비롯한 기존 커미조아의 라이브러리와도 상호 호환성을 보장하며, 기존 라이브러리 사용자도 빠른 시간 내에 ㈜ 커미조아 CMMSDK 를 통해 제품 개발을 더욱 향상시키실 수 있습니다.

다만, 기존 라이브러리 고객(顧客)님들은 본 매뉴얼을 통해 신규 CMMSDK 의 변화된 모습과 기능향상에 대해서 확인(確認)해주시기를 권고 드리겠습니다.

1.2.5 신뢰성

㈜ 커미조아의 라이브러리 제품 군은 오랜 시간 산업현장에서의 현장 경험을 바탕으로 형성된 신뢰성 있는 제품 군입니다. '부드럽고, 정확하고, 빠르고, 쉬운' 모션제어 기능을 통해 모션의 기본에 충실하였으며, 각 동작에 대한 입력과 출력을 CMMSDK 인터페이스 전역에 걸쳐 사용하기 쉽도록 안내해 놓았습니다.

또한, 응용프로그램에 기반하지 않는 자체 디버그 기능을 바탕으로 응용프로그램에 의한 오류를 최단시간 내에 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.

1.2.6 풍부한 예제와 신속한 기술 지원

지금 이 시간에도 ㈜ 커미조아는 타사에 비해 보다 많은 개발 선상의 활용 가능한 라이브러리 예제를 지원해드리려고 노력하고 있으며, 예제간 중복 코드와 라이브러리 구성 이외의 부분을 최소화한 예제로 빠른 시간 내에 예제의 코드를 바로 적용시켜드릴 수 있도록 노력하고 있습니다. 특히 Smart Update 기능을 통해 라이브러리의 최신 버전을 인터넷을 통해 다운로드 받아 바로 업그레이드할 수 있는 기능을 제공하고 있습니다.

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

Chapter

2

Before working with CMMSDK

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

본 장에서는 고객지원(顧客支援)과 저희 ㈜ 커미조아의 CMMSDK 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 기존 라이브러리를 사용중인 고객(顧客)을 위한 안내

본 장에서 설명 드리는 라이브러리 안내 사항은 변화된 CMMSDK 의 사항과 더불어 기존 라이브러리 사용자를 위한 변경 사항을 자세히 안내해 드리고 있습니다.

2.1 ㈜ 커미조아의 COMI-AUTOMATION 패키지 구성

커미조아 모션 라이브러리의 배포는 COMI-AUTOMATION 이라는 패키지 단위로 배포되며, 지난 2007 년 1 월 시점을 기준으로 기존COMI-AUTOMAION V1 을 새롭게 업그레이드 한 COMI-AUTOMATION V2 를 배포하고 있습니다. 또한 2010 년 부터는 COMI-AUTOMATION V3 를 배포하고 있으며 32/64 비트 OS 를 모두 지원하도록 드라이버 및 라이브러리 파일을 제공하고 있습니다. 최신 COMI-AUTOMATION V3 패키지를 설치 하시면 최신의 커미조아 제품 통합 환경에서 제품을 사용하실 수 있습니다.

COMI-AUTOMATION 1 과 COMI-AUTOMATION 2 의 주요 구성은 다음과 같습니다.

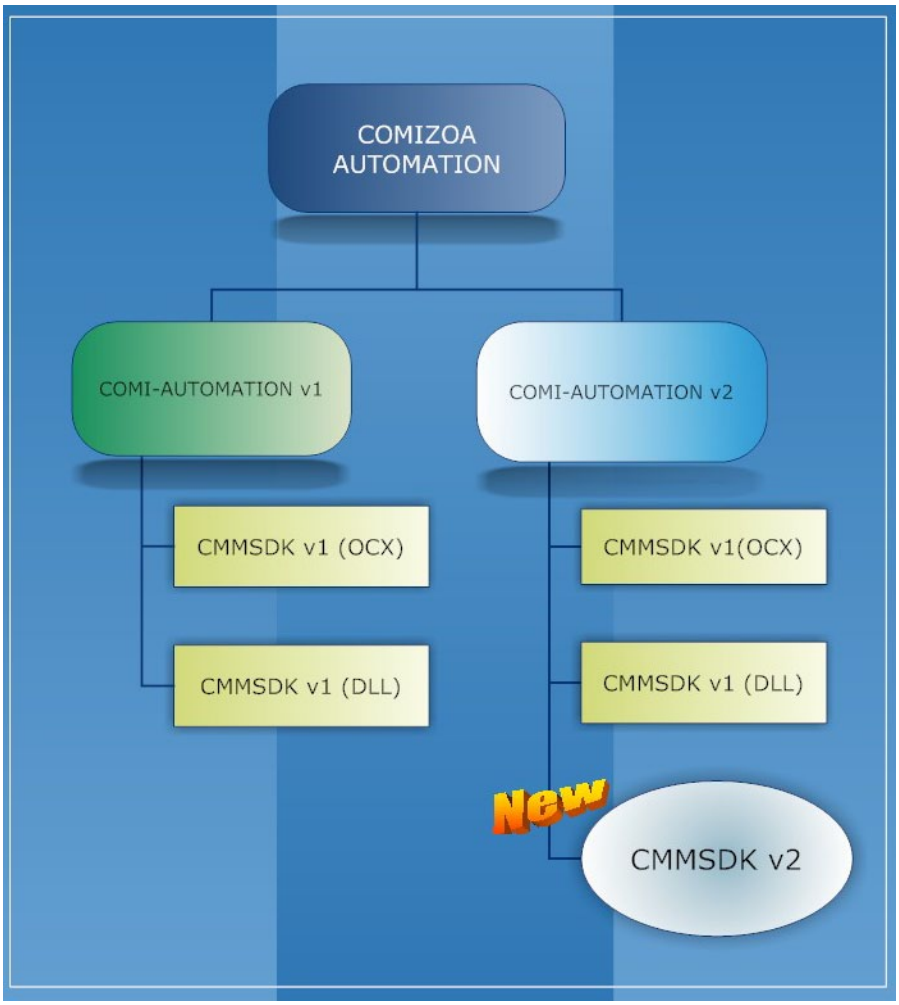


그림 2-1 COMI-AUTOMATION 구성도

2.2 라이브러리 별 기능 변화 표

다음 "표 3 ㈜ 커미조아 라이브러리 버전 별 기능 대응 표" 에서는 각 라이브러리의 배포 시기와 배포 버전에 대해서 설명해드리고 있습니다. 참고적으로 CMMSDK v1 과 CMMSDK v2 는 윈도우 표준 동적 라이브러리 형태인 DLL(Dynamic Link Library) 형태를 가지고 있으며, CMMSDK v1 OCX 는 Active X 컨트롤 형태를 가지고 있습니다.

구분	CMMSDK v1	CMMSDK v1 OCX	CMMSDK v2
배포버젼	3.1.3.2 이상	4.1.9.3 이상	5.0.0.0 이상
배포 시기	2003 년 4 월	2005 년 10 월	2007 년 1 월
형태	윈도우 표준 동적 연결 라이브러리(Dynamic Link Library)	윈도우 표준 응용프로그램 및 개발 환경에서 사용 가능한 컴포넌트(Component)	윈도우 표준 동적 연결 라이브러리(Dynamic Link Library)
개요	초기 DLL 버전으로 모션 제어 기본 기능을 바탕으로 한 뛰어난 라이브러리 인터페이스 구성	편리한 라이브러리 구성을 중심으로 부드럽고 정확한 모션제어에 초점으로 설계된 라이브러리 Master / Slave 동기 제어 지원 Gantry 제어	이전 버전의 모든 기능 포함 모션 제어 / 범용 디지털 입출력 통합 제어 속도 구간별 비젼 Trigger 출력 지원 기준속도(Standard Speed) 개념의 도입

표 3 ㈜ 커미조아 라이브러리 버전 별 기능 대응 표

다음 "그림 2-2 커미조아 라이브러리 배포 시기 안내"는 각 라이브러리 별, 발표 시기를 안내하고 있습니다.

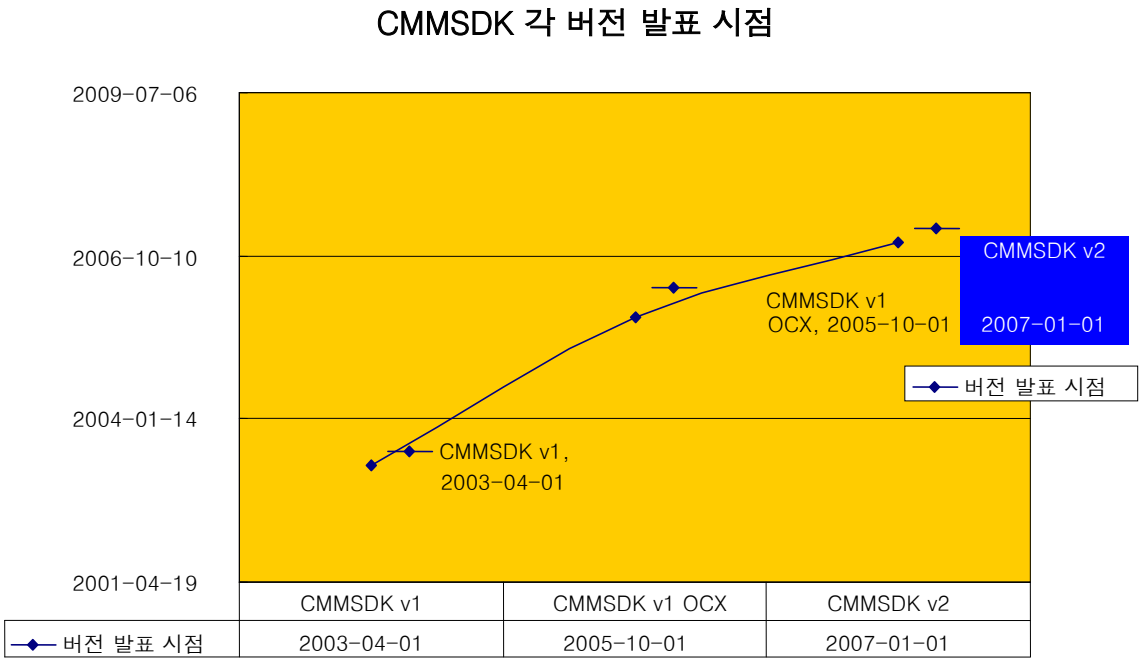


그림 2-2 커미조아 라이브러리 배포 시기 안내

2.3 CMMSDK 라이브러리 주요 내용

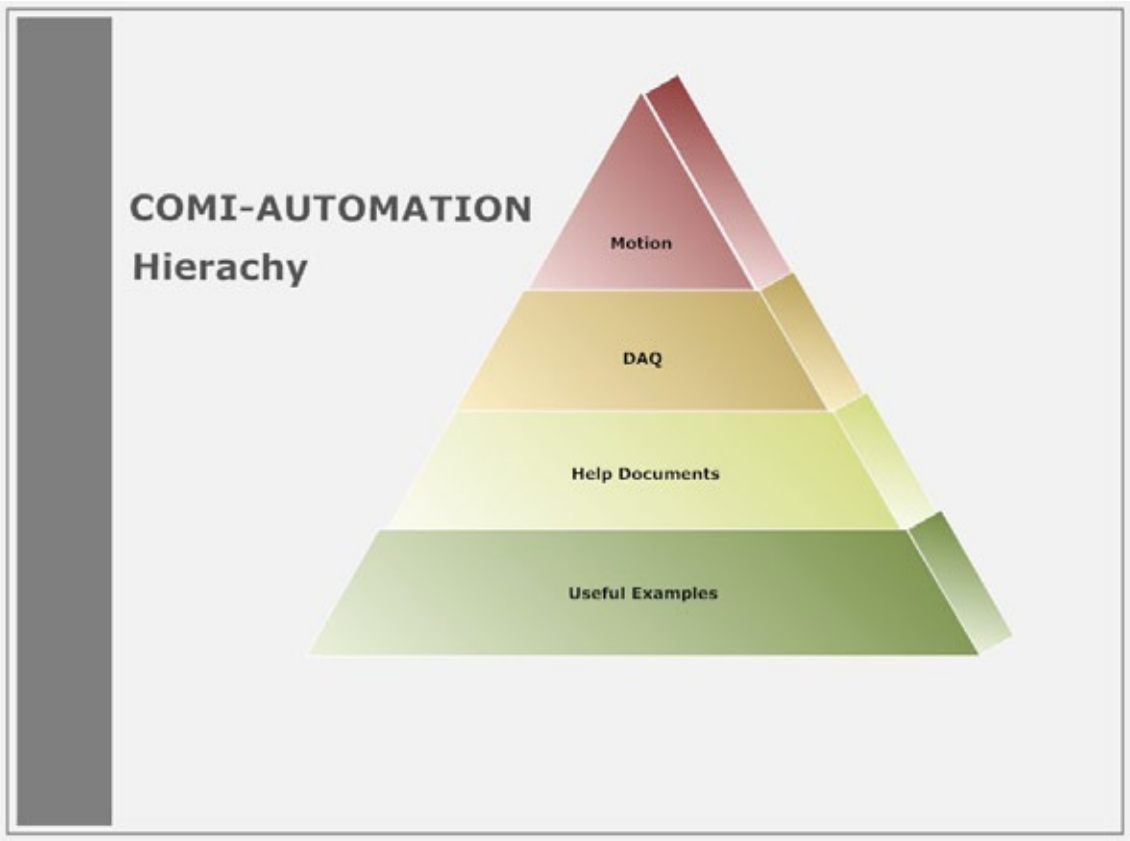


그림 2-3 COMI-AUTOMATION Hierarchy

㈜ 커미조아의 라이브러리는 그 동안 최신 산업용 컴퓨터 (Industrial Computer) 와 개인용 컴퓨터 및 워크스테이션과의 뛰어난 호환성으로 호평 받아 왔습니다. 이번 CMMSDK 에서는 안정된 하드웨어 환경을 바탕으로 소프트웨어 라이브러리가 직접 하드웨어 환경을 통합하여 관리하는 방식의 통합 시스템 제어 (Integration System Control) 기법을 통하여, 주요 고객(顧客)님들께 그 동안 저희 커미조아 제품의 각 PCI 카드별 라이브러리 인터페이스를 별도로 해야 하는 부분들을 편리하게 안내하고 있습니다.

2.3.1 CMMSDK 개요 사항

시스템에 장착된 모션 컨트롤러 제품과 범용 디지털 입출력 제품을 종합하여 운용할 수 있도록 해드립니다. 새로운 통합라이브러리 CMMSDK 는 기존 라이브러리보다 향상된 기능 구성과 더불어 모션 제어 축(Axis) 의 조합과 범용 디지털 입출력(I/O) 의 조합을 자유롭게 구성할 수 있도록 도움을 드립니다. 특히 그 동안 각 제품별로 개별 라이브러리로 관리해야 했던 어려움과 하드웨어 구성에 따라 편리하지 못했던 제어 방법에 새로운 가능성을 제시하여 드립니다.

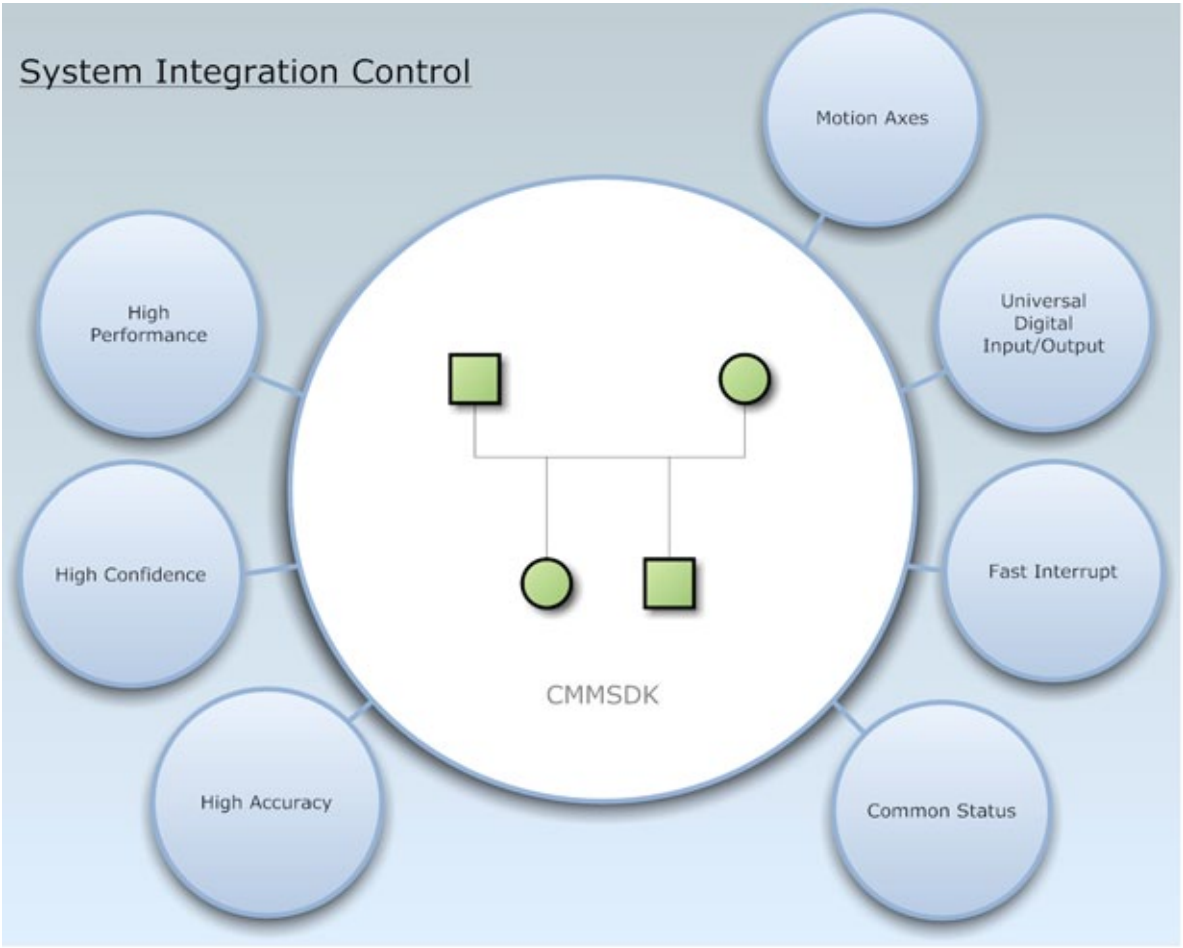


그림 2-4 시스템 통합 제어

안내	㈜ 커미조아의 시스템 통합 제어의 가장 큰 이점은 무엇인가요?
	당사의 제품 라이브러리는 그 동안 개별 PCI 카드(1) 마다 개별적인 라이브러리를 호출하여 사용해 왔습니다. 당사의 최신 모션제품이 응용되는 장비의 동향을 보면, 그 제어 제품의 수가 기하급수적으로 늘어남에 따라, 소프트웨어 엔지니어들의 다중 모션 및 범용 디지털 I/O 제어의 부담이 가중되어 왔습니다. CMMSDK 는 시스템에 장착되어 있는 모션 보드 제품과 디지털 입출력 제품을 집중적으로 관리하는 혁신을 가져왔습니다.

2.3.2 함수 이름 규칙 가이드

구분 명		CMMSDK v1	CMMSDK OCX	CMMSDK v2
장치 초기화 (Device Initialize)		COMILX_LoadDevice	GnDeivceLoad	cmmGnDeviceLoad
		COMILX_UnloadDevice	GnDeviceUnLoad	cmmGnDeviceUnLoad
MOTION	단축 제어 (Single Axis)	COMILX_MC_Move	SxMove	cmmSxMove
		COMILX_MC_MoveTo	SxMoveTo	cmmSxMoveTo
	다축 제어 (Multi Axes)	COMILX_MC_MoveAll	MxMove	cmmMxMove
		COMILX_MC_MoveToAll	MxMoveTo	cmmMxMoveTo
	보간 제어 (Interpolation)	COMILX_MC_MapAxes	IxMapAxes	cmmIxMapAxes
		COMILX_MC_Line	IxLine	cmmIxLine

1 인텔사를 중심으로 한 미국의 주요 개인용 컴퓨터(PC) 관련 제조업체 백수십 개 회사가 참가하여 작성한 로컬 버스 규격. 일반적으로 약어로 불리는 로컬 버스의 하나이다.

	원점 복귀 (Home Return)	COMILX_MC_SetHomeConfig	HomeSetConfig	cmmHomeSetConfig
		COMILX_MC_HomeMove	HomeMove	cmmHomeMove
	외부신호 동기제어 (Manual Pulsar)	COMILX_MC_PulserMove	PlsrMove	cmmPlsrMove
		COMILX_MC_PulserMoveTo	PlsrMoveTo	cmmPlsrMoveTo
	리스트 모션 (Listed Motion)	COMILX_MC_BeginList	LmBeginList	cmmLmBeginList
		COMILX_MC_EndList	LmEndList	cmmLmEndList
D I O	범용 디지털 입력 (Digital Input)	COMILX_DI_GetOne	DiGetOne	cmmDiGetOne
		COMILX_DI_GetAll	DiGetAll	cmmDiGetAll
	범용 디지털 출력 (Digital Output)	COMILX_DO_PutOne	DoPutOne	cmmDoPutOne
		COMILX_DO_PutAll	DoPutAll	cmmDoPutAll

표 4 함수이름 규칙

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

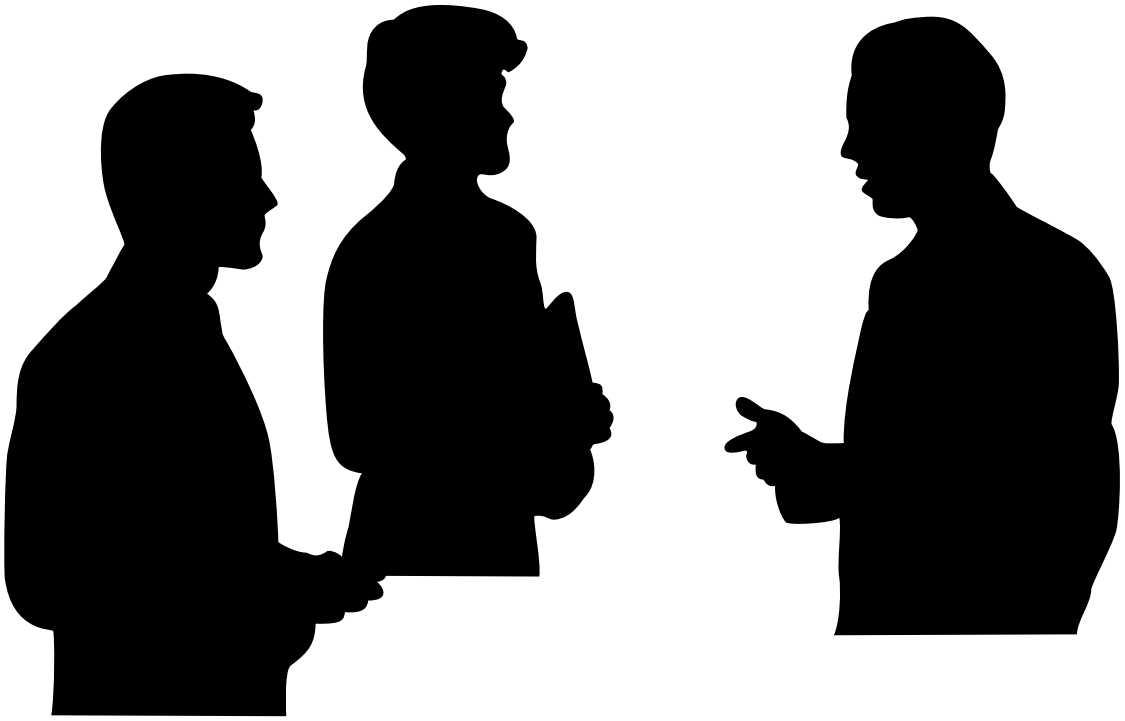
Chapter

3

Development Environment for CMMSDK

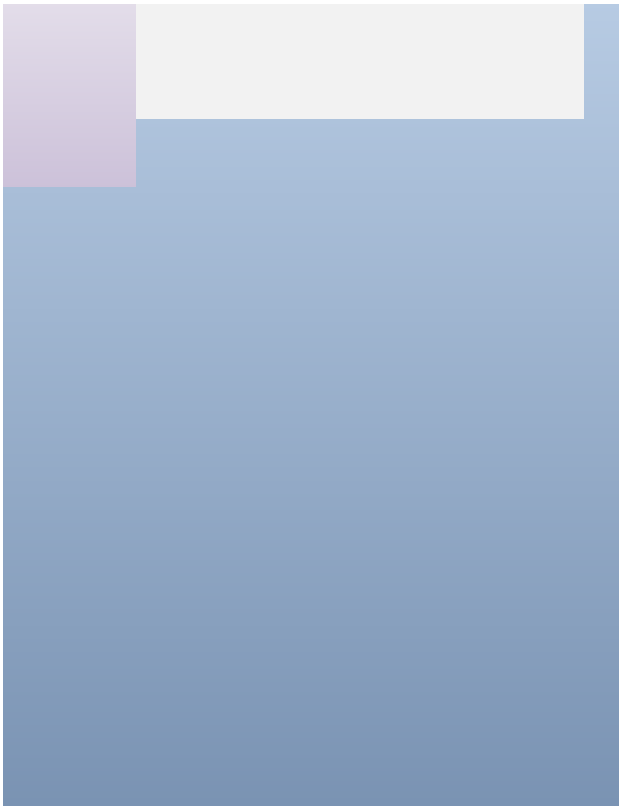
㈜커미조아는 통합 라이브러리 CMMSDK 를 통해 다양한 최신 개발환경을 지원하기 위해 노력하고 있습니다. 본 장에서 다루지 않는 개발 환경을 이용하시는 고객(顧客)님께서는 저희 ㈜ 커미조아를 통해 문의해주시면 신속히 대처해 드리도록 하겠으며, 제공되는 CMMSDK 인터페이스를 통해 보다 편리하고 빠르게 저희 라이브러리에서 사용할 수 있도록 지원하여 드립니다.

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



3 개발 환경 별 CMMSDK 사용 안내

3.1 개발 환경 지원 안내



3.2 CMMSDK 구성

CMMSDK 는 아래 그림과 같이 실제 COMIZOA 제품 군의 하드웨어를 추상화하여, 공통(共通) API 구조를 제공하고 있습니다.

CMMSDK 는 "Integration Motion System Control Application Programming Interface" 라 명명하며, 그 의미는 저희 ㈜ 커미조아 제품 군의 기능과 그 기능을 사용하는 방법을 정의한 함수(Function)의 집합이라고 말할 수 있습니다. 고객(顧客)님께서는 이제 저희 ㈜ 커미조아 API 를 통해 제품 군이 가지고 있는 다양한 기능을 사용하실 수 있습니다.

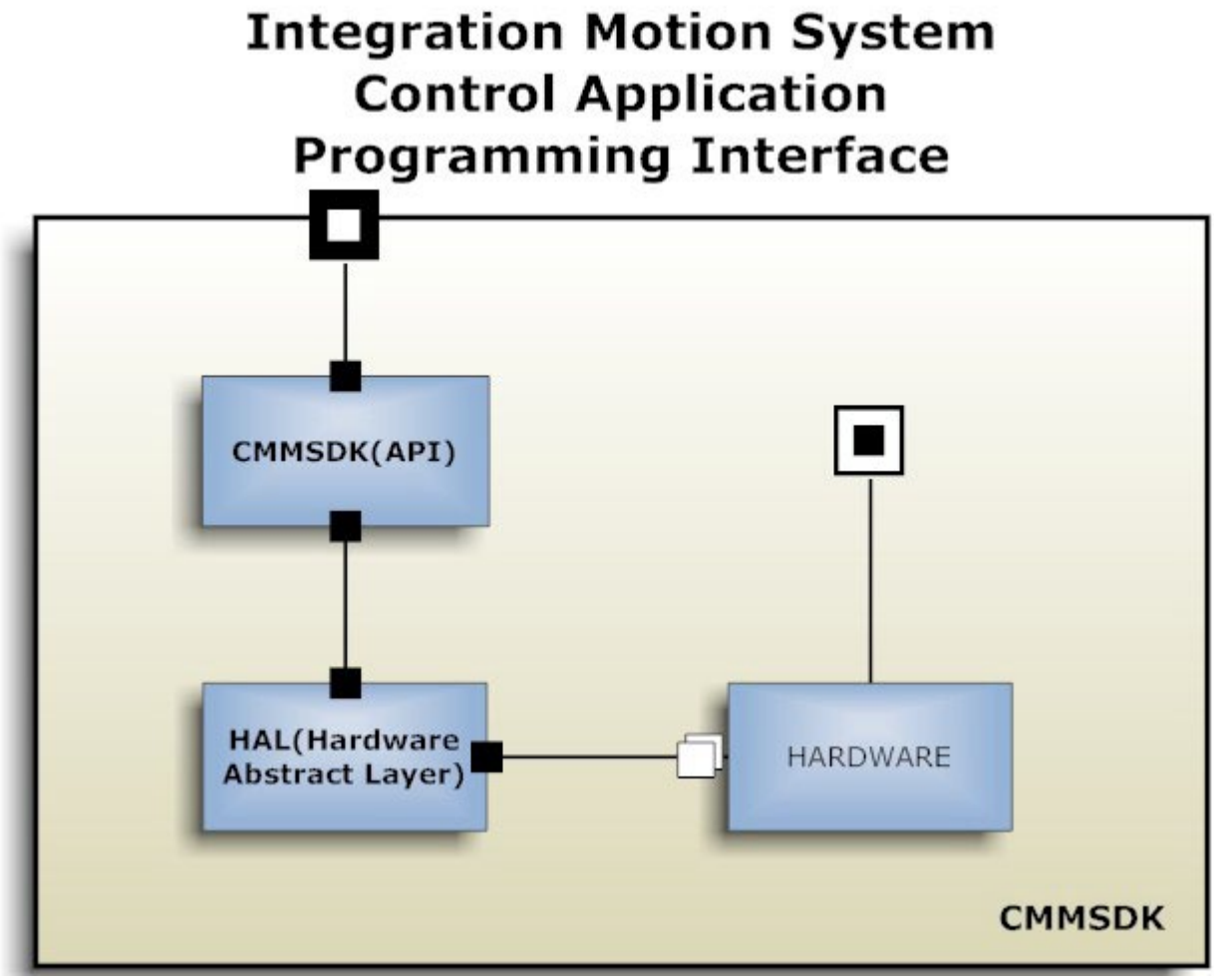


그림 3-1 CMMSDK 의 구조

3.2.1 HARDWARE Layer

다양한 커미조아의 제품 군은 개별적인 인터페이스를 필요로 합니다. COMI-SD4xx Series 등의 범용 디지털 입출력장치와 COMI-LX504 모션 보드는 분명 별도의 인터페이스가 필요합니다. 그러나 이러한 개별적인 장치에 대한 개별적인 제어의 어려움은 CMMSDK 를 통해 해결되었습니다. 이제 하드웨어 구조를 추상화하여, 일관적인 API 인터페이스를 제공하게 해주는 CMMSDK 를 통해 편리하게 모든 모션 장치와 범용 디지털 입 출력 장치를 일관적으로 제어하실 수 있습니다.

3.2.2 HAL(Hardware Abstract Layer)

Hardware Device 는 CMMSDK가 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 CMMSDK에서 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 CMMSDK 가 고급 HAL(Hardware Abstract Layer) 기능을 바탕화 했기 때문입니다.

3.2.3 CMMSDK Layer (API Layer)

마이크로소프트社의 Windows 98/ME/2000/XP/Vista등의 제품 군에서 동적 연결 라이브러리 방식을 지원하는 라이브러리를 의미합니다. 사용자는 Integration Motion System Control API, 즉 CMMSDK를 통해 제품 개별적인 하드웨어 인터페이스에 대한 사항들을 사용자는 직접 인지하고 있지 않아도 CMMSDK는 고객(顧客)여러분들의 요구에 맞는 인터페이스를 제공하며, 신속하고 안정적인 응용프로그램 개발에 도움을 드립니다.

3.2.4 CMMSDK 인터페이스 구성 파일

개발 환경 / 항목	MS VC++	Borland C++ Builder	Borland Delphi	MS Visual Basic	MS C# (C Sharp)
CMMSDK 인터페이스 파일 명	Cmmsdk.h	Cmmsdk.h	CmmSDK.PAS	CmmSDK.BAS	CmmSDK.CS
CMMSDK 상수(常數)정의 파일	Cmmsdk.cpp CmmsdkDef.h	Cmmsdk.cpp CmmsdkDef.h			

표 6 CMMSDK 인터페이스 구성 파일 안내

Microsoft 社의 윈도우 운영체제에서 동작하는 DLL(Dynamic Link Library) 형태의 CMMSDK는 상기 표에서 나타낸 것처럼, 라이브러리 인터페이스를 위해 "CMMSDK 인터페이스" 파일과 라이브러리의 반환값 및 전달인자, 각 데이터 표기 등을 위한 "CMMSDK 상수(常數) 정의 파일"을 제공하고 있습니다.

㈜ 커미조아 고객(顧客)님께서는 "CMMSDK 인터페이스 파일" 을 통해 CMMSDK의 함수를 명시적으로 응용프로그램에 포함시킬 수 있으며, 이에 따라 상기 표기한 개발 환경을 지원해드리고 있습니다. 만약 이외의 환경에서 저희 CMMSDK를 사용하시기 원하신다면, 'Appendix C, 다른 개발 환경 사용자를 위한 라이브러리 사용 안내' 편을 참고해주시거나, 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 제공**하고 있습니다. 특히 이번 CMMSDK 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 파일 또는 프로젝트 구성파일로 반드시 사용이 됩니다. 저희 ㈜ 커미조아 CMMSDK 에서 제공하는 인터페이스파일의 경로는 다음과 같습니다.

Visual C++ / Borland C++ Builder	
경로 명	C:\Program Files\COMIZOA\AUTOMATION2\Libraries\Motion(DLL)
파일 명	Cmmsdk.cpp, Cmmsdk.h, CmmsdkDef.h
Delphi	
경로 명	C:\Program Files\COMIZOA\AUTOMATION2\Libraries\Motion(DLL)
파일 명	CmmSDK.PAS
Visual Basic	
경로 명	C:\Program Files\COMIZOA\AUTOMATION2\Libraries\Motion(DLL)
파일 명	CmmSDK.BAS
C# (CSharp)	
경로 명	C:\Program Files\COMIZOA\AUTOMATION2\Libraries\Motion(DLL)
파일 명	CmmSDK.cs CmmSDKDef.CS

표 7 이 경로는 ㈜ 커미조아에서 제공하는 COMI-AUTOMATION 설치 프로그램이 기본 경로인 'C:' 드라이브에 설치된 것을 기준으로 합니다.

인터페이스 파일에 대해서 자세히 설명해 주십시오.

먼저, 저희 ㈜ 커미조아에서 제공하는 CMMSDK 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, 다시 말해 함수의 사용법을 알 수가 없습니다.

CMMSDK 에서는 해당 DLL CMMSDK 에 대해서 그 사용방법을 각 개발환경에 맞게 제공해드리기 위해 [인터페이스] 파일을 제공하고 있습니다. 각 개발환경(VC++, Delphi) 등의 환경에서 개발을 하시는 고객(顧客) 여러분들께서는 반드시 이 [인터페이스] 파일을 사용하셔야만, CMMSDK 를 사용할 수 있습니다.

3.3.1 Visual C++ 6.x 개발자를 위한 안내

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

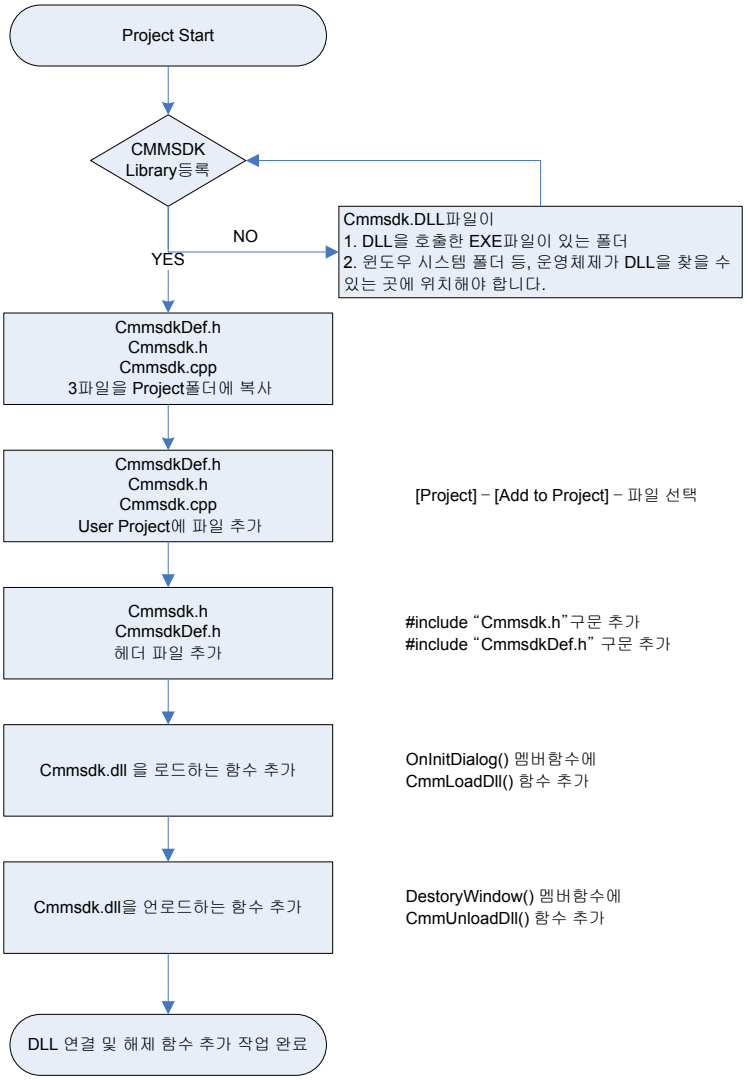


그림 3-2 Visual Studio 6.x 에서의 CMMSDK 사용 순서도

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

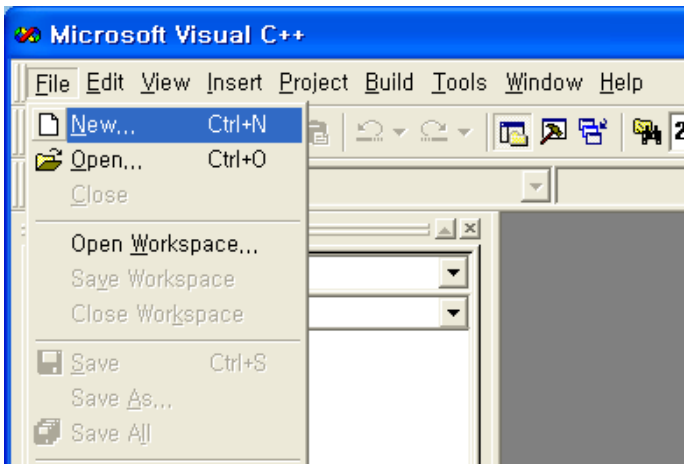


그림 3-3 Visual C++ 6.x 의 새로운 프로젝트 생성 화면

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

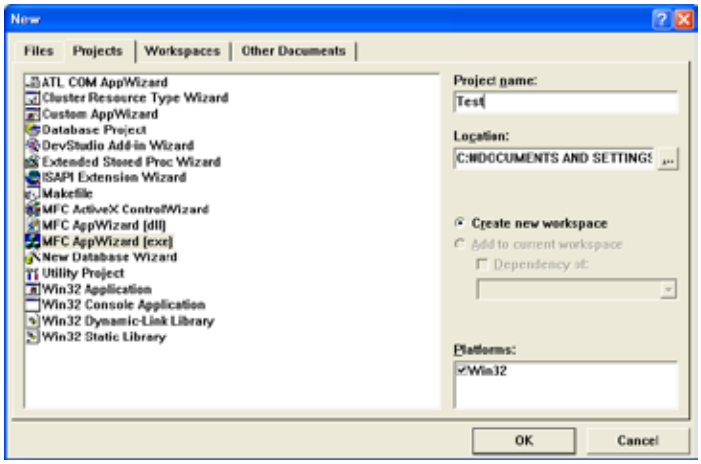


그림 3-4 새로운 프로젝트 생성 화면

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

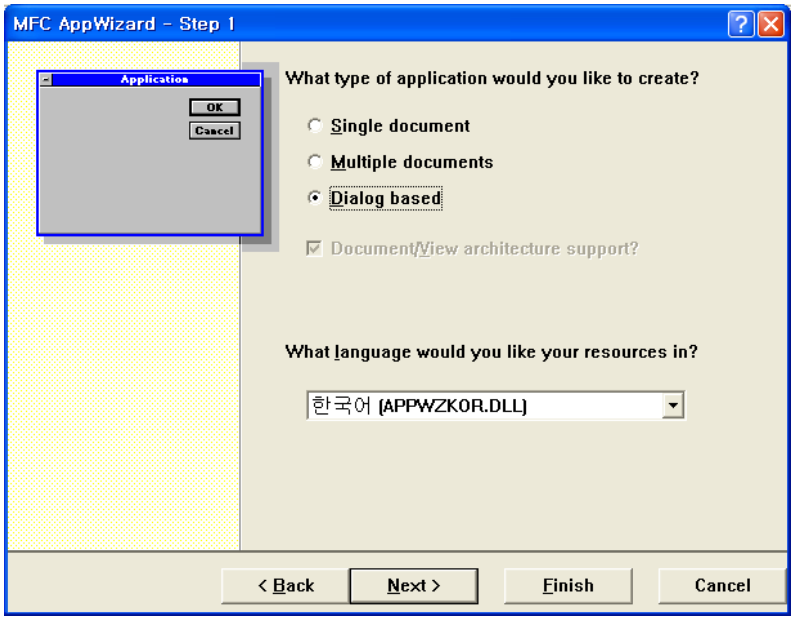


그림 3-5 MFC AppWizard 의 Application Type 선택 화면

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

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

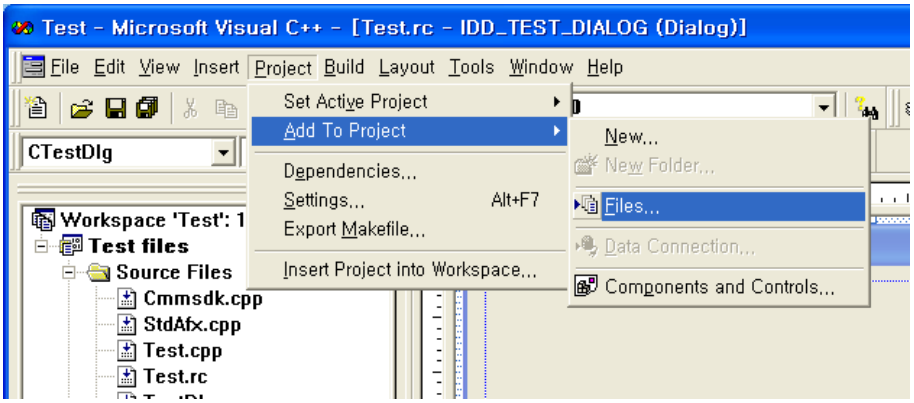


그림 3-6 프로젝트에 새로운 파일 추가 선택화면

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

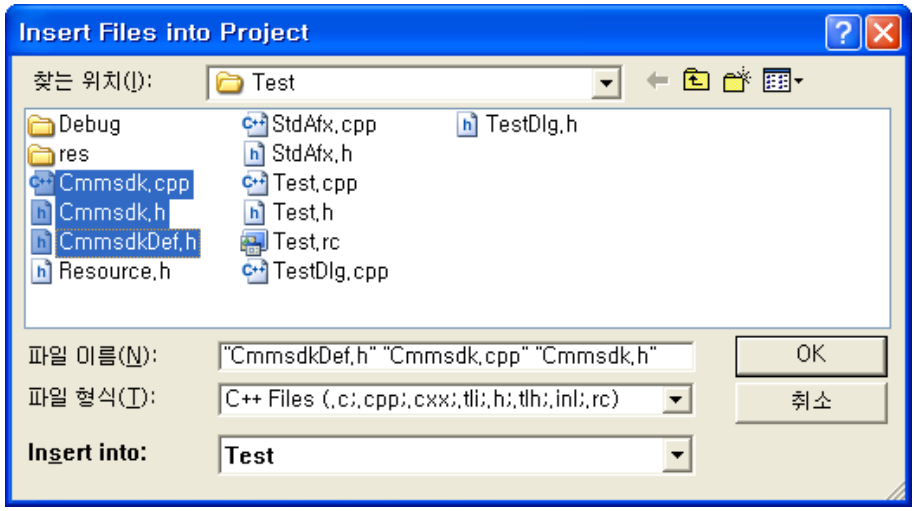


그림 3-7 프로젝트에 추가할 파일 선택 화면

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

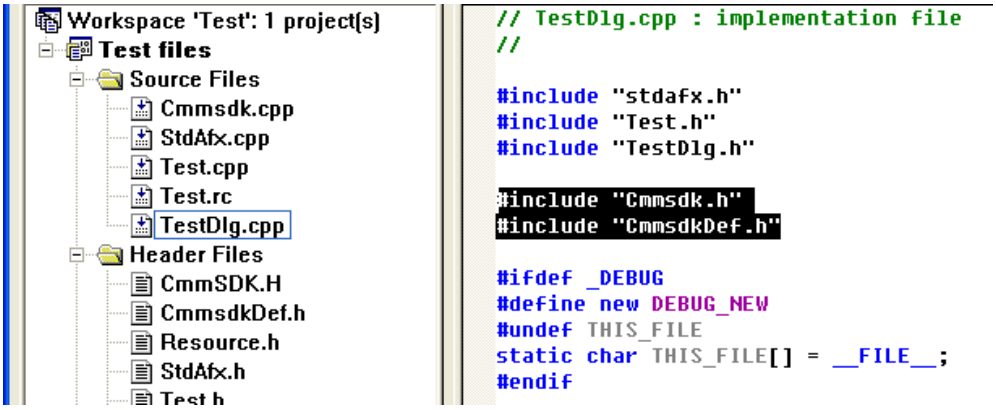


그림 3-8 사용자가 MFC AppWizard 를 통해서 생성한 소스코드에 CMMSDK 파일을 추가 함

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

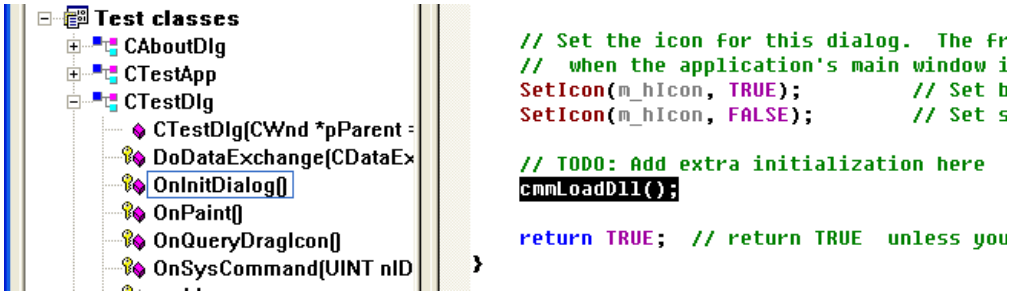


그림 3-9 DLL 로드 함수 호출 화면

사용자 작성 프로그램이 종료되면 DLL 을 Unload 시켜야 합니다. DLL 의 Unload 는 사용자 작성 프로그램의 종료 시 이루어져야 하며 cmmUnloadDll()이라는 함수를 통해서 이루어 집니다. cmmUnloadDll()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.

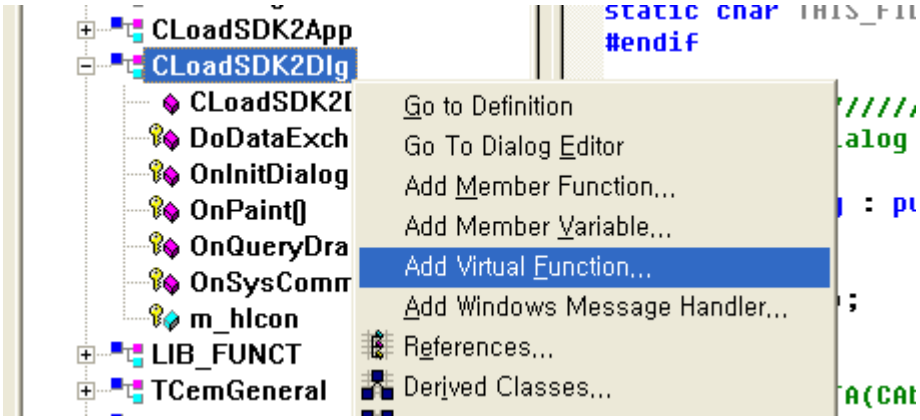


그림 3-10 가상 함수 추가

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

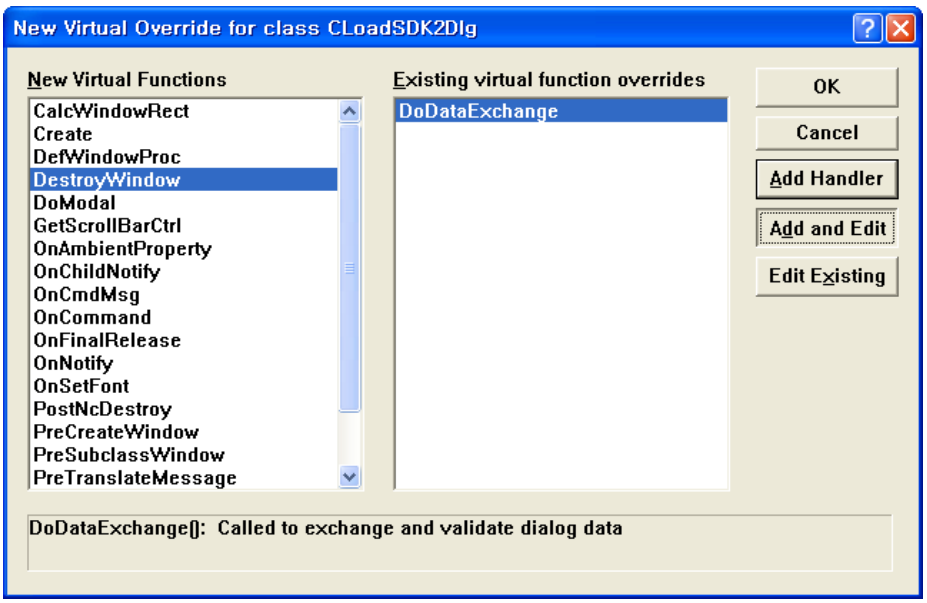


그림 3-11 DestroyWindow 함수 추가

(생성한 프로젝트 이름)+Dlg 클래스의 멤버함수인 'DestroyWindow()'에 'cmmUnloadDll();'을 추가합니다. 'cmmUnloadDll()'함수를 추가하면 윈도우가 종료 될 때 자동으로 DLL 도 해제됩니다.

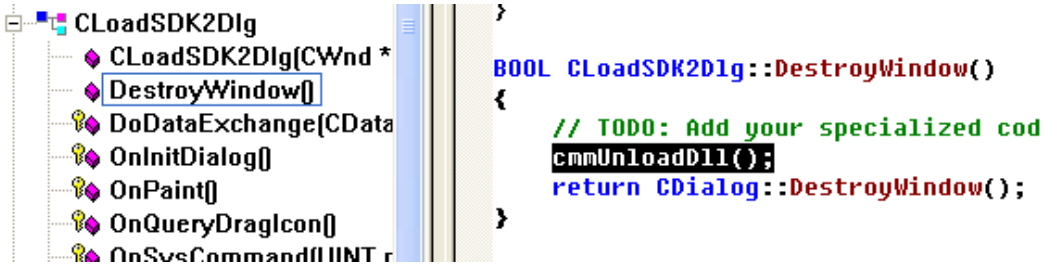


그림 3-12 DLL 로드 및 언로드 코드 추가

3.3.2 Visual C++ 7.x 개발자를 위한 안내

Microsoft 社의Visual C++ 7.x(Visual Studio 2003)에서 CMMSDK를 사용하시려면 다음의 절차에 따라 사용하시면 됩니다.

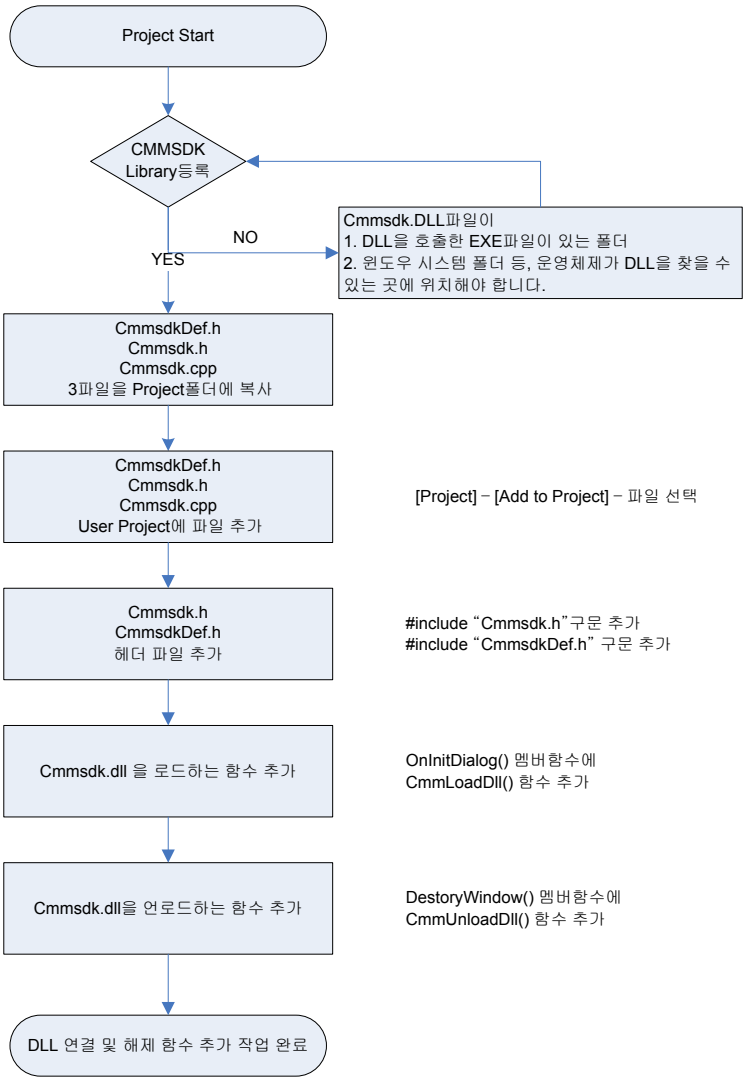


그림 3-13 Visual Studio 7.x 에서의 CMMSDK 사용 순서도

Microsoft 社의 Visual Studio 2003 을 실행합니다. 메뉴에서 [파일]->[새로 만들기]를 선택하여 새로운 프로젝트를 시작합니다.

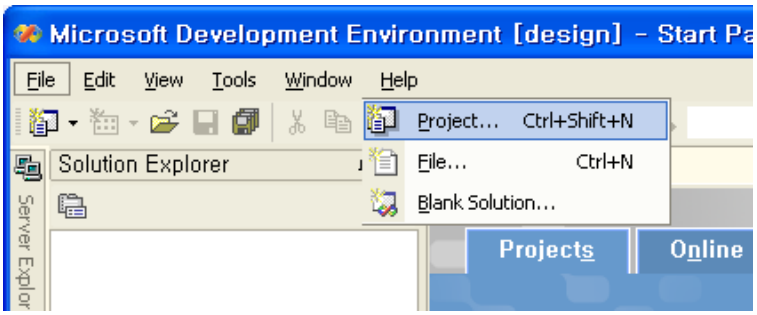


그림 3-14 프로젝트 생성의 시작

[새 프로젝트]창의 [프로젝트 형식]에서 [Visual C++프로젝트]를 선택하고, [템플릿]에서 [MFC 응용 프로그램]을 선택합니다. 그리고 프로젝트를 생성할 위치와 프로젝트 이름을 입력한 후 [확인(確認)] 버튼을 클릭합니다.

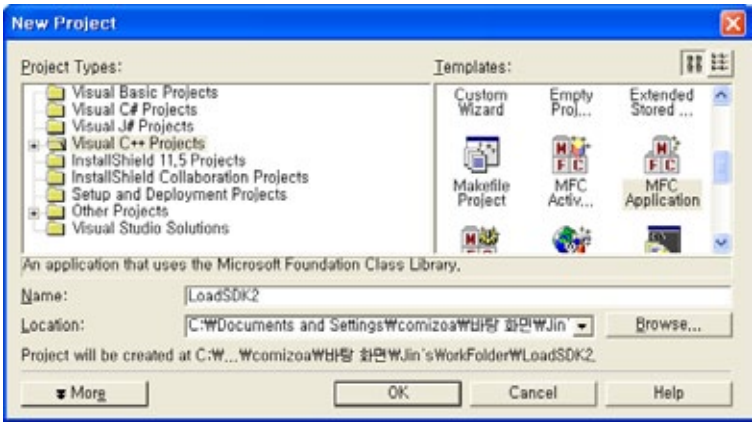


그림 3-15 프로젝트 경로 입력

[MFC 응용 프로그램 마법사]창이 나타나면, [응용 프로그램 종류]를 선택합니다.

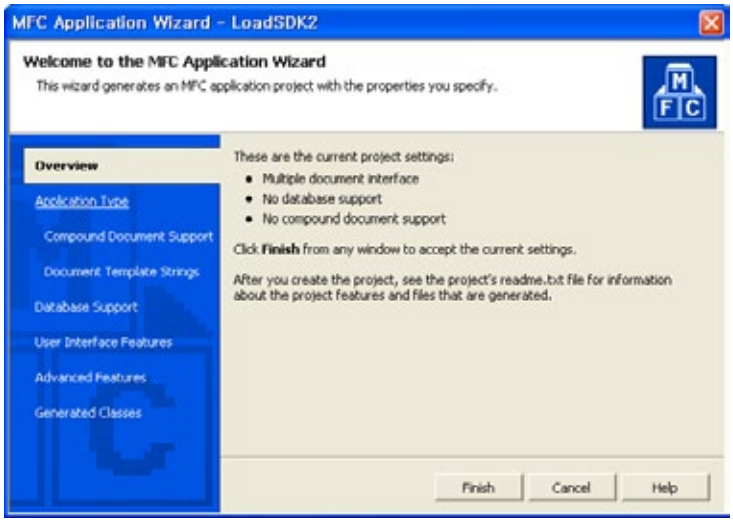


그림 3-16 응용 프로그램 종류 선택

[응용 프로그램 종류]카테고리에서 [대화 상자 기반]을 선택한 후 마침을 클릭합니다.

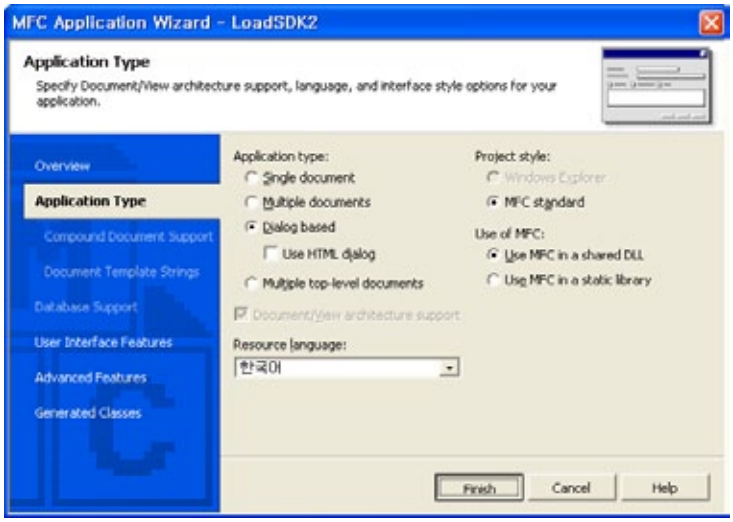


그림 3-17 대화상자 기반 프로젝트 생성

VC++ 용 인터페이스 정의 파일인 Cmmsdk.h, Cmmsdk.cpp, CmmsdkDef.h 파일을 신규로 생성한 프로젝트 폴더로 복사합니다.

메뉴에서 [프로젝트]->[기존 항목 추가]를 선택합니다.

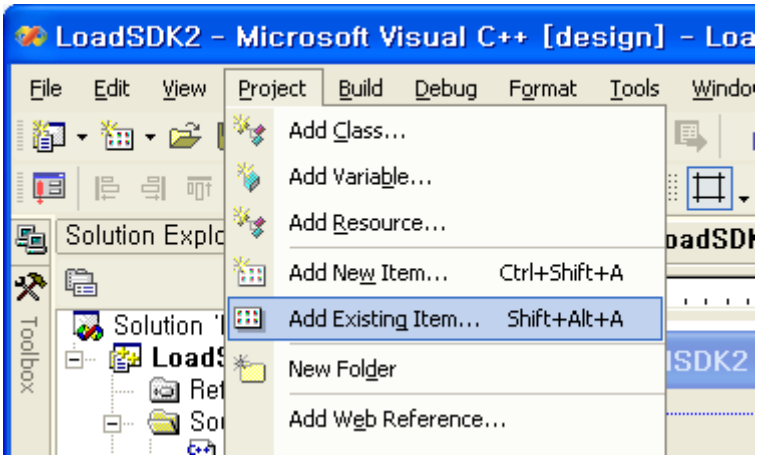


그림 3-18 기존 항목 추가 선택

추가될 파일을 선택한 후 [OK]버튼을 클릭하여 제공되는 헤더 및 소스 코드 파일을 프로젝트에 추가합니다.

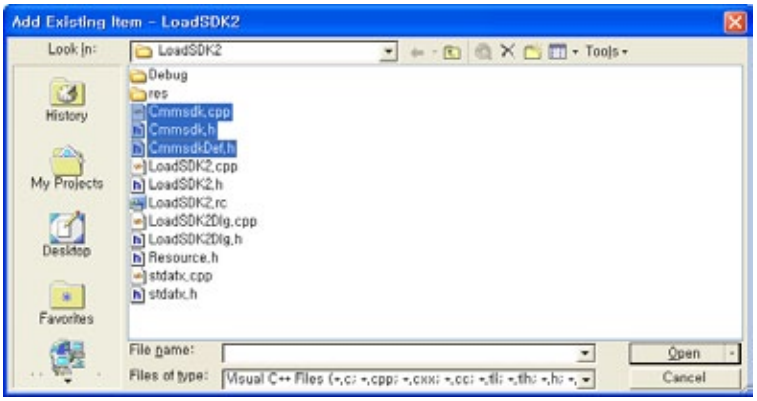


그림 3-19 헤더파일 및 소스 파일 추가

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

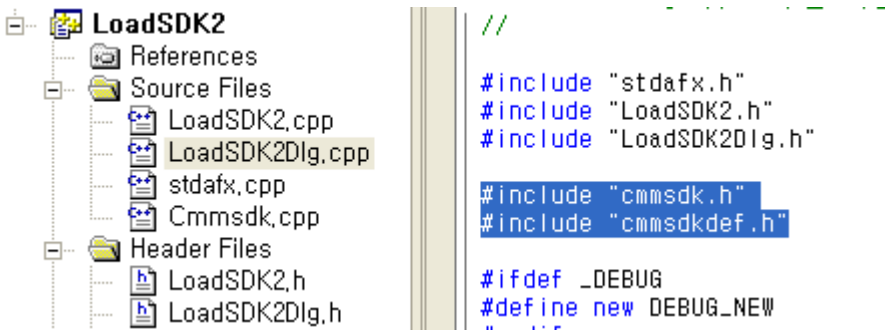


그림 3-20 인터페이스 헤더 파일 추가

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

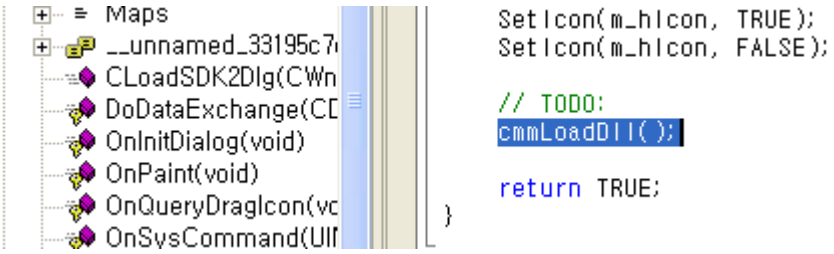


그림 3-21 Load Dll

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

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

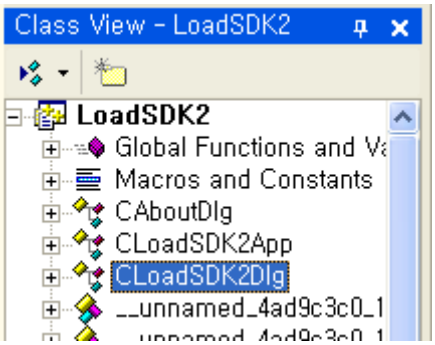


그림 3-22 사용자 생성 Dialog Class 선택

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

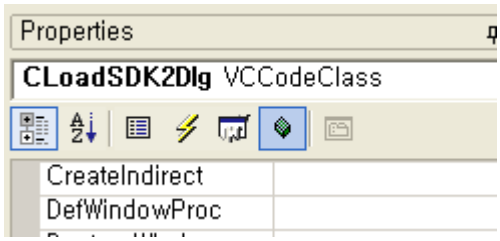


그림 3-23 Overrides 선택

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

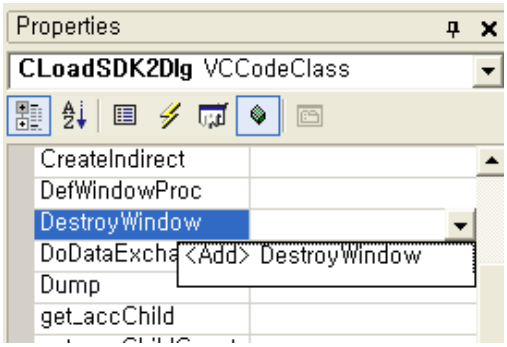


그림 3-24 Destroy Window 함수 추가

(생성한 프로젝트 이름)+Dlg 클래스의 멤버함수인 DestroyWindow()에 'cmmUnloadDll();'을 추가합니다. 'cmmUnloadDll()'함수를 추가하면 윈도우가 종료 될 때 자동으로 동적으로 연결된 CMMSDK 가 함께 해제됩니다.

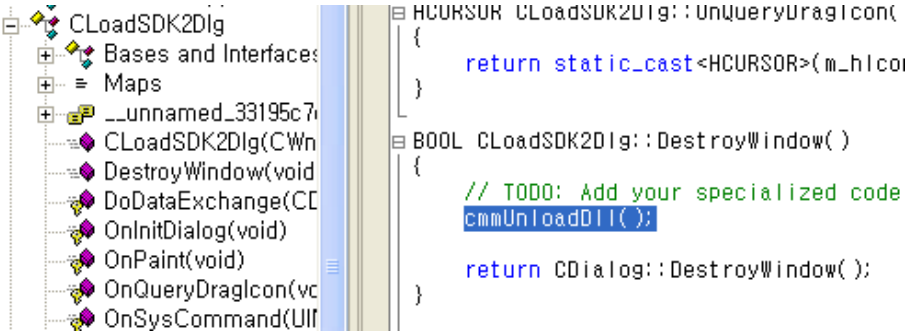


그림 3-25 UnloadDll 추가

CMMSDK 의 라이브러리 방식인 DLL 의 이점은 어떤 것이 있습니까?

예를 들어 드리겠습니다. 일반적인 C 나 C++ 개발환경에서 기본적인 Standard Input/Output Header 파일을 포함해서 사용합니다. 여기서 분명, 고객(顧客)님들께서 잘 아시는 <stdio.h> 인터페이스 파일을 사용하게 됩니다.

하지만 이러한 stdio.h 에 정의된 함수들을 우리가 다 사용하는 것은 아닙니다. 그러나 컴파일러는 이러한 함수의 내용들을 실제 고객(顧客)님께서 개발하시는 응용프로그램에 컴파일 시간(엄밀히 말하면 Linking 시간)에 모두 포함시켜버립니다. 이러한 형태는 실제 사용되지 않는 함수들 때문에 실행파일의 크기가 커져버리며, 그만큼 메모리를 많이 차지하게 됩니다.

DLL 은 이러한 라이브러리를 실행시간 때에 연결 시키는 것을 가능하게 합니다. 즉, 필요한 기능을 실행 시에 연결시킬 수 있도록 만들어 놓은 것이 DLL 이며, 각 응용프로그램의 프로세스(Process) 의 코드와 데이터들은 메모리 영역의 각기 별도의 영역을 갖지만 CMMSDK 부분과 공유 데이터 영역은 단지 한 번만 올라왔을 뿐이고 그 이후부터는 공유해서 사용합니다.
DLL 라이브러리는 함수를 다른 프로세스와 공유해서 사용하기 위해 개발되었으며, ㈜커미조아 CMMSDK 자체는 공유영역의 메모리에 올라가서 상주하게 됩니다. 인텔社의 80386 이상의 CPU 에서는 프로세서(Processor) 자체에서 공유 영역을 구현할 수 있는 특수한 기법을 제공합니다.

3.3.3 Visual C++ 8.x 개발자를 위한 안내

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

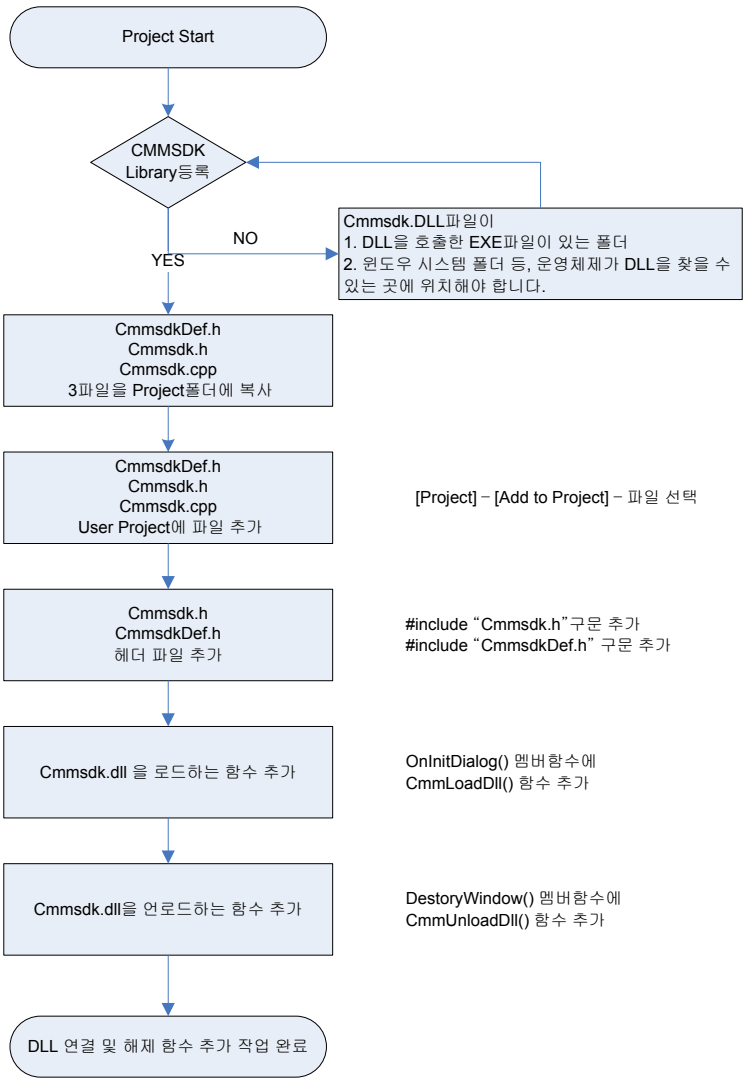


그림 3-26 Visual Studio 8.x 에서의 CMMSDK 사용 순서도

Microsoft 社의 Visual Studio 2005(이하 VS2005)를 실행합니다. 메뉴에서 [File]->[New]->[Project]를 선택하여 새로운 프로젝트를 시작합니다.

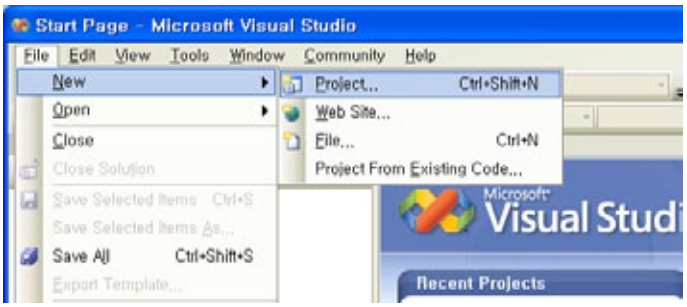


그림 3-27 새로운 프로젝트 생성 시작

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

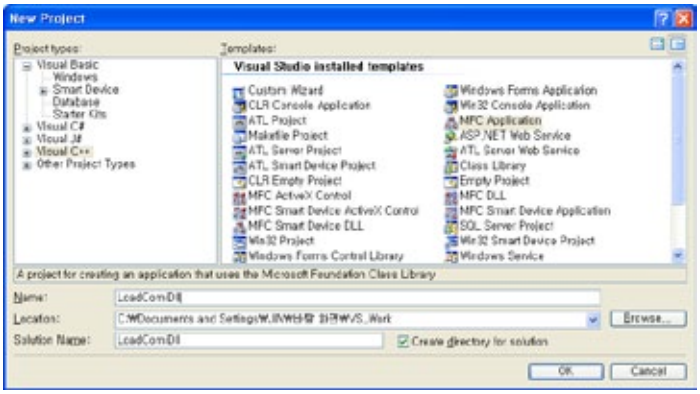


그림 3-28 새 프로젝트 옵션 선택화면

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

그림 3-29 MFC Application Wizard 의 Overview 화면

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

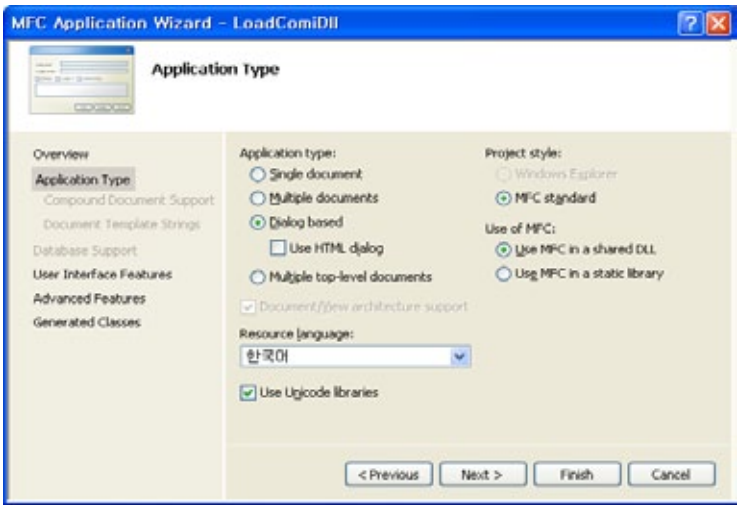


그림 3-30 MFC Application Wizard 의 Application Type 화면

VC++ 용 인터페이스 정의 파일인 Cmmsdk.h, Cmmsdk.cpp, CmmsdkDef.h 파일을 신규로 생성한 프로젝트 폴더로 복사합니다.

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

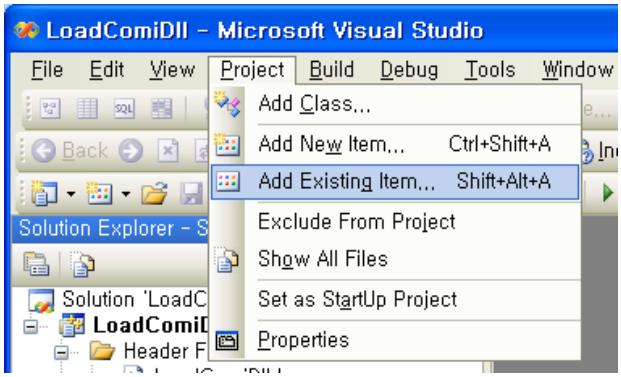


그림 3-31 메뉴에서 Add Existing Item 선택 화면

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

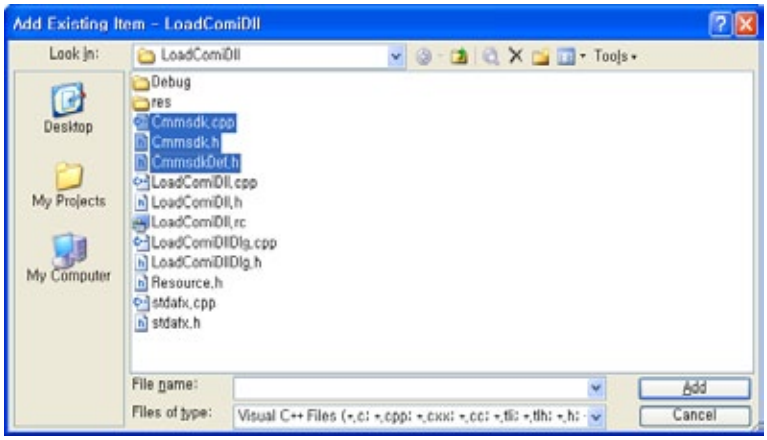


그림 3-32 새로 추가할 파일들 선택 화면

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

파일의 가장 위쪽에 파일의 가장 위쪽에 인터페이스 파일을 추가합니다.

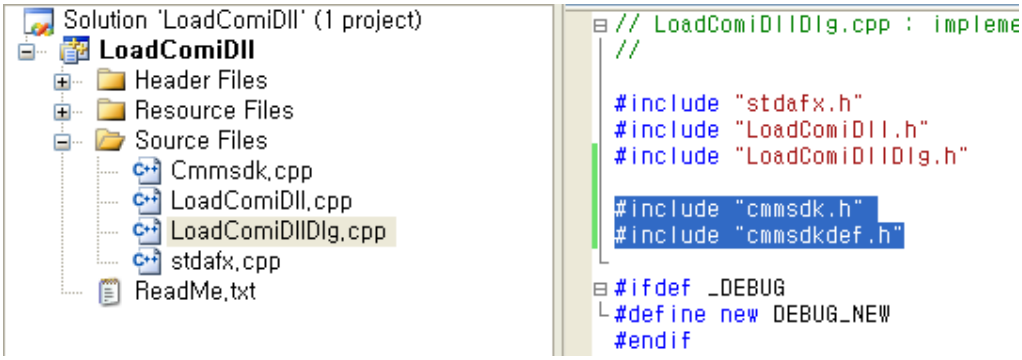


그림 3-33 사용자 생성 CPP 파일에 헤더 추가 화면

(생성한 프로젝트 이름)+Dlg.cpp 파일의 OnInitDialog() 함수 내부의 "TODO"아래에 "cmmLoadDll();"를 추가합니다.

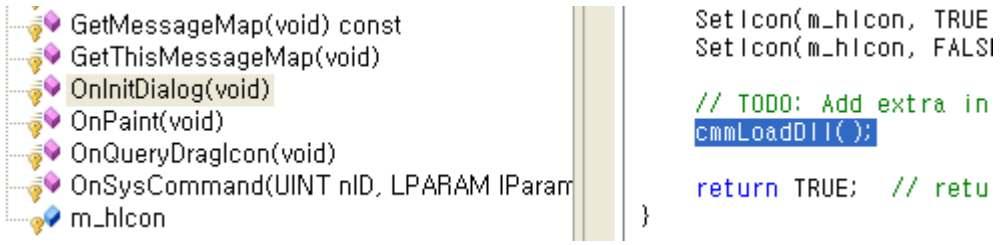


그림 3-34 LoadDll 추가

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

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

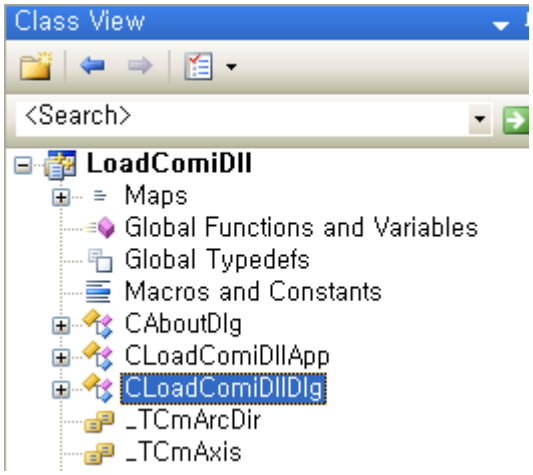


그림 3-35 사용자 생성 Dialog Class 선택

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

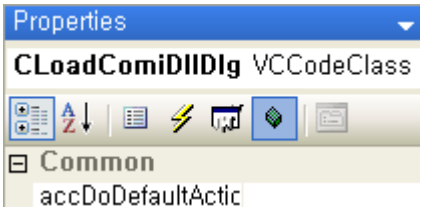


그림 3-36 Overrides 선택

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

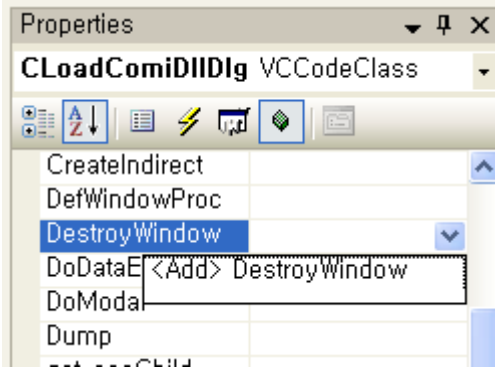


그림 3-37 Destroy Window 함수 추가

([생성한 프로젝트 이름]+Dlg) 클래스의 멤버함수인 DestroyWindow()에 'cmmUnloadDll();'을 추가합니다. 'cmmUnloadDll()'함수를 추가하면 윈도우가 종료 될 때 자동으로 DLL 도 해제됩니다.

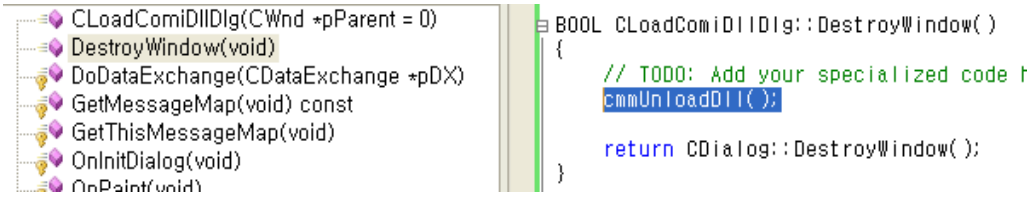


그림 3-38 UnloadDll 함수 추가

3.3.4 Borland C++ Builder 개발자를 위한 안내

Borland C++ Builder 는 해당 개발 환경 버전인 BCB 5, BCB 6 및 BDS 2006 버전에서 CMMSDK의 인터페이스 연결 방법이 매우 유사하기 때문에 공통적인 부분으로서 안내를 해드립니다.

전체 버전(Version)의 Borland C++ Builder 에서 CMMSDK를 사용하시려면 다음의 절차를 통해 안내 받으시기 바랍니다.

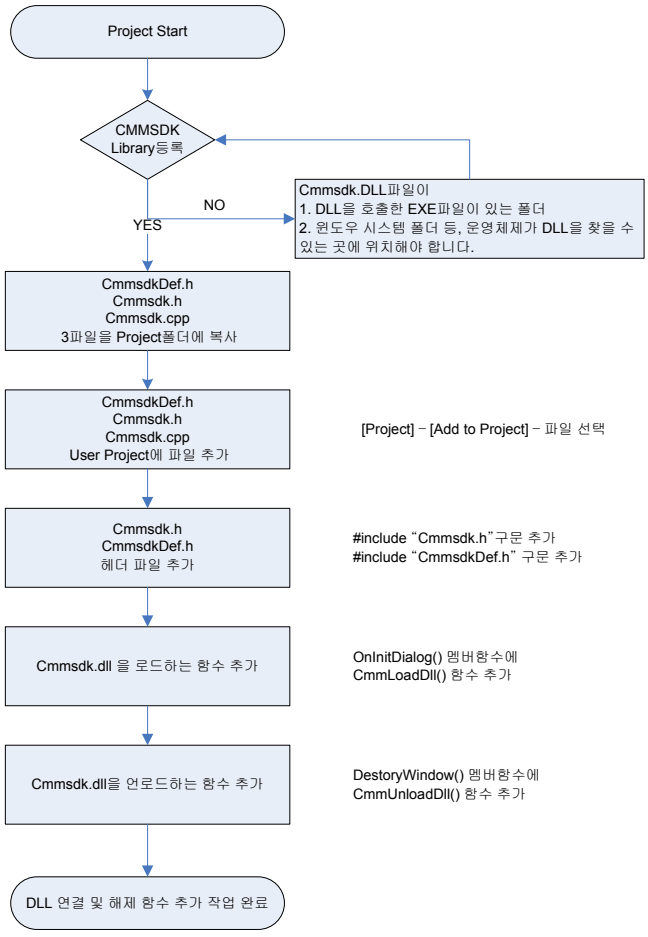


그림 3-39 Borland C++ Builder 에서 CMMSDK 사용 순서도

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

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

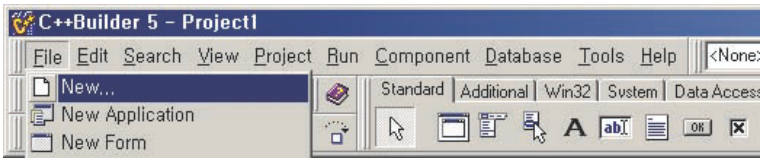


그림 3-40 BCB 5 에서 새로운 프로젝트 생성

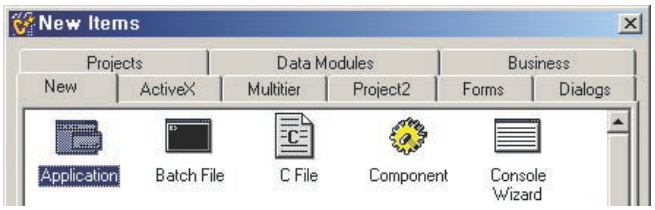


그림 3-41 BCB 5 에서 새로운 프로젝트 생성

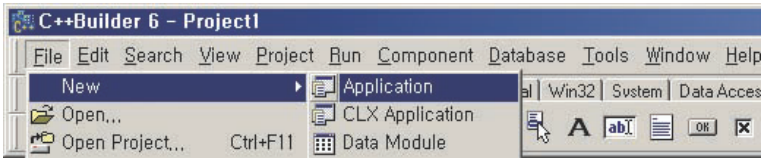


그림 3-42 BCB 6 에서 새로운 프로젝트 생성

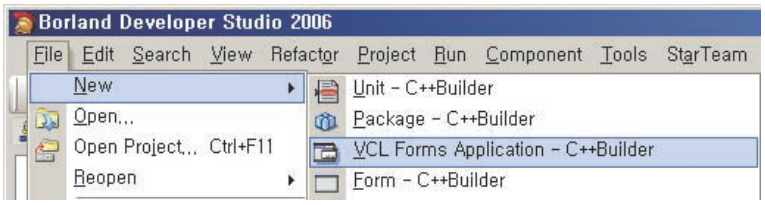


그림 3-43 BDS 2006 에서 새로운 프로젝트 생성

Borland C++ 및 VC++ 용 공용 인터페이스 정의 파일인 Cmmsdk.h, Cmmsdk.cpp, CmmsdkDef.h 파일을 신규로 생성한 프로젝트 폴더로 복사합니다.

그림 3-44 CMMSDK 사용시 공통으로 사용되는 파일

	참고 사항
안내	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.

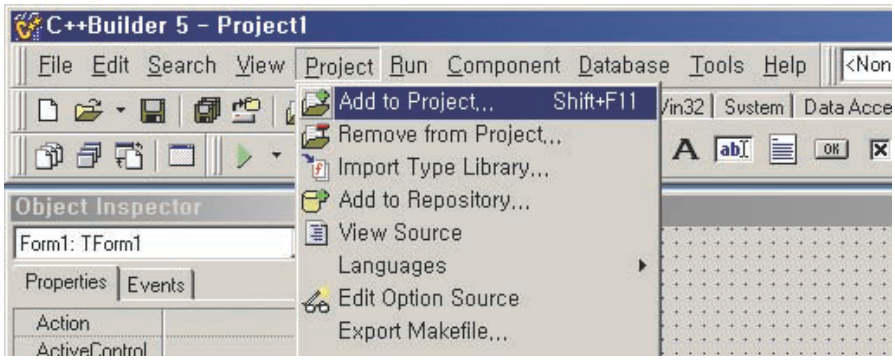


그림 3-45 C++ Builder 에서 프로젝트에 파일 추가 단계 1

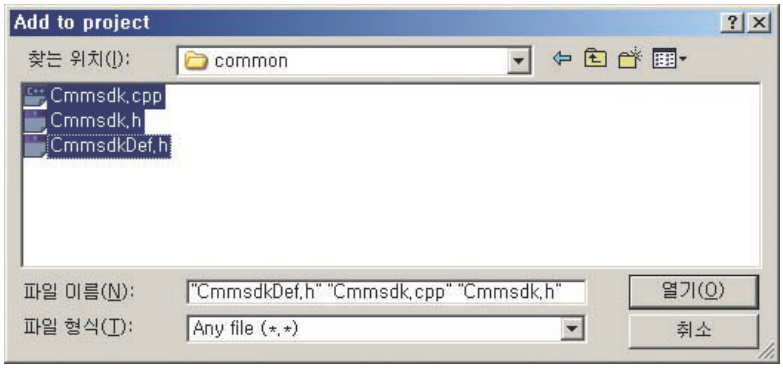


그림 3-46 C++ Builder 에서 프로젝트에 파일 추가

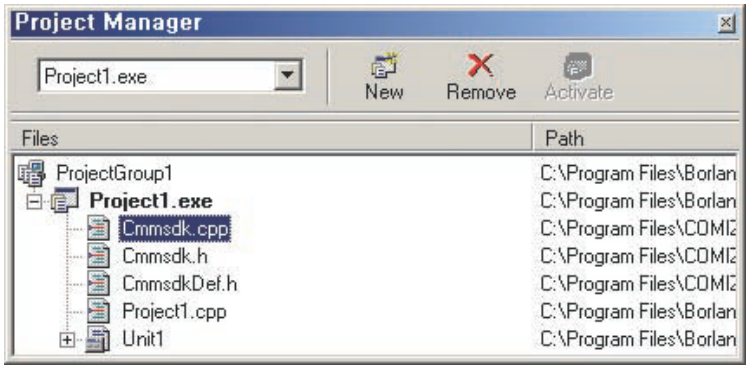


그림 3-47 Project Manager 에서 추가된 파일 확인(確認)

라이브러리 함수를 사용하고자 하는 대상 구현 부 응용프로그램 파일에 인터페이스 파일을 선언합니다.

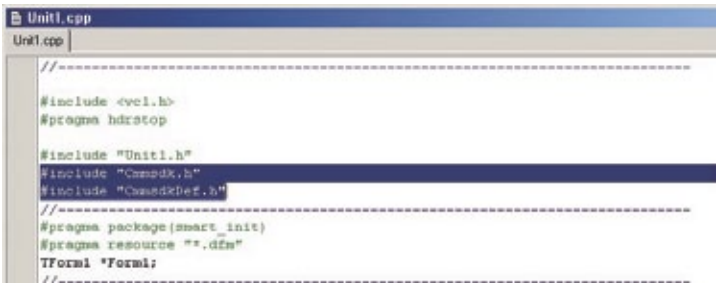


그림 3-48 라이브러리 사용시 필요한 헤더 파일 선언

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

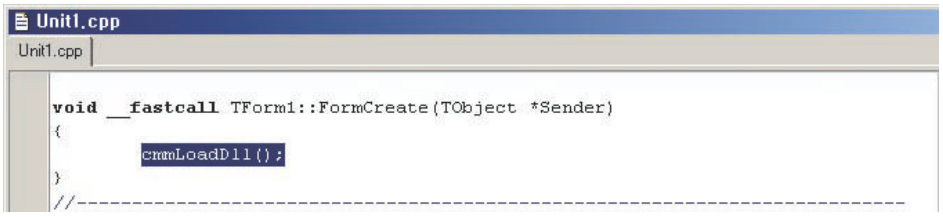


그림 3-49 Cmmsdk.dll 파일 로드

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

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

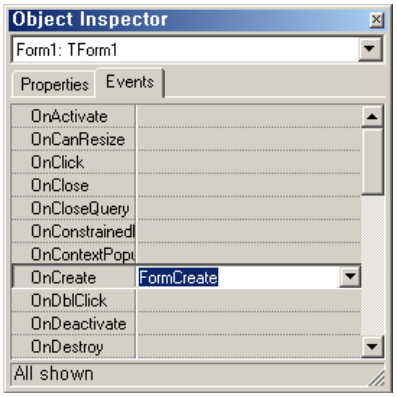
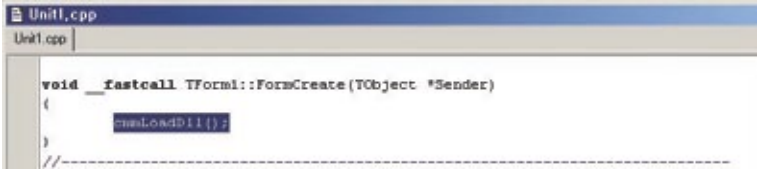


그림 3-50 OnCreate Event 추가하여 FormCreate 함수와 연결

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



[그림 12] FormCreate 함수에 cmmLoadDll 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 cmmUnloadDll()이라는 함수를 통해서 이루어 집니다. cmmUnloadDll()을 추가 하는 방법은 다음과 같습니다.

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

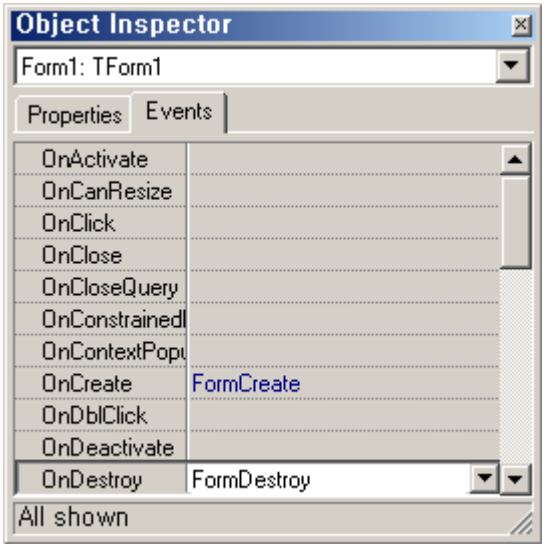


그림 3-51 응용프로그램의 종료시 DLL 이 명시적으로 UnLoad 될 수 있도록 OnDestroy Event 와 함수의 연결

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

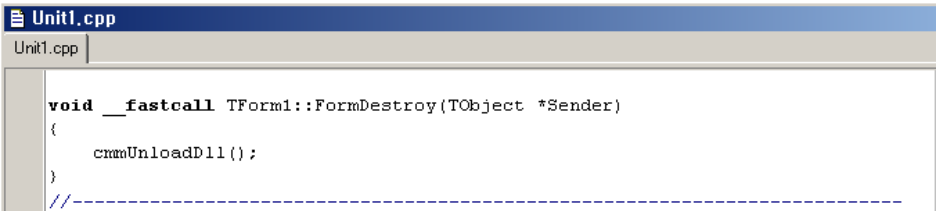


그림 3-52 FormDestroy 함수를 통하여 명시적인 UnLoadDll 함수 구현

3.3.5 Borland Delphi 개발자를 위한 안내

Borland Delphi 는 해당 개발 환경 버전인 Delphi 5, Delphi 6 및 Delphi 7, BDS 2006 버전에서 CMMSDK의 인터페이스 연결 방법이 매우 유사하기 때문에 공통적인 부분으로서 안내를 해드립니다.

전체 버전(Version)의 Delphi 에서 CMMSDK를 사용하시려면 다음의 절차를 통해 안내 받으시기 바랍니다.

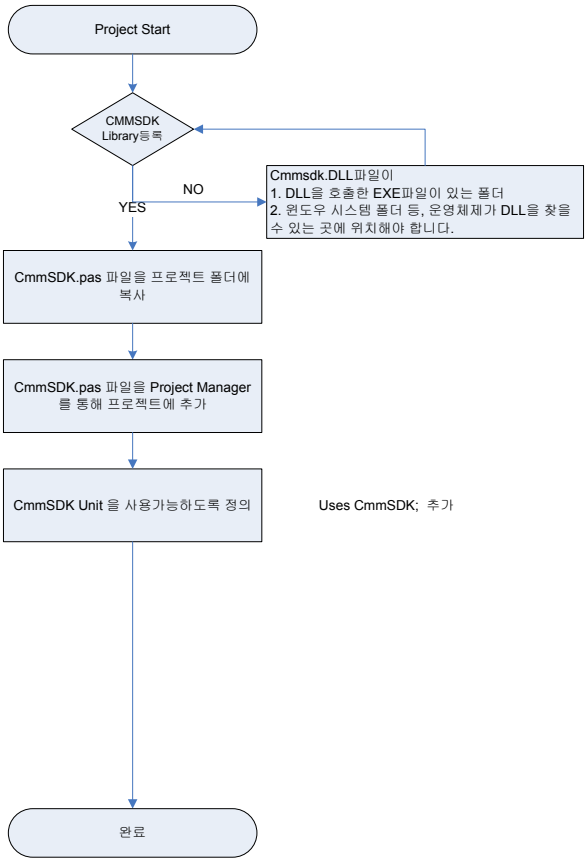


그림 3-53 Borland Delphi 에서 CMMSDK 사용 순서도

Delphi 모션 응용프로그램을 개발에 필요한 환경인 Borland Delphi 를 실행합니다.

그림 3-54 Borland 社의 Delphi 5 화면

이 안내는 Delphi 5 를 기준으로 설명드리겠습니다. 만약 안내드리는 도중에 다른 개발 환경과 구분이 되어야 할 내용은 별도로 설명 드리겠습니다.

프로젝트 시작 전에 ㈜ 커미조아 CMMSDK Delphi 용 공용 인터페이스 파일을 프로젝트 폴더에 복사합니다. 이 파일은 ㈜ 커미조아 CMMSDK 의 DLL(Dynamic Link Library) 와 고객(顧客)님의 응용프로그램과의 인터페이스를 정의 하여 놓은 파일입니다.

델파이(Delphi)는 명시적으로 프로젝트파일 간의 상호 변환이 필요가 없습니다. 따라서 Delphi 5 나, 6 그리고 7 버전에서 몇가지 기본적인 컴포넌트에 기반한 내용을 제외한 부분들을 그대로 사용할 수 있습니다. ㈜ 커미조아 CMMSDK 에서는 Delphi (5/6/7) 에 대한 풍부한 예제를 제공하고 있습니다.

새로운 프로젝트를 시작하기 위해서, “File” 메뉴의 “New” 에 대한 항목을 클릭하여, 새로운 응용프로그램 개발을 시작합니다.

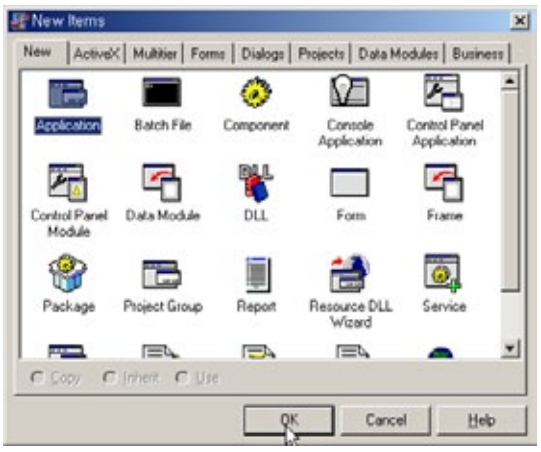


그림 3-55 Delphi 5 의 프로젝트 시작

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

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

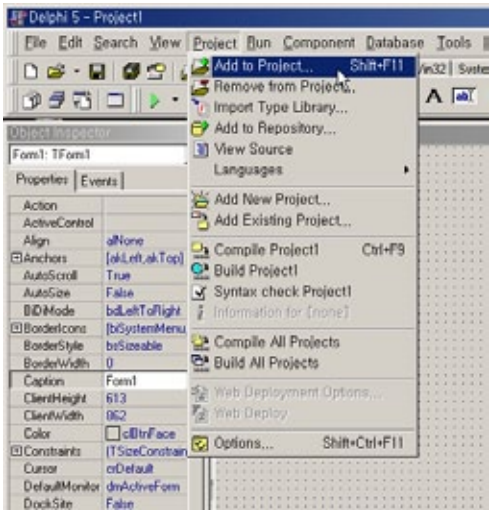


그림 3-56 Delphi 용 인터페이스 파일 추가

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

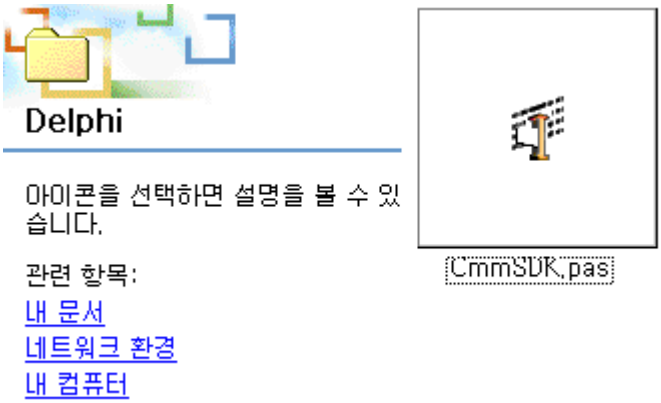


그림 3-57 Delphi 용 인터페이스 파일

Project Manager 를 통해 확인(確認)해 보면 CmmSDK.PAS 파일이 프로젝트에 등록된 것을 확인(確認)할 수 있습니다.

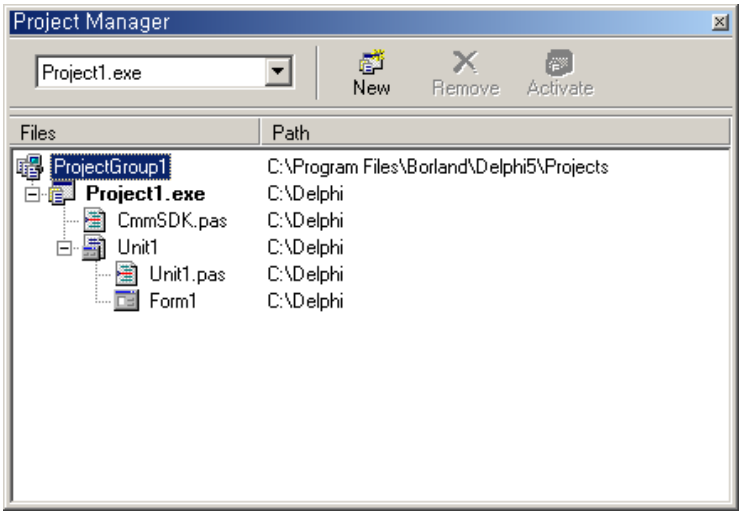


그림 3-58 Delphi 용 인터페이스 파일을 프로젝트 매니저에 추가

보충	Delphi 의 CMMSDK 인터페이스 파일의 사용에 대한 부가 안내
	저희 ㈜ 커미조아의 CMMSDK 의 CmmSDK.Pas 파일은 다른 개발환경(VC++, C++ Builder)와 달리 DLL 의 명시적인 로드와 언로드가 자동으로 이루어집니다. 이것은 델파이가 가지고 있는 Initialization 과 Finalization 을 이용한 것으로 프로젝트에 별다른 LoadDll 함수 호출 없이 자동으로 DLL 이 로드가 됩니다. 또한 응용프로그램 종료 시에 UnloadDll 이 명시적으로 되지 않아도, 자동적으로 응용프로그램이 종료 시에 UnloadDll 이 실행됩니다.

실제 Unit1.pas 혹은 구현 부의 코드를 에디터를 통해 확인(確認)합니다.

```
unit Unit1;

interface

uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs,
  StdCtrls;

type
  TForm1 = class(TForm)

  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  Form1: TForm1;

implementation
////////////////////////////////////////////////////////////////
// COMZIOA SDK Library 를 위한 인터페이스 파일을 사용합니다.
uses CmmSDK;
////////////////////////////////////////////////////////////////

{$R *.DFM}


end.
```

그림 3-59 uses 구문을 통해 CMMSDK Unit 사용

위와 같이 implementation 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.

3.3.6 Visual Basic 개발자를 위한 안내

Visual Basic 6.0 은 마이크로소프트의 컴포넌트 기반 응용프로그램 개발을 위해 태어난 뛰어난 개발 환경입니다. CMMSDK는 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 CMMSDK 인터페이스 파일인 CmmSDK.BAS 파일과 CMMSDK 파일 'Cmmsdk.dll' 파일을 반드시 프로젝트 디렉토리에 복사해주시기 바랍니다.

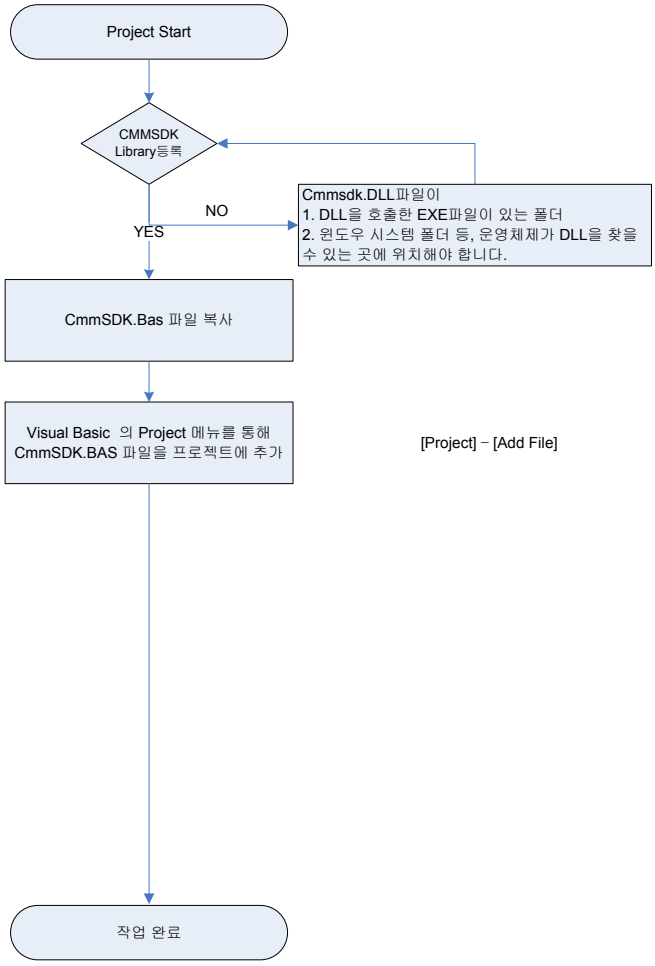


그림 3-60 Visual Basic 에서 CMMSDK 사용 순서도

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

그림 3-61 새로운 프로젝트 생성

	Visual Basic 에 대한 CMMSDK 지원 사항은 어떠합니까?
안내	CMMSDK 는 .NET 환경의 Visual Basic 까지 지원을 하고 있습니다. 특히 미리 구성된 CMMSDK 인터페이스 파일은 간단히 프로젝트에 추가만 하시면, 바로 CMMSDK 를 사용하실 수 있도록 도움을 드리고 있습니다. 만약, Visual Basic 사용 고객(顧客)님께서 CMMSDK 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.

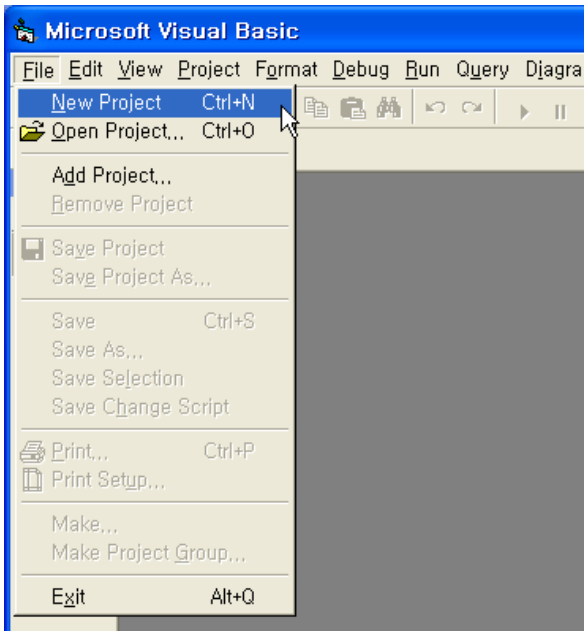


그림 3-62 신규 프로젝트의 시작

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

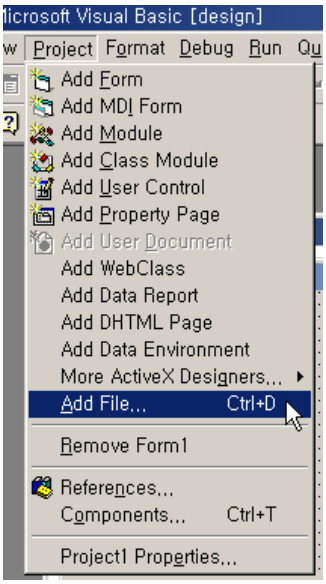


그림 3-63 프로젝트에 인터페이스 파일을 추가하기 위한 과정

그림 3-64 프로젝트에 추가 대상이 되는 CmmSDK.BAS 파일

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

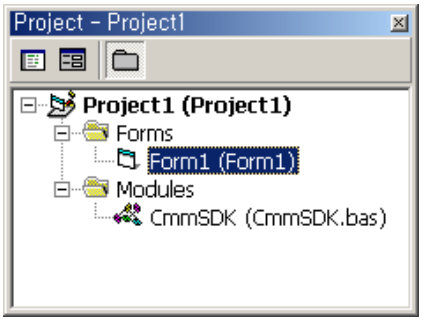


그림 3-65 프로젝트에 모듈로 추가된 CmmSDK.BAS 파일

추가된 인터페이스 파일을 통해 다음과 같이 실제 응용프로그램 구현이 가능합니다.

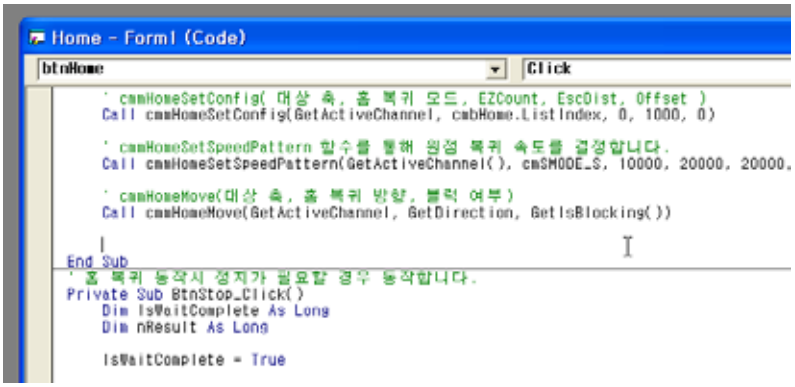


그림 3-66 CMMSDK 를 통한 응용프로그램 구현

보충	Visual Basic 의 명시적인 CMMSDK 인터페이스에 대한 부가 안내
	CMMSDK 에 포함된 Visual Basic 용 인터페이스 파일은 별도의 DLL 라이브러리(CMMSDK)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

CMMSDK Introduction

다년간의 보다 강력하고 편리한 라이브러리 기술개발을 통해 자신 있게 제공하여 드리는 CMMSDK 는 편리한 함수명의 규칙(Rule)을 통해 사용자 편의성을 극대화 하였습니다. ㈜커미조아 모션 라이브러리의 핵심 결정판인 CMMSDK 의 최신 기능과 숙련된 모션 제어 기술은 결코 흉내낼 수 없는 커미조아의 기술입니다. 지금 확인(確認)하십시오.

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

4 CMMSDK 소개

4.1 함수의 명명 규칙

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

cmmLoadDll(), cmmUnloadDll(), cmmGnDeviceLoad(), cmmGnDeviceUnload(),...

그리고 "cmm" 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): cmmGnDeviceLoad(), cmmGnSetServoOn(), ...

○. 환경설정 함수들 (Cfg): cmmCfgSetMioProperty(), cmmCfgSetOutMode(), ...

○. 원점복귀 관련 함수들 (Home): cmmHomeMove(), cmmHomeSetConfig(), ...

○. 단축구동 관련 함수들 (Sx): cmmSxSetSpeedRatio(), cmmSxMove(), ...

○. 다축구동 관련 함수들 (Mx): cmmMxMove(), cmmMxStop(), ...

○. 보간구동 관련 함수들 (Ix): cmmIxMapAxes(), cmmIxLine(), ...

4.2 데이터형 표기

당사의 CMMSDK 인터페이스는 매뉴얼에서 명시한 윈도우 표준 Dynamic Link Library 를 지원하는 어떠한 개발환경에서도 사용 가능합니다. 하지만 데이터형에 대한 이름은 개발환경에 따라서 서로 다릅니다. 따라서 본 매뉴얼에서는 데이터의 형 표기를 표 8 과 같이 통일하여 표기합니다. 이에 대한 각 컴파일러의 대응되는 데이터형 표기는 표 8 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 부호 있는 정수 표현형	long	Long (ByVal)	LongInt	Int
VT_PI4	4 바이트 부호 있는 정수 변수의 주소 값 (포인터) 또는 배열주소 표현형	long *	Long (ByRef)	PLongInt	Int[]
VT_R4	4 바이트 부호 있는 실수 표현형	float	Double (ByVal)	Double	Float
VT_PR4	4 바이트 부호 있는 실수 변수의 주소 값(포인터) 또는 배열주소 표현형	float *	Double (ByRef)	PDouble	float[]

VT_R8	8 바이트 부호 있는 실수 표현형	double	Double (ByVal)	Double	double
VT_PR8	8 바이트 부호 있는 실수변수의 주소 값(포인터) 또는 배열주소 표현형	double *	Double (ByRef)	PDouble	double[]
VT_STR	선형 메모리 상의 문자열 선두 주소를 지시하는 4 바이트 주소 표현형	char *	String (ByVal)	PChar	String

표 8 언어독립적 데이터 표기 및 각 언어별 대응 데이터 형

Chapter

5

General Functions

㈜커미조아는 CMMSDK를 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.

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

5 General Functions

5.1 함수 요약

Summary of Functions
❒ BOOL cmmLoadDll([none] VT_EMPTY) 라이브러리를 사용자(使用者) 응용프로그램의 사용을 위해 로드(Load) 합니다.
❒ VT_EMPTY cmmUnloadDll([none] VT_EMPTY) 라이브러리를 사용자(使用者) 응용프로그램의 사용 종료(終了)를 위해 언로드(Unload) 합니다.
❒ VT_I4 cmmGnDeviceLoad([in] VT_I4 IsResetDevice, [out] VT_PI4 NumAxes) 라이브러리가 로드된 상태에서 장치를 로드(Load) 하는 역할을 합니다.
❒ VT_I4 cmmGnDeviceUnload([[none] VT_EMPTY) 라이브러리가 로드된 상태에서 장치를 언로드(Unload) 하는 역할을 합니다.
❒ VT_I4 cmmGnDeviceIsLoaded([out] VT_PI4 IsLoaded) 라이브러리가 로드된 상태를 확인(確認) 할 수 있습니다.
❒ VT_I4 cmmGnDeviceReset([none] VT_EMPTY) 장치를 리셋(Reset) 하는 역할(役割)을 수행합니다.

5.2 함수 설명

NAME

cmmLoadDll
- 라이브러리(Library) 로드

INFORMATION

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

SYNOPSIS

❒ BOOL cmmLoadDll ([none] VT_EMPTY)

DESCRIPTION

CMMSDK 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 CMMSDK 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 리턴 값은 불 형(Boolean Type) 을 가지고 있습니다.

Value	Meaning
FALSE	DLL 을 로드하는데 실패하였음을 의미합니다.
TRUE	DLL 을 성공적으로 로드하였음을 의미합니다.

SEE ALSO

cmmUnloadDll

EXAMPLE

C/C++

```
#include "Cmmsdk.h"
#include "CmmsdkDef.h"

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


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

Visual Basic

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

```
Delphi

//* Delphi 에서는 명시적인 DLL 로드가 필요없습니다.
//* 단, 처음 선언 시에 다음과 같은 내용이 포함되어야 합니다.
////////////////////////////////////////////////////////////
// COMZIOA SDK Library 를 위한 인터페이스 파일을 사용합니다.
uses CmmSDK;
////////////////////////////////////////////////////////////
procedure TForm1.OnCreate(Sender: TObject);
var
        g_nAxes : LongInt;

begin

if ( cmmGnDeviceLoad(cmTRUE, @g_nAxes) <> cmERR_NONE ) then
begin

        cmmErrShowLast(Form1.Handle);
        exit;
end;

end;
```

NAME

cmmUnloadDll
- 라이브러리(Library) 로드 해제(解除)

INFORMATION

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

SYNOPSIS

❒ VT_EMPTY cmmUnloadDll ([none] VT_EMPTY)

DESCRIPTION

CMMSDK 를 고객(顧客)님의 응용프로그램의 메모리 공간에서 해제합니다. 이 의미는 이 함수가 호출되는 순간 ㈜커미조아의 CMMSDK 는 고객(顧客)님의 응용프로그램에서 제거됩니다. 이 함수는 고객(顧客)님의 전체 프로그램에서 CMMSDK 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

cmmLoadDll

EXAMPLE

C/C++

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


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

Visual Basic

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

Delphi

```
//* Delphi 에서는 명시적인 DLL 로드가 필요없습니다.
//* 단, 처음 선언 시에 다음과 같은 내용이 포함되어야 합니다.
////////////////////////////////////////////////////////////
// COMZIOA SDK Library 를 위한 인터페이스 파일을 사용합니다.
uses CmmSDK;
////////////////////////////////////////////////////////////
procedure TForm1.OnCreate(Sender: TObject);
var
```

```
        g_nAxes : LongInt;

begin

if ( cmmGnDeviceLoad(cmTRUE, @g_nAxes) <> cmERR_NONE ) then
begin

        cmmErrShowLast(Form1.Handle);
        exit;
end;

end;
```

NAME

cmmGnDeviceLoad
– 장치(裝置) 로드(Load)

INFORMATION

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

SYNOPSIS

❒ VT_I4 cmmGnDeviceLoad ([in] VT_I4 IsResetDevice, [out] VT_PI4 NumAxes)

DESCRIPTION

시스템에 설치된 하드웨어 장치를 로드하고 장치를 초기화합니다. 이 함수는 CMMSDK 의 다른 함수가 호출되기 전에 반드시 한번은 수행되어야 합니다. 일반적으로 프로그램의 시작부분에서 수행해주면 됩니다.
CMMSDK 에서는 디지털 입출력 보드와 모션 보드를 통합해서 관리하고 있습니다. 본 함수는 시스템에 설치되어 있는 ㈜ 커미조아 디지털 입출력 보드(SD4xx) Series 와 모션 전 제품을 초기화합니다.

PARAMETER

▶ IsResetDevice : 장치 초기화를 수행할지를 결정합니다.

Value	Meaning
0 또는 cmFALSE	장치 로드 시에 장치를 초기 값으로 초기화하지 않습니다.
1 또는 cmTRUE	장치 로드 시에 장치를 리셋(Reset)한 후에 장치의 여러 가지 속성들을 기본값으로 초기화합니다. 또한, 이 매개변수가 TRUE 설정되면, cmmGnDeviceUnload() 나, CMMSDK 의 DLL Unload 시에 자동으로 모든 모션 이송을 정지(停止)시킵니다.

▶ NumAxes : 이 매개 변수를 통하여 실제로 로드(Load)된 모션 제어 축의 개수를 반환합니다. 단, 이 매개 변수에 NULL 을 전달하면 제어 축 수를 반환하지 않습니다.

RETURN VALUE

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

㈜ 커미조아의 디바이스 초기화 값은 어떤 값으로 설정됩니까?

cmmGnDeviceLoad 함수를 통해 장치 초기화 시의 각 속성별로 사용되는 기본값은 "Appendix B"를 참조하시기 바랍니다.

SEE ALSO

cmmGnDeviceUnload

EXAMPLE

C/C++

```
#include “Cmmsdk.h”
#include “CmmsdkDef.h”

void ProgramInitial(void)
{
	long nNumAxes = 0;

	if ( cmmLoadDll() != 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. */

OutputDebugString(“DLL 로드에 실패하였습니다”);
// 이후 적절한 에러처리를 해주시기 바랍니다.
	}

	if ( cmmGnDeviceLoad(cmTRUE, &nNumAxes) != cmERR_NONE ){

	// MS VC++ 에서는 아래와 같이 에러 원인을 화면에 표시할 수 있습니다.
	cmmErrShowLast(GetSafeHwnd());

	// Borland C++ 계열에서는 아래와 같이 에러원인을 화면에 표시할 수 있습니다.
	cmmErrShowLast(Form1->Handle);
	}
} /* ProgramInitial(void) 함수의 끝 */
```

Visual Basic

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

Private Sub Form_Load()

Dim IRetVal As Long

Dim nTotalAxes As Long

IRetVal = cmmGnDeviceLoad(cmTRUE, nTotalAxes)

If  IRetVal <> cmERR_NONE Then

	MsgBox(“cmmGnDeviceLoas has been failed”)

End If

End Sub
```

Delphi

```
//* Delphi 에서는 명시적인 DLL 로드가 필요없습니다.
//* 단, 처음 선언 시에 다음과 같은 내용이 포함되어야 합니다.
////////////////////////////////////////////////////////////
// COMZIOA SDK Library 를 위한 인터페이스 파일을 사용합니다.
uses CmmSDK;
////////////////////////////////////////////////////////////
procedure TForm1.OnCreate(Sender: TObject);
var
	g_nAxes : LongInt;

begin

if ( cmmGnDeviceLoad(cmTRUE, @g_nAxes) <> cmERR_NONE ) then
begin
```

```
        cmmErrShowLast(Form1.Handle);
        exit;
end;

end;
```

NAME

cmmGnDeviceUnload
– 장치(裝置) 언로드(Unload)

INFORMATION

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

SYNOPSIS

❒ VT_I4 cmmGnDeviceUnload ([none] VT_EMPTY)

DESCRIPTION

시스템에 설치된 하드웨어 장치를 언로드하고 장치사용을 종료합니다. 이 함수는 CMMSDK 의 함수 사용을 종료하기 위해 명시적으로 호출되어야 합니다. 일반적으로 프로그램의 종료부분에서 수행해주면 됩니다. CMMSDK 에서는 디지털 입출력 보드와 모션 보드를 통합해서 관리하고 있습니다. 본 함수는 시스템에 설치되어 있는 ㈜ 커미조아 디지털 입출력 보드(SD4xx) Series 와 모션 전 제품 사용을 종료합니다.

RETURN VALUE

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

㈜ 커미조아의 디바이스 초기화 값은 어떤 값으로 설정됩니까?

cmmGnDeviceLoad 함수를 통해 장치 초기화 시의 각 속성별로 사용되는 기본값은 "Appendix B"를 참조하시기 바랍니다.

SEE ALSO

cmmGnDeviceLoad

EXAMPLE

```
//* cmmGnDeviceLoad 예제를 참고하여 주시기 바랍니다.
```

NAME

cmmGnDeviceIsLoaded
– 장치(裝置) 로드 상태확인(狀態確認)

INFORMATION

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

SYNOPSIS

❒ VT_I4 cmmGnDeviceIsLoaded ([out] VT_PI4 IsLoaded)

DESCRIPTION

이 함수는 지정된 하드웨어 장치가 로드된 상태를 반환합니다. 사용자는 cmmGnDeviceLoad() 함수를 통하여 디바이스를 로드하며, 이 때의 GnDeviceLoad() 함수의 반환값을 확인(確認)함으로써, 디바이스 로드의 성공/실패 여부를 알 수 있으므로 일반적인 경우에는 cmmGnDeviceIsLoaded() 함수는 사용하지 않으셔도 됩니다.

PARAMETER

▶ IsLoaded : 현재 디바이스가 로드되어 있는지 여부

Value	Meaning
0 또는 cmFALSE	장치가 정상적으로 로드 되지 않았습니다.
1 또는 cmTRUE	장치가 정상적으로 로드 되었습니다.

RETURN VALUE

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

SEE ALSO

cmmGnDeviceLoad

EXAMPLE

C/C++

```
#include "Cmmsdk.h"
#include "CmmsdkDef.h"

long TestRunLoadStatus(void)
{
        long IsLoaded = 0;

        if (cmmGnDeviceIsLoaded(&IsLoaded) != cmERR_NONE )
        {
                printf("cmmGnDeviceIsLoaded has been failed");
                return FALSE;
        }
        if ( IsLoaded == cmTRUE)
                printf("Device load has been completed");
```

```
        else
                printf("Device load hasn't been completed");

return IsLoaded;
}
```

Visual Basic

```
Function TestRunLoadStatus(void) As Long

        Dim IRetVal As Long

        IRetVal = cmmGnDeviceIsLoaded(TestRunLoadStatus)

        If IRetVal <> cmERR_NONE Then
            MsgBox ("cmmGnDeviceIsLoaded had been failed")
        End If

        If TestRunLoadStatus = cmTRUE Then
            MsgBox ("Device load has been completed")
        Else
            MsgBox ("Device load hasn't been completed")
        End If

End Function
```

Delphi

```
function TestRunLoadStatus(): LongInt;
var
        IsLoaded : LongInt;
begin
        IsLoaded := 0;
        if (cmmGnDeviceIsLoaded(@IsLoaded) <> cmERR_NONE ) then
        begin
                Writeln('cmmGnDeviceIsLoaded has been failed');
                Writeln('cmmGnDeviceIsLoaded has been failed');
                Result := cmFALSE;
        end;

        if ( IsLoaded = cmTRUE) then
        begin
                Writeln('Device load has been completed');
        end
        else
        begin
                Writeln('Device load hasn`t been completed');
        end;

Result := IsLoaded;

end;
```

NAME

cmmGnDeviceReset
– 장치(裝置) 초기화

INFORMATION

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

SYNOPSIS

❒ VT_I4 cmmGnDeviceReset ([None] VT_EMPTY)

DESCRIPTION

장치를 리셋한 후에 초기화합니다. 일반적으로 cmmGnDeivceLoad 함수에서 장치초기화를 수행하므로 초기에 장치를 따로 리셋할 필요는 없습니다. 이 함수는 장치가 비정상적으로 동작하여서 장치를 리셋하고 다시 초기화하고자 할 때 사용할 수 있습니다. 이때 각 속성별로 사용되는 초기값은 부록(Appendix) 편을 참조하시기 바랍니다.

RETURN VALUE

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

SEE ALSO

cmmGnDeviceLoad

Chapter

6

Etc General Functions

CMMSDK 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 편의기능과 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 있어 모든 내용이 반드시 습득(習得) 될 필요는 없지만, 모션 제어를 위해 꼭 필요한 내용의 기타 General 기타 General 함수 편으로 구성되어 있습니다. 기본적인 서보 ON 출력 신호부터, Visual Basic 사용자를 위해 사용자를 위해 구성된 편리한 '비트 연산' 함수까지, 제공(提供)하여 드립니다.

6 Etc General Functions

6.1 함수 요약

Summary of Functions
❒ VT_I4 cmmGnInitFromFile([in] VT_STR szCmeFile) CME(COMIZOA Motion Environment) 파일을 통한 모션 통합 시스템 환경 초기화
❒ VT_I4cmmGnInitFromFile_MapOnly ([in] VT_STR szCmeFile, [in] VT_I4 nMapType) CME(COMIZOA Motion Environment) 파일을 통한 모션 통합 시스템 환경 초기화
❒ VT_I4 cmmGnSetServoOn ([in] VT_I4 Axis, [in] VT_I4 Enable) 서보 드라이브를 SERVO-ON 신호 출력을 인가(認可) 혹은 차단(遮斷) 합니다.
❒ VT_I4 cmmGnGetServoOn ([in] VT_I4 Axis, [out] VT_PI4 Enable) 서보 드라이브를 SERVO-ON 신호 출력(出力) 상태를 반환(返還)합니다.
❒ VT_I4 cmmGnPulseAlarmRes ([in] VT_I4 Axis, [in] VT_I4 IsOnPulse, [in] VT_I4 dwDuration, [in] VT_I4 IsWaitPulseEnd) 펄스 형태의 알람 리셋(Alarm Reset) 신호를 출력 및 제어(制御) 합니다.
❒ VT_I4 cmmGnSetAlarmRes ([in] VT_I4 Axis, [in] VT_I4 IsOn) 알람 리셋(Alarm Reset) 신호 출력(出力)을 제어합니다.
❒ VT_I4 cmmGnGetAlarmRes ([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 cmmGnSetSimulMode ([in] VT_I4 Axis, [in] VT_I4 IsSimulMode) 시뮬레이션 모드를 활성(活性) 혹은 비활성(非活性) 합니다.
❒ VT_I4 cmmGnGetSimulMode ([in] VT_I4 Axis, [out] VT_PI4 IsSimulMode) 시뮬레이션 모드의 설정 상태를 반환(返還)합니다.
❒ VT_I4 cmmGnPutInternalSTA([in] VT_I4 AxesMask) 소프트웨어적인 STA 신호를 발생(發生)합니다.
❒ VT_I4 cmmGnSetEmergency ([in] VT_I4 IsEnable, [in] VT_I4 IsDecStop) 소프트웨어적인 비상 상황을 설정(設定)합니다.
❒ VT_I4 cmmGnGetEmergency ([out] VT_PI4 IsEnabled) 소프트웨어적인 비상 상황 상태를 반환(返還)합니다.
❒ VT_I4 cmmGnBitShift ([in] VT_I4 Value, [in] VT_I4 ShiftOption, [out] VT_PI4 Result) 32 비트 정수형 데이터 값을 대상으로 비트 연산(Bit Operation) 을 수행합니다.

6.2 함수 설명

NAME

cmmGnInitFromFile
– CME 파일을 통한 환경(環境) 초기화

INFORMATION

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

SYNOPSIS

❒ VT_I4 cmmGnInitFromFile ([in] VT_STR szCmeFile)

DESCRIPTION

이 함수는 ㈜커미조아 환경 설정 파일(.CME2) 을 통해 설정된 값으로 모션제어기 및 디지털 입출력 장치의 각종 환경을 자동으로 설정해주는 함수입니다. CME2 파일은 COMI-AUTOMATION 설치 시에 함께 제공되는 소프트웨어 'CME Builder' 에서 구성이 가능합니다. ㈜커미조아의 'CME Builder' 는 GUI(Graphics User Interface) 환경에서 모션 제어를 위한 편리한 환경 설정을 지원합니다. 자세한 설명은 'Appendix 편' 을 참조해 주시기 바랍니다.

PARAMETER

▶ szCmeFile : CME2 파일의 절대 경로 혹은 파일의 상대 경로를 지정합니다.

RETURN VALUE

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

EXAMPLE

C/C++

```
void InitMotionDevices()
{
	////////////////////////////////////////////////////////////////////////////////
	// [CAUTION]: 아래와 다른 CME 파일을 사용하는 경우에는 아래 코드를 수정하여 올바른 CME 파일의
	// 경로를 지정하십시오.
	#define CME_FILE_NAME	"Default.cme2"
	char szCmeFilePath[MAX_PATH], szSystemDir[MAX_PATH];

	/* 윈도우즈 시스템 디렉토리명 얻어오기 (일반적으로는 C:\Windows\System32) */
	GetSystemDirectory(szSystemDir, MAX_PATH);
	sprintf(szCmeFilePath, "%s\\%s", szSystemDir, CME_FILE_NAME);

	if(cmmGnInitFromFile(szCmeFilePath) != cmERR_NONE){
		cmmErrShowLast(NULL); // 에러 발생하였으면 에러 디스플레이.
	}
	else{
		////////////////////////////////////////////////////////////////////////////
		// CME 파일에서 정의한 축수와 실제 장착되어 있는 모션 축수를 비교확인(確認) 하고자 할 때는
		// 아래와 같이 하면 됩니다.
		long nNumAxes_installed, nNumAxes_defined;
```

```
                cmmAdvGetNumAvailAxes(&nNumAxes_installed); // 현재 실제로 PC 에 장착되어 있는 모션축 수
(cmmGnDeviceLoad() 함수에서 반환하는 축수와 동일)
                cmmAdvGetNumDefinedAxes(&nNumAxes_defined); // CME 파일에서 정의한 축 수(CME 파일이
지정되지 않는 경우에는 현재 장착된 축 수와 일치합니다.)
                if(nNumAxes_installed != nNumAxes_defined){
                        MessageBox(NULL, "현재 장착되어 있는 모션 축 수가 정의된 축 수와 일치하지
않습니다", "Warning", MB_OK | MB_ICONWARNING);
                }
        }

        // SERVO-ON //
        long nNumAxes;
        cmmAdvGetNumDefinedAxes(&nNumAxes);
        for(int i=0; i<nNumAxes; i++){
                cmmGnSetServoOn(i, TRUE);
        }
}
```

Visual Basic

```
' ****************************************************************
' 윈도우 시스템 디렉토리명을 얻어오려면 비주얼 베이직에서
' 다음과 같은 코드를 CmmSDK.bas 모듈에 추가해 준다.
' Public Declare Function GetWindowsDirectory _
' Lib "kernel32.dll" Alias "GetWindowsDirectoryA" _
' (ByVal lpBuffer As String, ByVal nSize As Long) _
' As Long
' Public Declare Function GetSystemDirectory _
' Lib "kernel32.dll" Alias "GetSystemDirectoryA" _
' (ByVal lpBuffer As String, ByVal nSize As Long) _
' As Long
' ****************************************************************

' ************************************************************************************************
' [CAUTION]: 아래와 다른 CME 파일을 사용하는 경우에는 아래 코드를 수정하여 올바른 CME 파일의
' 경로를 지정하십시오.
' ************************************************************************************************
Const MAX_PATH = 100
Const CME_FILE_NAME = "Default.cme2"


Private Sub InitMotionDevices(void)

   Dim szCmeFilePath As String
   Dim szSystemDir As String
   Dim nNumAxes_installed As Long
   Dim nNumAxes_defined As Long
   Dim nNumAxes As Long
   Dim length As Long
   Dim i As Long

   length = GetSystemDirectory(szSystemDir, MAX_PATH)
   szSystemDir = Left(szSystemDir, MAX_PATH)
   szCmeFilePath = szSystemDir & CME_FILE_NAME

   If cmmGnInitFromFile(szCmeFilePath) <> cmERR_NONE Then
      Call cmmErrShowLast(Null)
   Else
      ' ************************************************************************************************
      ' CME 파일에서 정의한 축수와 실제 장착되어 있는 모션 축수를 비교확인 (確認)하고자 할 때는
      ' 아래와 같이 하면 됩니다.
      ' ************************************************************************************************
      Call cmmAdvGetNumAvailAxes(nNumAxes_installed)
      Call cmmAdvGetNumfinedAxes(nNumAxes_defined)
      If nNumAxes_installed <> nNumAxes_defined Then
         MsgBox ("현재 장착되어 있는 모션 축 수가 정의된 축 수와 일치하지 않습니다")
      End If
   End If

   ' SERVO-ON
```

```
    Call cmmAdvGetNumDefinedAxes(nNumAxes)

    For i = 0 To nNumAxes
        Call cmmGnSetServoOn(i, cmTRUE)
    Next

End Sub
```

Delphi

```
Const CME_FILE_NAME = 'Default.cme2';
Const MAX_PATH = 100;

procedure InitMotionDevices();
var
        szCmeFilePath : Array[0..MAX_PATH] of Char;
        szSystemDir : Array[0..MAX_PATH] of Char;
        nNumAxes_installed: LongInt;
        nNumAxes_defined: LongInt;
        nNumAxes : LongInt;
        i : LongInt;

begin
        //윈도우즈 시스템 디렉토리명 얻어오기 (일반적으로는 C:\Windows\System32)
        GetSystemDirectory(@szSystemDir, MAX_PATH);
        Format('%s\\%s',[ szSystemDir, CME_FILE_NAME]);

        if(cmmGnInitFromFile(@szCmeFilePath) <> cmERR_NONE) then
        begin
                cmmErrShowLast(0); // 에러 발생하였으면 에러 디스플레이.
        end

        else
        begin
                ////////////////////////////////////////////////////////////////////////////
                // CME 파일에서 정의한 축수와 실제 장착되어 있는 모션 축수를 비교확인(確認) 하고자 할 때는
                // 아래와 같이 하면 됩니다.
                cmmAdvGetNumAvailAxes(@nNumAxes_installed);
        // 현재 실제로 PC 에 장착되어 있는 모션축 수 (cmmGnDeviceLoad() 함수에서 반환하는 축수와 동일)
                cmmAdvGetNumDefinedAxes(@nNumAxes_defined);
        // CME 파일에서 정의한 축 수(CME 파일이 지정되지 않는 경우에는 현재 장착된 축 수와 일치합니다.)
                if(nNumAxes_installed <> nNumAxes_defined) then
                begin
                        Writeln('현재 장착되어 있는 모션 축 수가 정의된 축 수와 일치하지 않습니다');
                end;
        end;

        // SERVO-ON //
        cmmAdvGetNumDefinedAxes(@nNumAxes);
        //i := 0;
        For i := 0 to nNumAxes do cmmGnSetServoOn(i, cmTRUE);

end;
```

NAME

cmmGnInitFromFile_MapOnly
–CME 파일을 통한 환경(環境) 초기화

INFORMATION

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

SYNOPSIS

❒ VT_I4cmmGnInitFromFile_MapOnly ([in] VT_STR szCmeFile, [in] VT_I4 nMapType)

DESCRIPTION

이 함수는 ㈜커미조아 환경 설정 파일(.CME2) 을 통해 설정된 값으로 모션 제어기 및 디지털 입출력 장치의 각종 환경 중 모션 또는 범용 디지털 입출력 채널에 대한 맵핑 정보만을 자동으로 설정해 주는 함수입니다. CME2 파일은 COMI-AUTOMATION 설치 시에 함께 제공되는 소프트웨어 'CME Builder' 에서 구성이 가능합니다. ㈜커미조아의 'CME Builder' 는 GUI(Graphics User Interface) 환경에서 모션 제어를 위한 편리한 환경 설정을 지원합니다. 자세한 설명은 'Appendix 편' 을 참조해 주시기 바랍니다.

PARAMETER

▶szCmeFile : CME2 파일의 절대 경로 혹은 파일의 상대 경로를 지정합니다.

▶nMapType : 모션 혹은 범용 디지털 입출력 채널 맵핑 정보 중 하나 또는 두 가지 모두의 Mapping 형식을 선택합니다.

Value	Meaning
cmDMAP_MOTION 또는 0	모션 축에 대한 맵핑 정보
cmDMAP_DIO 또는 1	범용 디지털 입출력 채널에 대한 맵핑 정보
cmDMAP_ALL 또는 2	모션 및 범용 디지털 입출력 채널에 대한 맵핑 정보

SEE ALSO

cmmGnInitFromFile

RETURN VALUE

Value	Meaning
음수	수행 실패. 자세한 내용은 '에러처리' 편을 참고합니다
cmErr_NONE	수행 성공 t

REFERENCE

□ 이 함수는 cmmGnInitFromFile 처럼 모션 환경 설정에 대한 모든 정보를 가져오지 않고 모션 또는 I/O 채널 맵핑 정보에 대해서만 로드 하는 것에 유의 하십시오.

EXAMPLE

C/C++

```
void InitMotionDevices()
{
	////////////////////////////////////////////////////////////////////////////////
	// [CAUTION]: 아래와 다른 CME 파일을 사용하는 경우에는 아래 코드를 수정하여 올바른 CME 파일의
	// 경로를 지정하십시오.
	#define CME_FILE_NAME	"Default.cme2"
	char szCmeFilePath[MAX_PATH], szSystemDir[MAX_PATH];

/* 윈도우즈 시스템 디렉토리명 얻어오기 (일반적으로는 C:\Windows\System32) */
	GetSystemDirectory(szSystemDir, MAX_PATH);
	sprintf(szCmeFilePath, "%s\\%s", szSystemDir, CME_FILE_NAME);

	if(cmmGnInitFromFile_MapOnly(szCmeFilePath, cmDMAP_ALL) != cmERR_NONE){
		cmmErrShowLast(NULL); // 에러 발생하였으면 에러 디스플레이.
	}
	else{
		////////////////////////////////////////////////////////////////////////////
		// CME 파일에서 정의한 축수와 실제 장착되어 있는 모션 축수를 비교확인(確認) 하고자 할 때는
		// 아래와 같이 하면 됩니다.
		long nNumAxes_installed, nNumAxes_defined;
		cmmAdvGetNumAvailAxes(&nNumAxes_installed); // 현재 실제로 PC 에 장착되어 있는
		// 모션축 수 (cmmGnDeviceLoad() 함수에서 반환하는 축수와 동일)
		cmmAdvGetNumDefinedAxes(&nNumAxes_defined); // CME 파일에서 정의한 축 수
		//(CME 파일이 지정되지 않는 경우에는 현재 장착된 축 수와 일치합니다.)
		if(nNumAxes_installed != nNumAxes_defined){
			MessageBox(NULL, "현재 장착되어 있는 모션 축 수가 정의된 축 수와 일치하지
				않습니다", "Warning", MB_OK | MB_ICONWARNING);
		}
	}

	// SERVO-ON //
	long nNumAxes;
	cmmAdvGetNumDefinedAxes(&nNumAxes);
	for(int i=0; i<nNumAxes; i++){
		cmmGnSetServoOn(i, TRUE);
	}
}
```

Visual Basic

```
' ****************************************************************
' 윈도우 시스템 디렉토리명을 얻어오려면 비주얼 베이직에서
' 다음과 같은 코드를 CmmSDK.bas 모듈에 추가해 준다.
' Public Declare Function GetWindowsDirectory _
' Lib "kernel32.dll" Alias "GetWindowsDirectoryA" _
' (ByVal lpBuffer As String, ByVal nSize As Long) _
' As Long
' Public Declare Function GetSystemDirectory _
' Lib "kernel32.dll" Alias "GetSystemDirectoryA" _
' (ByVal lpBuffer As String, ByVal nSize As Long) _
' As Long
' ****************************************************************

' ************************************************************************************************
' [CAUTION]: 아래와 다른 CME 파일을 사용하는 경우에는 아래 코드를 수정하여 올바른 CME 파일의
' 경로를 지정하십시오.
' ************************************************************************************************
Const MAX_PATH = 100
Const CME_FILE_NAME = "Default.cme2"


Private Sub InitMotionDevices(void)

   Dim szCmeFilePath As String
   Dim szSystemDir As String
   Dim nNumAxes_installed As Long
   Dim nNumAxes_defined As Long
```

```
    Dim nNumAxes As Long
    Dim length As Long
    Dim i As Long

    length = GetSystemDirectory(szSystemDir, MAX_PATH)
    szSystemDir = Left(szSystemDir, MAX_PATH)
    szCmeFilePath = szSystemDir & CME_FILE_NAME

    If cmmGnInitFromFile_MapOnly (szCmeFilePath, cmDMAP_ALL) <> cmERR_NONE Then
        Call cmmErrShowLast(Null)
    Else
        ' ********************************************************************************************
        ' CME 파일에서 정의한 축수와 실제 장착되어 있는 모션 축수를 비교확인 (確認)하고자 할 때는
        ' 아래와 같이 하면 됩니다.
        ' ********************************************************************************************
        Call cmmAdvGetNumAvailAxes(nNumAxes_installed)
        Call cmmAdvGetNumfinedAxes(nNumAxes_defined)
        If nNumAxes_installed <> nNumAxes_defined Then
            MsgBox ("현재 장착되어 있는 모션 축 수가 정의된 축 수와 일치하지 않습니다")
        End If
    End If

    ' SERVO-ON

    Call cmmAdvGetNumDefinedAxes(nNumAxes)

    For i = 0 To nNumAxes
        Call cmmGnSetServoOn(i, cmTRUE)
    Next

End Sub
```

```
Delphi

Const CME_FILE_NAME = 'Default.cme2';
Const MAX_PATH = 100;

procedure InitMotionDevices();
var
        szCmeFilePath : Array[0..MAX_PATH] of Char;
        szSystemDir : Array[0..MAX_PATH] of Char;
        nNumAxes_installed: LongInt;
        nNumAxes_defined: LongInt;
        nNumAxes : LongInt;
        i : LongInt;

begin
        //윈도우즈 시스템 디렉토리명 얻어오기 (일반적으로는 C:\Windows\System32)
        GetSystemDirectory(@szSystemDir, MAX_PATH);
        Format('%s\\%s',[ szSystemDir, CME_FILE_NAME]);

        if(cmmGnInitFromFile_MapOnly(@szCmeFilePath, cmDMAP_ALL) <> cmERR_NONE) then
        begin
                cmmErrShowLast(0); // 에러 발생하였으면 에러 디스플레이.
        end

        else
        begin
                ////////////////////////////////////////////////////////////////////////////
                // CME 파일에서 정의한 축수와 실제 장착되어 있는 모션 축수를 비교확인(確認) 하고자 할 때는
                // 아래와 같이 하면 됩니다.
                cmmAdvGetNumAvailAxes(@nNumAxes_installed);
        // 현재 실제로 PC 에 장착되어 있는 모션축 수 (cmmGnDeviceLoad() 함수에서 반환하는 축수와 동일)
                cmmAdvGetNumDefinedAxes(@nNumAxes_defined);
        // CME 파일에서 정의한 축 수(CME 파일이 지정되지 않는 경우에는 현재 장착된 축 수와 일치합니다.)
                if(nNumAxes_installed <> nNumAxes_defined) then
                begin
                        Writeln('현재 장착되어 있는 모션 축 수가 정의된 축 수와 일치하지 않습니다');
                end;
        end;
```

```
        // SERVO-ON //
        cmmAdvGetNumDefinedAxes(@nNumAxes);
        //i := 0;
        For i := 0 to nNumAxes do cmmGnSetServoOn(i, cmTRUE);

end;
```

NAME

cmmGnSetServoOn
cmmGnGetServoOn
– 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 동작 시에 반드시 주의 환경을 점검하고, 안전 사항에 유의해야 합니다.

SYNOPSIS

❒ VT_I4 cmmGnSetServoOn ([in] VT_I4 Axis, [in] VT_I4 Enable)

❒ VT_I4 cmmGnGetServoOn ([in] VT_I4 Axis, [out] VT_PI4 Enable)

DESCRIPTION

cmmGnSetServoOn() 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.
cmmGnGetServoOn() 함수는 현재의 SERVO-ON 신호의 출력 상태를 반환합니다.

PARAMETER

▶ Axis: 축 번호(0 부터 시작합니다).

▶ Enable : cmmGnSetServoOn 함수의 인자이며, SERVO-ON 신호의 출력 상태를 설정합니다.

Value	Meaning
0 또는 cmFALSE	SERVO-OFF
1 또는 cmTRUE	SERVO-ON

▶ Enable : cmmGnGetServoOn 함수의 인자이며, SERVO-ON 신호의 출력 상태를 반환합니다.

Value	Meaning
0 또는 cmFALSE	SERVO-OFF
1 또는 cmTRUE	SERVO-ON

RETURN VALUE

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

REFERENCE

□ 서보를 On 시켜야 하는 경우 cmmGnDeivceLoad() 함수 호출 이후에 cmmGnSetServoOn(Axis#, cmTRUE)를 호출해 주셔야 서보 모터가 정상 동작 합니다.

□ cmmCfgSetMioProperty (0/*축번호*/, cmSVON_LOGIC, cmLOGIC_A) 또는 cmmCfgSetMioProperty (0/*축번호*/, cmSVON_LOGIC, cmLOGIC_B) 를 실행하여 서보온 신호의 출력로직을 설정할 수 있습니다.

EXAMPLE

C/C++

```
#include “Cmmsdk.h”
#include “CmmsdkDef.h”

void OnSetServoOn ()
{
	long nAxisNo = 1;			// Servo-On 신호를 제어할 축 선택.
	long nMioStates;			// 모션 상태 정보

	/* 해당 축의 Servo-On 상태를 확인한 후, 설정된 상태에 따라 ON/OFF 상태 제어 */

	if ( cmmStReadMioStatuses ( nAxisNo,  &nMioStates ) == cmERR_NONE )
	{
		if ( (nMioStates >> cmIOST_SVON) & 0x1 == cmFALSE )
		{
			// Servo OFF 상태이므로 Servo ON 상태로 설정합니다.
			cmmGnSetServoOn ( nAxisNo, cmTRUE );
		}
		else
		{
		// Servo ON 상태이므로 Servo OFF 상태로 설정합니다.
		cmmGnSetServoOn ( nAxisNo, cmFALSE );
		}
	}
}
```

Visual Basic

```
Private Sub OnSetServoOn()

	Dim nAxisNo As Long	' Servo-On 신호를 제어할 축 선택.
	Dim nMioStates As Long	' 모션 상태 정보
	Dim nResult As Long

	nAxisNo = 1

	' 해당 축의 Servo-On 상태를 확인한 후, 설정된 상태에 따라 ON/OFF 상태 제어

	If cmmStReadMioStatuses(nAxisNo, nMioStates) = cmERR_NONE Then

		Call cmmGnBitShift(nMioStates, cmIOST_SVON, nResult)

		If nResult = cmFALSE Then
			' Servo OFF 상태이므로 Servo ON 상태로 설정합니다.
			Call cmmGnSetServoOn(nAxisNo, cmTRUE)

		Else
			' Servo ON 상태이므로 Servo OFF 상태로 설정합니다.
			  Call cmmGnSetServoOn(nAxisNo, cmFALSE)
		End If
	End If

End Sub
```

Delphi

```
procedure OnSetServoOn ();
var
	nAxisNo : LongInt;			// Servo-On 신호를 제어할 축 선택.
	nMioStates : LongInt;			// 모션 상태 정보
```

```
begin
	nAxisNo := 1;

	// 해당 축의 Servo-On 상태를 확인한 후, 설정된 상태에 따라 ON/OFF 상태 제어

	if cmmStReadMioStatuses ( nAxisNo,  @nMioStates ) = cmERR_NONE then
	begin
		if  ( ( nMioStates shr cmIOST_SVON ) and $1 ) = cmFALSE then
		begin
			// Servo OFF 상태이므로 Servo ON 상태로 설정합니다.
			cmmGnSetServoOn ( nAxisNo, cmTRUE );
		end
		else
			// Servo ON 상태이므로 Servo OFF 상태로 설정합니다.
			cmmGnSetServoOn ( nAxisNo, cmFALSE );
		end;
	end;

end;
```

NAME

cmmGnPulseAlarmRes
– 알람 리셋(Alarm Reset) 펄스 출력(出力) 및 제어(制御)

INFORMATION

Etc General Function

VC++/VB

BCB/Delphi/.NET

Level 1

☺ 다소 주의

LX534 제품에서는 지원하지 않는 기능입니다.

SYNOPSIS

❒ VT_I4 cmmGnPulseAlarmRes

([in] VT_I4 Axis, [in] VT_I4 IsOnPulse, [in] VT_I4 dwDuration,[in] VT_I4 IsWaitPulseEnd)

DESCRIPTION

cmmGnPulseAlarmRes() 함수는 지정한 축의 알람 리셋(Reset) 펄스 출력을 제어합니다. 서보 드라이버를 사용하실 때는 외부에서 디지털 접점을 이용하여, 서보에서 발생한 알림 상황을 초기화 할 수 있도록 서보 드라이브에서 상에서 요구하는 입력 신호가 존재하는데, 이 신호 입력에 대응하기 위해, 모션 보드에서는 기본으로 제공되는 디지털 출력 신호를 통해 발생하게 되며, 이 신호를 알람 리셋 신호(Alarm Reset Signal)라 합니다. 이 함수를 통해 서보에 발생한 알람을 해제시킬 수 있습니다.

이 함수를 통해 알람 리셋을 사용하실 때 주의 하실 점은 다음과 같습니다.

1. 서보 드라이브에 발생한 모든 알람을 이 신호의 출력을 통해 해제할 수 없습니다. 서보드라이브에서 발생한 알람은 그 성격과 종류에 따라서 이와 같은 알람 리셋 신호(Alarm Reset Signal)로 해제될 수 있거나, 해제되지 않을 수 있습니다. 대표적으로 토크 과부하와 같은 알람 상황은 대부분 알람 리셋 신호로 해제가 되지만, 주 전원 부족이나 서보 상태 및 조건 등이 동작이 불가한 경우에 발생하는 심각한 경우에도 알람 상황 자체는 동일하게 발생할 수 있습니다. 중요한 것은 이 신호의 출력은 모션 보드에서 출력하는 순수한 단방향 디지털 출력 신호기 때문에 서보드라이브의 알람 상태를 제어하는 신호가 아닙니다.

2. ㈜ 커미조아의 LX50x 제품군에서는 2 축마다, DI(디지털 입력) 3 점, DO(디지털 출력) 3 점을 제공하는데, 여기서 제공하는 디지털 출력을 사용해 알람 리셋 신호를 출력합니다. 알람 리셋으로 디지털 출력을 사용하기 위해서는 하드웨어 매뉴얼을 참고하거나 다음과 같이 터미널 상의 J1 점퍼를 1-2 번에 위치시켜서 알람 리셋 신호로 디지털 출력 신호 라인을 활성화 하도록 해야 합니다.

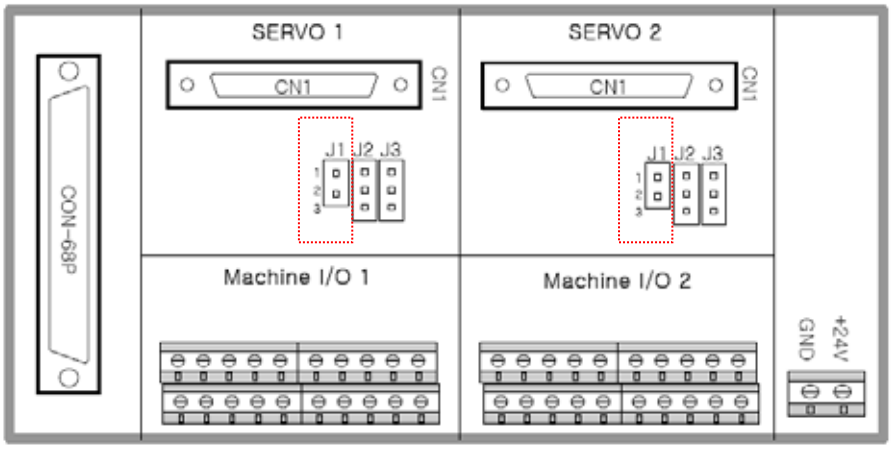


그림 6-1 커미조아 서보 전용 터미널

PARAMETER

▶ Axis: 축 번호. 축 번호는 상수값으로 [cmX1] 부터 0 번째 축을 기준 축으로 임의의 축을 설정할 수 있습니다.

▶ IsOnPulse : 펄스 출력상태를 설정 (設定) 합니다.

Value	Meaning
0 또는 cmFALSE	초기상태 : OFF 종료상태 : ON
1 또는 cmTRUE	초기상태 : ON 종료상태 : OFF

▶ dwDuration : 펄스의 출력 유지 시간을 지정합니다. 이 출력 시간은 밀리초(millisec) 로 단위로 설정 할 수 있습니다. 설정된 시간 동안 초기 펄스 출력 상태는 유지되다가, 설정 (設定) 된 시간 후에 종료상태가 됩니다.

▶ IsWaitPulseEnd : 펄스 출력이 종료상태가 될 때까지 함수 반환을 대기할 것인지에 대한 설정을 합니다.

RETURN VALUE

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

SEE ALSO

cmmGnSetAlarmRes, cmmGnGetAlarmRes

REFERENCE

□ 본 함수의 전달 인자 중 하나인 'dwDuration' 과 'IsOnPulse'인자에 대해서 주의하실 필요가 있습니다. dwDuration 인자는 서보에 알람 리셋 출력이 발생했을 때 Pulse 입력이 전달되는 시간을 결정합니다. 이 시간은 서보 드라이브 사양마다 다르며, 서보 드라이브 알람 리셋 'dwDuration' 인자의 시간이 너무 짧거나 길면, 알람 리셋이 되지 않거나 서보 드라이브에 무리를 줄 수도 있습니다.

□ 'IsOnPulse' 인자는 초기의 펄스 출력의 상태를 지정합니다. 해당 펄스 출력이 Rising Edge 형태인지, Falling Edge 방식인지를 결정하게 됩니다. 서보 드라이브의 알림 리셋 입력 로직을 확인(確認)하셔서 해당 입력 논리에 맞게 설정해주시기 바랍니다.

NAME

cmmGnSetAlarmRes
cmmGnGetAlarmRes
– 알람 리셋(Alarm Reset) 신호 출력(出力) 제어(制御)

INFORMATION

Etc General Function

VC++/VB

BCB/Delphi/.NET

Level 1

다소 주의

LX534 제품에서는 지원하지 않는 기능입니다.

SYNOPSIS

- VT_I4 cmmGnSetAlarmRes ([in] VT_I4 Axis, [in] VT_I4 IsOn)
- VT_I4 cmmGnGetAlarmRes ([in] VT_I4 Axis, [out] VT_PI4 IsOn)

DESCRIPTION

cmmGnSetAlarmRes() 함수는 지정한 축의 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.

본 함수가 ***cmmGnPulseAlarmRes()*** 와 다른 점은 본 함수를 통한 하드웨어적인 출력은 펄스 신호(Pulse Signal) 형태가 아니며, 지속적인 출력 혹은 그 반대의 지속적인 단절 상태를 명시적으로 제어하게 됩니다. 일반적인 상황에서 ***cmmGnSetAlarmRes()*** 함수보다는 ***cmmGnPulseAlarmRes()*** 함수를 사용하는 것이 편리할 수 있으며, 서보드라이브에 출력되는 알람 리셋 출력을 펄스 신호(Pulse Signal) 로 인가할 수 있습니다.

이 함수를 통해 알람 리셋을 사용하실 때 주의 하실 점은 다음과 같습니다.

1. 서보 드라이브에 발생한 모든 알람을 이 신호의 출력을 통해 해제할 수 없습니다. 서보드라이브에서 발생한 알람은 그 성격과 종류에 따라서 이와 같은 알람 리셋 신호(Alarm Reset Signal)로 해제될 수 있거나, 해제되지 않을 수 있습니다. 대표적으로 토크 과부하와 같은 알람 상황은 대부분 알람 리셋 신호로 해제가 되지만, 주 전원 부족이나 서보 상태 및 조건 등이 동작이 불가한 경우에 발생하는 심각한 경우에도 알람 상황 자체는 동일하게 발생할 수 있습니다. 중요한 것은 이 신호의 출력은 모션 보드에서 출력하는 순수한 단방향 디지털 출력 신호기 때문에 서보드라이브의 알람 상태를 제어하는 신호가 아닙니다.

2. ㈜ 커미조아의 LX50x 제품군에서는 2 축에서, DI(디지털 입력) 3 점, DO(디지털 출력) 3 점을 제공하는데, 여기서 제공하는 디지털 출력을 사용해 알람 리셋 신호를 출력합니다. 알람 리셋으로 디지털 출력을 사용하기 위해서는 하드웨어 매뉴얼을 참고하거나 다음과 같이 터미널의 상의 J1 점퍼를 1-2 번에 위치시켜서 알람 리셋 신호로 디지털 출력 신호 라인을 활성화 하도록 해야 합니다.

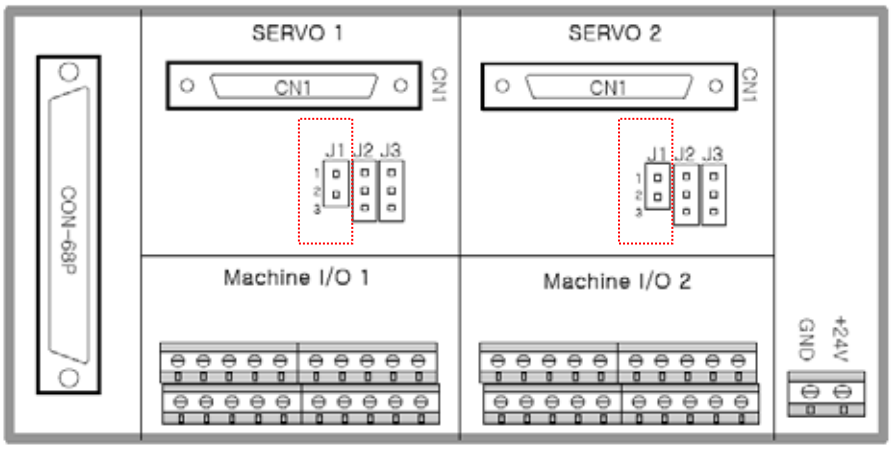


그림 6-2 커미조아 서보 전용 터미널

PARAMETER

▶ Axis: 축 번호. 축 번호는 상수값으로 [cmX1] 부터 0 번째 축을 기준 축으로 임의의 축을 설정할 수 있습니다.

▶ IsOn : cmmGnSetAlarmRes 함수의 인자이며, 출력 상태를 설정합니다. 출력 상태는 다음과 같이 설정할 수 있습니다.

Value	Meaning
0 (cmFALSE)	출력 상태를 비(非) 활성화 합니다.
1 (cmTRUE)	출력 상태를 활성화 합니다.

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

Value	Meaning
0 (cmFALSE)	출력 상태가 비(非)활성화 상태입니다.
1 (cmTRUE)	출력 상태가 활성화 상태입니다.

RETURN VALUE

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

SEE ALSO

cmmGnPulseAlarmRes

REFERENCE

□ 본 함수의 사용에 대해서 주의하실 필요가 있습니다. 알람 리셋 출력을 위해서 출력 상태를 너무 오랜 시간 지속하게 되거나 그 반대의 경우에 있어, 주요한 사항입니다. 알람 리셋 출력에 의한 반응 시간은 서보 드라이브 사양마다 다르며, 서보 드라이브 알람 출력 시간이 너무 짧거나 길면, 알람 리셋이 되지 않거나 서보 드라이브에 무리를 줄 수도 있습니다.

□ 초기의 펄스 출력의 상태를 지정하기 위해서는 SEE ALSO 에 명시된 cmmGnPulseAlarmRes 함수를 사용하십시오. **cmmGnPulseAlarmRes 해당 함수는 내정된 기능을 통해서, 본 함수의 알람 리셋 On/Off 의 동작과 시간을 바탕으로 함께 조합하여, 최종 결과물인 펄스 출력의 Alarm Reset 을 동작할 수 있으며, 해당 펄스 출력이 Rising Edge 형태인지, Falling Edge 방식인지를 결정**할 수 있습니다.

EXAMPLE

C/C++

```
#include “Cmmsdk.h”
#include “CmmsdkDef.h”

void OnAlarmReset ()
{
	long nAxisNo = 1;			// ARST 신호를 출력 할 축을 선택합니다.

	/* 일정 시간 동안 ARST 신호를 내보내고, 다시 클리어 시켜줍니다. */

	cmmGnSetAlarmRes( nAxisNo, cmTRUE );		// ARST ON

	// 서보 드라이버의 알람 리셋 출력에 의한 반응 시간을 확인하여 설정하여 주시기 바랍니다.
	Sleep(50);

	cmmGnSetAlarmRes( nAxisNo, cmFALSE ); // ARST OFF
}
```

Visual Basic

```
Private Sub OnAlarmReset ()

	Dim nAxisNo As Long		‘ ARST 신호를 출력 할 축을 선택합니다.
	nAxisNo = 1

	‘ 일정 시간 동안 ARST 신호를 내보내고, 다시 클리어 시켜줍니다.

	Call cmmGnSetAlarmRes( nAxisNo, cmTRUE)		‘ ARST ON

	‘ 서보 드라이버의 알람 리셋 출력에 의한 반응 시간을 확인하여 설정하여 주시기 바랍니다.
	Sleep(50)

	Call cmmGnSetAlarmRes( nAxisNo, cmFALSE )		‘ ARST OFF

End Sub
```

Delphi

```
procedure OnAlarmReset ();
var
	nAxisNo : LongInt;	// ARST 신호를 출력 할 축을 선택합니다.

begin
	nAxisNo := 1;

	// 일정 시간 동안 ARST 신호를 내보내고, 다시 클리어 시켜줍니다.
	cmmGnSetAlarmRes ( nAxisNo, cmTRUE);  // ARST ON

	// 서보 드라이버의 알람 리셋 출력에 의한 반응 시간을 확인하여 설정하여 주시기 바랍니다.
	Sleep(50);

	cmmGnSetAlarmRes ( nAxisNo, cmFALSE );// ARST OFF

end;
```

NAME

cmmGnSetSimulMode
cmmGnGetSimulMode
– 시뮬레이션 모드 (활성(活性), 비활성(非活性))

INFORMATION

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

SYNOPSIS

- ❒ VT_I4 cmmGnSetSimulMode ([in] VT_I4 Axis, [in] VT_I4 IsSimulMode)
- ❒ VT_I4 cmmGnGetSimulMode ([in] VT_I4 Axis, [out] VT_PI4 IsSimulMode)

DESCRIPTION

cmmGnSetSimulMode() 함수는 지정한 축의 "시뮬레이션모드"를 활성 또는 비활성화합니다. "시뮬레이션 모드" 기능은 모든 환경은 그대로 두고 COMMAND 펄스 출력만 내보내지 않는 모드입니다. 단, 이때에도 논리적으로는 COMMAND 펄스는 정상적으로 출력되는 것으로 됩니다. 따라서 COMMAND 기준으로 보았을 때 논리적으로는 정상적으로 동작합니다. 이 모드는 기구물을 구동하는 것이 위험하거나 불안할 때 기구물을 실제로 움직이지 않고 프로그램을 확인(確認)하거나 디버깅하기 위한 기능입니다.
cmmGnGetSimulMode() 함수는 현재 지정한 축의 "시뮬레이션모드"가 활성화 되어 있는지를 반환합니다.

PARAMETER

▶ Axis: 축 번호. 축 번호는 0 부터 시작합니다.

▶ IsSimulMode : cmmGnSetSimulMode 의 인자이며, "시뮬레이션모드" 활성/비활성 설정 값을 설정합니다.

Value	Meaning
0 (cmFALSE)	"시뮬레이션모드" 비활성화 합니다.
1 (cmTRUE)	"시뮬레이션모드" 활성화 합니다.

▶ IsSimulMode : cmmGnGetSimulMode 의 인자이며, "시뮬레이션모드" 활성/비활성 상태를 반환합니다.

Value	Meaning
0 (cmFALSE)	"시뮬레이션모드" 비활성화 상태입니다.
1 (cmTRUE)	"시뮬레이션모드" 활성화 상태입니다.

RETURN VALUE

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

NAME

cmmGnPutInternalSTA
– 소프트웨어적으로 STA 신호 발생(發生)

INFORMATION

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

SYNOPSIS

❒ VT_I4 cmmGnPutInternalSTA ([in] VT_I4 AxesMask)

DESCRIPTION

이 함수는 소프트웨어적으로 STA 신호를 발생하는 함수입니다. STA 신호는 하드웨어적으로 모션을 START 시키는 신호입니다. 기본적으로 모션은 소프트웨어 이송명령이 실행되면 그에 맞는 모션을 바로 시작합니다.

그러나, Machine I/O 환경 설정 함수인 cmmCfgSetMioProperty 함수를 통해서 STA 모드를 HARDWARE 모드로 설정하면 이송명령이 하달되어도 바로 이송을 시작하지 않고 STA 입력이 ON 이 되어야지만 이송을 시작합니다.

이와 같이 STA 신호는 하드웨어 신호에 동기되어서 출발하도록 하기 위해서 사용됩니다. 또한 여러 축을 동시에 출발시키기 위한 용도로도 사용됩니다. 본 함수는 이러한 용도로 사용되는 STA 입력을 소프트웨어적으로 Turn ON 상태로 만들어주는 함수입니다.

위에서 설명한 STA 모드를 HARDWARE 모드로 설정할 때에는 cmmCfgSetMioProperty(Axis, cmSTA_MODE, 1) 와 같이, 설정하면, 위에서 설명한 STA 입력에 동기하여 해당 모션 채널(혹은 다른 말로 모션 축)이 동작하게 됩니다.

PARAMETER

▶ AxesMask: 적용할 축 마스크값

RETURN VALUE

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

NAME

cmmGnSetEmergency
cmmGnGetEmergency
– 소프트웨어적인 비상상황(非常狀況) 제어

INFORMATION

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

SYNOPSIS

- ❒ VT_I4 cmmGnSetEmergency ([in] VT_I4 IsEnable, [in] VT_I4 IsDecStop)
- ❒ VT_I4 cmmGnGetEmergency ([out] VT_PI4 IsEnabled)

DESCRIPTION

cmmGnSetEmergency() 함수는 소프트웨어적으로 모션컨트롤러를 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.
cmmGnGetEmergency() 함수는 모션컨트롤러의 소프트웨어적인 Emergency 상태를 반환합니다.

PARAMETER

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

Value	Meaning
0 또는 cmFALSE	비상정지(停止) 비활성 (정상 상태)
1 또는 cmTRUE	비상정지(停止) 활성

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

Value	Meaning
0 또는 cmFALSE	비상정지(停止) 비활성 (정상 상태)
1 또는 cmTRUE	비상정지(停止) 활성

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

Value	Meaning
0 또는 cmFALSE	급정지(停止) (감속 없음)
1 또는 cmTRUE	감속 후 정지(停止) (감속도는 현재 각 축별로 설정된 감속도 적용)

RETURN VALUE

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

NAME

cmmGnBitShift
– 비트 연산(演算)을 수행

INFORMATION

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

SYNOPSIS

❒ VT_I4 cmmGnBitShift ([in] VT_I4 Value, [in] VT_I4 ShiftOption, [out] VT_PI4 Result)

DESCRIPTION

32 비트 값에 대해 오른쪽 또는 왼쪽 비트쉬프트(Bit Shift)를 수행합니다. 이 함수는 VB 에서 비트마스크가 용이하지 않아, VB 용으로 제공되는 함수입니다. 그러나 다른 개발 환경에서도 편리한 비트 쉬프트(Bit Shift) 기능을 위해 본 함수를 사용할 수 있습니다.

PARAMETER

▶ Value : 비트쉬프트를 수행할 대상 32 비트값

▶ ShiftOption : 몇 비트를 쉬프트할 것인지를 결정합니다. 이 값이 양수이면 왼쪽 쉬프트를 수행하고 음수이면 오른쪽쉬프트를 수행합니다.

▶ Result : 비트쉬프트 결과값.

RETURN VALUE

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

EXAMPLE

C/C++

```
long Value = 0xf;
long nResult = 0;

    // 4 비트 왼쪽쉬프트 => 결과값은 0xf0 가 된다 //
    Value = cmmGnBitShift(Value, 4, &nResult);

    // 4 비트 오른쪽쉬프트 => 결과값은 다시 0xf 가 된다 //
    Value = cmmGnBitShift(Value, -4, &nResult);
```

Visual Basic

```
Dim Value As Long

Dim nResult As Long

Value = &HF
```

```
' 4 비트 왼쪽쉬프트 => 결과값은 0xf0 가 된다
Value = cmmGnBitShift(Value, 4, nResult);

' 4 비트 오른쪽쉬프트 => 결과값은 다시 0xf 가 된다
Value = cmmGnBitShift(Value, -4, nResult);
```

Delphi

```
Value : LongInt;

nResult : LongInt;

Value := $f;

// 4 비트 왼쪽쉬프트 => 결과값은 0xf0 가 된다 //
Value := cmmGnBitShift(Value, 4 , @nResult);

// 4 비트 오른쪽쉬프트 => 결과값은 다시 0xf 가 된다 //
Value := cmmGnBitShift(Value, -4, @nResult);
```

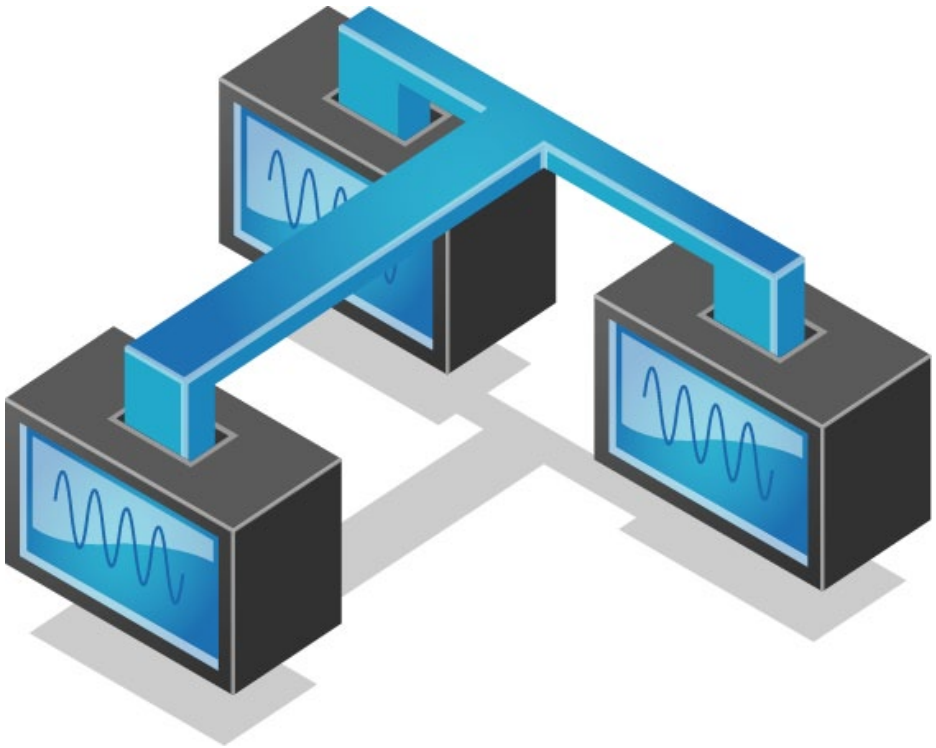
Chapter

7

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. 이 단원에서 단원에서 설명되는 모든 환경은 별도의 유틸리티 프로그램인 CME Builder 를 통하여 설정하여, 'Etc General 'Etc General Functions' 의 'cmmGnInitFromFile' 함수를 통해, 고객(顧客) 여러분들의 응용프로그램에 바로 응용프로그램에 바로 반영할 수 있습니다. 따라서 런타임시에 환경(環境)을 변경하고자 하는 경우가 경우가 아니면, 본 단원에서 설명되는 함수를 사용하실 필요가 없습니다. 그러나 필요에 따라 런타임시에 동적으로 환경(環境)을 변경해야 할 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 cmmCfgSetMioProperty ([in] VT_I4 Axis, [in] VT_I4 PropId, [in] VT_I4 PropVal) 모션 입출력 신호의 환경설정(環境設定)을 구성합니다.
❒ VT_I4 cmmCfgGetMioProperty ([in] VT_I4 Axis, [in] VT_I4 PropId, [out] VT_PI4 ProVal) 모션 입출력 신호의 환경설정(環境設定) 값을 반환(返還)합니다.
❒ VT_I4 cmmCfgSetFilter ([in] VT_I4 Axis, [in] VT_I4 IsEnable) 각종 I/O(Input/Output) 신호에 대해 노이즈 대응 필터(Filter) 기능을 설정(設定)합니다.
❒ VT_I4 cmmCfgGetFilter ([in] VT_I4 Axis, [out] VT_PI4 IsEnable) 각종 I/O(Input/Output) 신호에 대해 노이즈 대응 필터(Filter) 기능의 설정(設定) 상태를 반환(返還)합니다.
❒ VT_I4 cmmCfgSetInMode ([in] VT_I4 Axis, [in] VT_I4 InputMode, [in] VT_I4 IsInverse) 인코더의 펄스(Feedback Pulse) 신호의 입력 모드를 설정(設定)합니다.
❒ VT_I4 cmmCfgGetInMode ([in] VT_I4 Axis, [out] VT_PI4 InputMode, [out] VT_PI4 IsInverse) 인코더의 펄스(Feedback Pulse) 신호의 입력 모드 설정(設定) 상태를 반환합니다.
❒ VT_I4 cmmCfgSetOutMode ([in] VT_I4 Axis, [in] VT_I4 OutputMode) 지령 펄스(Command Pulse) 신호 출력 모드를 설정(設定)합니다.
❒ VT_I4 cmmCfgGetOutMode ([in] VT_I4 Axis, [out] VT_PI4 OutputMode) 지령 펄스(Command Pulse) 신호 출력 모드를 설정(設定) 상태를 반환합니다.
❒ VT_I4 cmmCfgSetInOutRatio ([in] VT_I4 Axis, [in] VT_R8 Ratio) 입력 펄스(Feedback Pulse)와 출력 펄스(Command Pulse)의 분해능(Resolution ratio)을 설정(設定)합니다.
❒ VT_I4 cmmCfgGetInOutRatio ([in] VT_I4 Axis, [out] VT_PR8 Ratio) 입력 펄스(Feedback Pulse)와 출력 펄스(Command Pulse)의 분해능(Resolution ratio)의 설정(設定) 상태를 반환합니다.
❒ VT_I4 cmmCfgSetUnitDist ([in] VT_I4 Axis, [in] VT_R8 UnitDist) 지정된 모션 축에 대한 논리적(論理的) 거리 단위를 설정(設定)합니다.
❒ VT_I4 cmmCfgGetUnitDist ([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 cmmCfgSetUnitSpeed ([in] VT_I4 Axis, [in] VT_R8 UnitSpeed) 지정된 모션 축에 대한 논리적(論理的) 속도 단위를 설정(設定)합니다.
❒ VT_I4 cmmCfgGetUnitSpeed ([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 cmmCfgSetSpeedRange ([in] VT_I4 Axis, [in] VT_R8 MaxPPS) 지정된 모션 축에 대한 모션 속도를 제한(制限) 하고, 제한범위(制限範圍)를 설정합니다.
❒ VT_I4 cmmCfgGetSpeedRange ([in] VT_I4 Axis, [out] VT_PR8 MinPPS, [out] VT_PR8 MaxPPS) 지정된 모션 축에 대한 모션 속도를 제한(制限) 하고, 제한범위(制限範圍)를 설정(設定) 상태를 반환합니다.
❒ VT_I4 cmmCfgSetSpeedPattern ([in] VT_I4 Axis, [in] VT_I4 SpeedMode, [in] VT_R8 WorkSpeed, [in] VT_R8 Accel, [in] VT_R8 Decel) 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 cmmCfgGetSpeedPattern ([in] VT_I4 Axis, [out] VT_PI4 SpeedMode, [out] VT_PR8 WorkSpeed, [out] VT_PR8 Accel, [out] VT_PR8 Decel) 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 cmmCfgSetActSpdCheck ([in] VT_I4 IsEnable, [in] VT_I4 Interval) 입력 펄스(Feedback Pulse) 를 통해 모션의 실제 속도가 산출(算出) 되며, 입력 펄스의 실제속도(實際速度) 검출을 설정합니다.

❒ VT_I4 cmmCfgGetActSpdCheck ([out] VT_PI4 IsEnable, [out] VT_PI4 Interval) 입력 펄스(Feedback Pulse) 를 통해 모션의 실제 속도가 산출(算出) 되며, 입력 펄스의 실제속도(實際速度) 검출의 설정을 반환합니다.
❒ VT_I4 cmmCfgSetSoftLimit ([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 cmmCfgGetSoftLimit ([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.
❒ VT_I4 cmmCfgSetRingCntr ([in] VT_I4 Axis, [in] VT_I4 TargCntr, [in] VT_I4 IsEnable, [in] VT_R8 CntMax) 지정된 모션 축에 대해 링카운터(Rign-Counter) 기능을 설정(設定)합니다.
❒ VT_I4 cmmCfgGetRingCntr ([in] VT_I4 Axis, [in] VT_I4 TargCntr, [out] VT_PI4 IsEnable, [out] VT_PR8 CntMax) 지정된 모션 축에 대해 링카운터 기능에 대한 해당 설정(設定)을 반환합니다.
❒ VT_I4 cmmCfgSetVelCorrRatio ([in] VT_I4 Axis, [in] VT_R8 CorrRatio) 모션의 작업속도 보정(補正) 시에 보정(補正)된 작업속도를 신출하는 비례(比例) 값을 설정합니다.
❒ VT_I4 cmmCfgGetVelCorrRatio ([in] VT_I4 Axis, [out] VT_PR8 CorrRatio) 모션의 작업속도 보정(補正) 시에 보정(補正)된 작업속도를 산출하는 비례(比例) 값을 반환합니다.
❒ VT_I4 cmmCfgSetFilterAB ([in] VT_I4 Channel, [in] VT_I4 Target, [in] VT_I4 IsEnable) EA/EB 또는 PA/PB 신호 입력 회로에 노이즈 필터를 적용할 지를 설정(設定)하는 함수입니다.
❒ VT_I4 cmmCfgGetFilterAB ([in] VT_I4 Channel, [in] VT_I4 Target, [out] VT_PI4 IsEnabled) EA/EB 또는 PA/PB 신호 입력 회로에 노이즈 필터를 적용할 지에 대한 설정(設定)을 반환합니다.
❒ VT_I4 cmmCfgSetCtrlMode ([in] VT_I4 Axis, [in] VT_I4 CtrlMode) 이송 목표 좌표의 기준을 커맨드 위치로 할지 피드백 위치로 할지를 설정(設定)하는 함수입니다.
❒ VT_I4 cmmCfgGetCtrlMode ([in] VT_I4 Axis, [out] VT_PI4 CtrlMode) 이송 목표 좌표의 기준을 커맨드 위치로 할지 피드백 위치로 할지에 대한 설정(設定)을 반환합니다.
❒ VT_I4 cmmCfgSetSeqMode ([in] VT_I4 SeqMode) 이전의 이송 작업이 완료되지 않은 축에 새로운 이송 명령이 하달되었을 때의 처리 정책을 설정(設定)하는 함수입니다.
❒ VT_I4 cmmCfgGetSeqMode ([out] VT_PI4 SeqMode) 이전의 이송 작업이 완료되지 않은 축에 새로운 이송 명령이 하달되었을 때의 처리 정책에 대한 설정(設定)을 반환하는 함수입니다.
❒ VT_I4 cmmCfgSetManExtLimit ([in] VT_I4 Axis, [in] VT_I4 IsSetELP, [in] VT_I4 IsEnable, [in] VT_I4 ManState) 수동(소프트웨어적으로) Limit 입력 기능을 사용할 것인지를 설정(設定)하는 함수입니다. 단, 이 함수는 COMI-LX504a 제품에서만 적용 가능한 함수입니다.
❒ VT_I4 cmmCfgGetManExtLimit ([in] VT_I4 Axis, [in] VT_I4 IsGetELP, [out] VT_PI4 IsEnable, [out] VT_PI4 ManState) 수동(소프트웨어적으로) Limit 입력 기능을 사용할 것인지에 대한 설정(設定)을 반환하는 함수입니다. 단, 이 함수는 COMI-LX504a 제품에서만 적용 가능한 함수입니다.

7.2 함수 설명

NAME

cmmCfgSetMioProperty
cmmCfgGetMioProperty
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 cmmCfgSetMioProperty ([in] VT_I4 Axis, [in] VT_I4 PropId, [in] VT_I4 PropVal)
- VT_I4 cmmCfgGetMioProperty ([in] VT_I4 Axis, [in] VT_I4 PropId, [out] VT_PI4 ProVal)

DESCRIPTION

cmmCfgSetMioProperty() 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.

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

PARAMETER

▶ Axis: 축 번호. 축번호는 상수값으로 [cmX1] 부터 0 번째 축을 기준 축으로 임의의 축을 설정할 수 있습니다.

▶ 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 (cmALM_LOGIC)	Alarm(ALM) 신호의 입력로직입니다. 설정 및 반환되는 PropVal 은 다음과 같습니다. ▪ 0 (cmLOGIC_A) : A 접점 방식 => 평상시 Open, 감지되면 Close 되는 스위치 방식 ▪ 1 (cmLOGIC_B) : B 접점 방식 => 평상시 Close, 감지되면 Open 되는 스위치
1 (cmALM_MODE)	Alarm 입력이 ON 되어 해당 축의 모션작업이 정지(停止)할 때 정지(停止)되는 방식의 설정 값입니다. 설정 및 반환되는 PropVal 은 다음과 같습니다. ▪ 0 : 즉시정지(停止) ▪ 1 : 감속 후 정지(停止)
2 (cmCMP_LOGIC)	위치비교출력(CMP) 신호의 출력방식 설정 값입니다. ▪ 0 : Active low => 평상시 HIGH 상태를 유지하다가 트리거 시점에서 LOW 로 떨어졌다가 다시 HIGH 로 올라갑니다. ▪ 1 : Active high => 평상시 LOW 상태를 유지하다가 트리거 시점에서 HIGH 로 올라갔다가 다시 LOW 로 떨어집니다.

PropId	Meaning & PropVal
3 (cmDR_LOGIC)	-/+ DR 신호의 입력로직 설정 값입니다. 설정 및 반환되는 PropVal 은 다음과 같습니다. ▪ 0 (cmLOGIC_A) : A 접점 방식 ▪ 1 (cmLOGIC_B) : B 접점 방식
4 (cmEL_LOGIC)	-EL 과 +EL 신호의 입력로직 설정 값입니다. 설정 및 반환되는 PropVal 은 다음과 같습니다. ▪ 0 (cmLOGIC_A) : A 접점 방식 ▪ 1 (cmLOGIC_B) : B 접점 방식
5 (cmEL_MODE)	-/+ EL 신호가 ON 되어 정지(停止)할 때 정지(停止) 방식의 설정 값입니다. 설정 및 반환되는 PropVal 은 다음과 같습니다. ▪ 0 : 즉시정지(停止) ▪ 1 : 감속 후 정지(停止)
6 (cmERC_LOGIC)	ERC 출력 신호의 출력로직의 설정 값입니다. 설정 및 반환되는 PropVal 은 다음과 같습니다. ▪ 0 (cmLOGIC_A) : A 접점 방식 ▪ 1 (cmLOGIC_B) : B 접점 방식
7 (cmERC_OUT)	원점복귀 완료 시에 ERC 출력여부의 설정 값입니다. 설정 및 반환되는 PropVal 은 다음과 같습니다. ▪ 0 (cmFALSE) : 원점복귀 완료 시에 ERC 출력 없음. ▪ 1 (cmTRUE) : 원점복귀 완료 시에 ERC 출력.
8 (cmEZ_LOGIC)	EZ(엔코더 Z 상) 입력 신호의 입력로직의 설정 값입니다. 설정 및 반환되는 PropVal 은 다음과 같습니다. ▪ 0 (cmLOGIC_A) : A 접점 방식 ▪ 1 (cmLOGIC_B) : B 접점 방식
9 (cmINP_EN)	INP 신호 입력 활성화의 설정 값입니다. 설정 및 반환되는 PropVal 은 다음과 같습니다. ▪ 0 (cmFALSE) : INP 비활성 ▪ 1 (cmTRUE) : INP 활성 => Command 출력이 완료되더라도 INP 신호가 ON 되기 전까지는 작업이 완료되지 않은 것으로 간주.
10 (cmINP_LOGIC)	INP(Inposition) 신호의 입력로직 설정 값입니다. 설정 및 반환되는 PropVal 은 다음과 같습니다. ▪ 0 (cmLOGIC_A) : A 접점 방식 ▪ 1 (cmLOGIC_B) : B 접점 방식
11 (cmLTC_LOGIC)	LTC(Latch) 신호의 입력로직 설정 값입니다. 설정 및 반환되는 PropVal 은 다음과 같습니다. ▪ 0 (cmLOGIC_A) : A 접점 방식 ▪ 1 (cmLOGIC_B) : B 접점 방식
12 (cmLTC_LTC2SRC)	두 번째 LATCH COUNTER 의 대상카운터 설정 값입니다. 설정 및 반환되는 PropVal 은 다음과 같습니다. ▪ 0 : Deviation counter value ▪ 1 : Preset speed of command pulse
13 (cmORG_LOGIC)	ORG(원점센서) 신호의 입력로직 설정 값입니다. 설정 및 반환값의 PropVal 은 다음과 같습니다. ▪ 0 (cmLOGIC_A) : A 접점 방식 ▪ 1 (cmLOGIC_B) : B 접점 방식
14 (cmSD_EN)	SD(Start of Deceleration) 신호의 입력상태를 설정합니다. 설정 및 반환되는 PropVal 은 다음과 같습니다. ▪ 0 (cmFALSE) : SD 입력을 비활성화합니다. ▪ 1 (cmTRUE) : SD 입력을 활성화합니다. 활성화되었을 때 SD 신호에 따른 동작방식은 cmSD_LATCH 와 cmSD_MODE 설정값에 의해 결정됩니다.
15 (cmSD_LOGIC)	SD(Start of Deceleration) 신호의 입력로직 설정 값입니다. 설정 및 반환되는 PropVal 은 다음과 같습니다. ▪ 0 (cmLOGIC_A) : A 접점 방식 ▪ 1 (cmLOGIC_B) : B 접점 방식

PropId	Meaning & PropVal
16 (cmSD_LATCH)	SD(Start of Deceleration) 신호를 래치(Latch)할 것인지에 대한 속성값입니다. 설정 및 반환되는 PropVal 은 다음과 같습니다. • 0 (cmFALSE) : SD 가 ON 되어 감속 중이거나 초기속도로 운전 중일 때 SD 신호가 다시 OFF 상태로 변경되면 작업속도까지 다시 가속됩니다. • 1 (cmTRUE) : SD 가 ON 상태에서 OFF 상태로 바뀌어도 작업속도로 가속하지 않습니다.
17 (cmSD_MODE)	SD 신호에 따른 동작모드 설정 값입니다. 설정 및 반환되는 PropVal 은 다음과 같습니다. • 0 : SD 신호가 ON 되면 초기속도까지 감속합니다(정지(停止)하지 않음). • 1 : SD 신호가 ON 되면 감속 후 정지(停止)합니다.
18 (cmSTA_MODE)	Start mode 의 설정 값입니다. 설정 및 반환되는 PropVal 은 다음과 같습니다. • 0 : STA 입력신호는 무시되며, 이동명령이 내려지면 바로 이동을 시작합니다. • 1 : 이동명령이 내려지면 바로 시작하지 않고, STA 신호가 ON 이 되면 이동을 시작합니다.
19 (cmSTA_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 입력신호가 HIGH 상태에서 LOW 상태로 천이될 때 ON
20 (cmSTP_MODE)	STP 신호모드의 설정 값입니다. 설정 및 반환되는 PropVal 은 다음과 같습니다. • 0 : Ignore STP => STP 입력신호 무시 • 1 : Immediate stop => STP 입력이 ON 되면 즉시정지(停止) • 2 : Stop after decel => STP 입력이 ON 되면 감속 후 정지(停止)
21 (cmCLR_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
22 (cmCLR_SIGTYPE)	CLR 신호의 신호 형태의 설정 값입니다. 설정 및 반환되는 PropVal 은 다음과 같습니다. • 0 : Falling edge => 스위치가 Open 상태에서 Close 되는 순간에 카운터가 클리어됩니다. • 1 : Rising edge => 스위치가 Close 상태에서 Open 되는 순간에 카운터가 클리어됩니다. • 2 : Low level => 스위치가 Close 되어 있는 동안에는 계속 카운터가 클리어됩니다. • 3 : High level => 스위치가 Open 되어 있는 동안에는 계속 카운터가 클리어됩니다.
23 (cmCMP_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 펄스폭과 동일한 펄스폭을 가짐 • 양수의 값 : 이 값에 1.5us 가 곱해진 값이 펄스폭이 됩니다. 즉, 이 값을 1 로 하면 1.5us, 2 로 하면 3us…와 같이 됩니다.
24 (cmERC_ONTIME)	ERC 출력펄스의 펄스폭의 설정 값입니다. 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
25 (cmSVON_LOGIC)	서보온(Servo-On) 신호의 출력 로직의 설정 값입니다. ▪ 0 (cmLOGIC_A) : A 접점 방식 ▪ 1 (cmLOGIC_B) : B 접점 방식

RETURN VALUE

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

EXAMPLE

C/C++

```
#include "cmmSDK.h"
#include "cmmSDKDef.h"

void OnSetMioProperty_Set ()
{
	long nAxisNo = 1;			// MIO 신호 환경을 설정할 축을 선택합니다.
	long nAlmLogic, nAlmMode;	// ALM 신호 환경 설정 정보.

	/* ALM 신호에 대해 설정되어 있는 로직 및 모드를 확인하여,
	'ALM Logic : B 접점, ALM Mode : 즉시 정지' 로 설정합니다. */


	// ALM 로직 설정 상태 확인 및 설정
	if (cmmCfgGetMioProperty ( nAxisNo, cmALM_LOGIC, &nAlmLogic ) == cmERR_NONE )
	{
		if ( nAlmLogic != cmLOGIC_B)
		{
			// ALM 로직을 'B 접점 방식'으로 설정
			cmmCfgSetMioProperty ( nAxisNo, cmALM_LOGIC, cmLOGIC_B );
		}
	}

	// ALM 신호 ON 시 정지 모드 설정
	if (cmmCfgGetMioProperty ( nAxisNo, cmALM_MODE, &nAlmMode ) == cmERR_NONE )
	{
		if ( nAlmMode != cmFALSE )
		{
			// ALM 신호 ON 시 정지 모드를 '즉시 정지' 모드로 설정
			cmmCfgSetMioProperty ( nAxisNo, cmALM_MODE, cmFALSE );
		}
	}
}
```

Visual Basic

```
Private Sub OnSetMioProperty_Set ()

	Dim nAxisNo As Long                                  ' MIO 신호 환경을 설정할 축을 선택합니다.
	Dim nAlmLogic As Long, nAlmMode As Long              ' ALM 신호 환경 설정 정보.

	nAxisNo = 1

	'ALM 신호에 대해 설정되어 있는 로직 및 모드를 확인하여,
	'ALM Logic : B 접점, ALM Mode : 즉시 정지' 로 설정합니다.


	' ALM 로직 설정 상태 확인 및 설정
	If cmmCfgGetMioProperty ( nAxisNo, cmALM_LOGIC, nAlmLogic ) = cmERR_NONE Then
		If nAlmLogic <> cmLOGIC_B Then
			'ALM 로직을 'B 접점 방식'으로 설정
			Call cmmCfgSetMioProperty ( nAxisNo, cmALM_LOGIC, cmLOGIC_B )
		End If
	End If

	' ALM 신호 ON 시 정지 모드 설정
```

```
    If cmmCfgGetMioProperty ( nAxisNo, cmALM_MODE, nAlmMode ) = cmERR_NONE Then
        If nAlmMode <> cmFALSE Then
            ' ALM 신호 ON 시 정지 모드를 '즉시 정지' 모드로 설정
            Call cmmCfgSetMioProperty ( nAxisNo, cmALM_MODE, cmFALSE )
        End If
    End If

End Sub
```

Delphi

```
procedure OnSetMioProperty_Set ();
var
    nAxisNo : LongInt;                  // MIO 신호 환경을 설정할 축을 선택합니다.
    nAlmLogic, nAlmMode : LongInt;      // ALM 신호 환경 설정 정보.

begin
    nAxisNo := 1;

    { ALM 신호에 대해 설정되어 있는 로직 및 모드를 확인하여,
    ALM Logic : B 접점, ALM Mode : 즉시 정지' 로 설정합니다. }


    // ALM 로직 설정 상태 확인 및 설정
    if cmmCfgGetMioProperty ( nAxisNo, cmALM_LOGIC, @nAlmLogic ) = cmERR_NONE then
    begin
        if nAlmLogic <> cmLOGIC_B then
        begin
            // ALM 로직을 'B 접점 방식'으로 설정
            cmmCfgSetMioProperty ( nAxisNo, cmALM_LOGIC, cmLOGIC_B );
        end;
    end;

    // ALM 신호 ON 시 정지 모드 설정
    if cmmCfgGetMioProperty ( nAxisNo, cmALM_MODE, @nAlmMode ) = cmERR_NONE then
    begin
        if nAlmMode <> cmFALSE then
        begin
            // ALM 신호 ON 시 정지 모드를 '즉시 정지' 모드로 설정
            cmmCfgSetMioProperty ( nAxisNo, cmALM_MODE, cmFALSE );
        end;
    end;

end;
```

NAME

cmmCfgSetFilter / cmmCfgGetFilter – I/O 신호 노이즈(Noise) 필터 설정(設定) 및 반환 (返還)

INFORMATION

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

SYNOPSIS

- ❒ VT_I4 cmmCfgSetFilter ([in] VT_I4 Axis, [in] VT_I4 IsEnable)
- ❒ VT_I4 cmmCfgGetFilter ([in] VT_I4 Axis, [out] VT_PI4 IsEnable)

DESCRIPTION

이 함수는 각종 I/O 신호에 필터 로직을 적용할지를 설정하는 함수입니다. 필터 로직이 적용되면 펄스폭이 너무 짧은 입력 신호는 무시되게 됩니다. 필터 로직이 적용되는 I/O 신호 및 펄스폭은 아래의 참고 항목을 참조하십시오.

PARAMETER

▶ Axis: 축번호. 축번호는 상수값으로 [cmX1] 부터 0 번째 축을 기준 축으로 임의의 축을 설정할 수 있습니다.

▶ IsEnable : cmmCfgSetFilter 함수의 인자이며, 필터로직을 적용 여부를 설정합니다.

Value	Meaning
0	Filter Disable (필터 로직 비활성화)
1	Filter Enable (필터 로직 활성화)

▶ IsEnable : cmmCfgGetFilter 함수의 인자이며, 필터로직의 적용 여부를 반환합니다.

Value	Meaning
0	Filter Disable (필터 로직 비활성화)
1	Filter Enable (필터 로직 활성화)

SEE ALSO

□ 필터 로직 적용이 Enable 됐을 때 필터 로직이 적용되는 신호 및 기준 펄스 폭은 다음과 같습니다.

필터 적용 I/O	필터 기준
+EL, -EL, SD, ORG, ALM, INP	펄스폭이 4μs 미만은 무시
+DR, -DR	펄스폭이 3.2ms 미만은 무시

RETURN VALUE

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

EXAMPLE

C/C++

```
#include “Cmmsdk.h”
#include “CmmsdkDef.h”

void OnSetNoiseFilter ()
{
	long nAxisNo = 1;			// Noise Filter 기능을 적용할 대상 축 선택.
	long nFilterEnable;			// Noise Filter 기능 활성 상태 정보.

	/* 노이즈 필터 로직 활성 여부를 확인하여 비활성 상태이면 활성 상태로 설정합니다. */
	if (cmmCfgGetFilter ( nAxisNo, &nFilterEnable ) == cmERR_NONE )
	{
		if ( nFilterEnable != cmTRUE)
		{
			// Noise Filter Enable
			cmmCfgSetFilter ( nAxisNo, cmTRUE);
		}
	}
}
```

Visual Basic

```
Private Sub OnSetNoiseFilter ()

	Dim nAxisNo As Long			‘ Noise Filter 기능을 적용할 대상 축 선택.
	Dim nFilterEnable As Long		‘ Noise Filter 기능 활성 상태 정보.

	nAxisNo = 1

	‘ 노이즈 필터 로직 활성 여부를 확인하여 비활성 상태이면 활성 상태로 설정합니다.
	If cmmCfgGetFilter ( nAxisNo, nFilterEnable ) = cmERR_NONE Then

		If nFilterEnable <> cmTRUE Then
			‘ Noise Filter Enable
			Call cmmCfgSetFilter ( nAxisNo, cmTRUE)
		End If
	End If

End Sub
```

Delphi

```
procedure OnSetNoiseFilter ();
var
	nAxisNo : LongInt;			// Noise Filter 기능을 적용할 대상 축 선택.
	nFilterEnable : LongInt;			// Noise Filter 기능 활성 상태 정보.

begin
	nAxisNo := 1;

	// 노이즈 필터 로직 활성 여부를 확인하여 비활성 상태이면 활성 상태로 설정합니다.
	if cmmCfgGetFilter ( nAxisNo, @nFilterEnable ) = cmERR_NONE then
	begin
		if nFilterEnable <> cmTRUE then
		begin
			// Noise Filter Enable
			cmmCfgSetFilter ( nAxisNo, cmTRUE );
		end;
	end;

end;
```

NAME

cmmCfgSetInMode
cmmCfgGetInMode
– 인코더 펄스 신호 입력(入力) 모드 설정(設定) 및 반환 (返還)

INFORMATION

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

SYNOPSIS

❒ VT_I4 cmmCfgSetInMode ([in] VT_I4 Axis, [in] VT_I4 InputMode, [in] VT_I4 IsInverse)

❒ VT_I4 cmmCfgGetInMode ([in] VT_I4 Axis, [out] VT_PI4 InputMode, [out] VT_PI4 IsInverse)

DESCRIPTION

cmmCfgSetInMode() 함수는 Feedback Pulse 의 입력 모드를 설정합니다. Feedback 펄스는 실제 모터 또는 구조물의 구동거리를 확인(確認)하기 위하여 사용되며 거의 대부분 인코더 입력을 사용합니다. 사용자는 4 가지 형태의 Feedback 펄스의 입력모드를 설정할 수 있습니다. 또한 이 함수는 입력 펄스의 로직(Logic)을 설정합니다. cmmCfgGetInMode() 함수는 현재 설정된 Feedback 펄스의 입력모드를 반환합니다.

PARAMETER

▶ Axis: 축번호. 축번호는 상수값으로 [cmX1] 부터 0 번째 축을 기준 축으로 임의의 축을 설정할 수 있습니다.

▶ InputMode : cmmCfgSetInMode 함수의 인자이며, Feedback 펄스의 입력 모드 설정값입니다. 입력 모드는 다음과 같이 5 가지 설정값을 가집니다.

Value	Meaning
0 (cmIMODE_AB1X)	1X A/B (1 채배 엔코더 입력 모드)
1 (cmIMODE_AB2X)	2X A/B (2 채배 엔코더 입력 모드)
2 (cmIMODE_AB4X)	4X A/B (4 채배 엔코더 입력 모드)
3 (cmIMODE_CWCCW)	CW/CCW (A 펄스 - 카운트 증가, B 펄스 - 카운트 감소)
4 (cmIMODE_STEP)	이 모드에서는 Feedback 위치값을 읽으면 Command 위치값이 바이패스(bypass)됩니다. 엔코더 피드백이 없는 경우(스텝모터)에 이 모드를 선택합니다.

▶ InputMode : cmmCfgGetInMode 함수의 인자이며, Feedback 펄스 입력 모드의 반환값입니다.

Value	Meaning
0 (cmIMODE_AB1X)	1X A/B (1 채배 엔코더 입력 모드)
1 (cmIMODE_AB2X)	2X A/B (2 채배 엔코더 입력 모드)
2 (cmIMODE_AB4X)	4X A/B (4 채배 엔코더 입력 모드)
3 (cmIMODE_CWCCW)	CW/CCW (A 펄스 - 카운트 증가, B 펄스 - 카운트 감소)
4 (cmIMODE_STEP)	스텝모터를 위한 Command 위치값 바이패스(bypass) 모드입니다.

▶ IsInverse : cmmCfgSetInMode 함수의 인자이며, Feedback Count 값의 UP/DOWN 방향을 반대로 할 것인지에 대한 설정 값입니다.

Value	Meaning
0	Feedback count 의 UP/DOWN 방향을 바꾸지 않습니다.
1	Feedback count 의 UP/DOWN 방향을 반대로 합니다.

▶ IsInverse : cmmCfgGetInMode 함수의 인자이며, Feedback Count 값의 UP/DOWN 방향을 반대로 할 것인지에 대한 반환값입니다.

Value	Meaning
0	Feedback count 의 UP/DOWN 방향이 반대가 아닙니다.
1	Feedback count 의 UP/DOWN 방향이 반대입니다.

RETURN VALUE

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

EXAMPLE

C/C++

```
#include "Cmmsdk.h"
#include "CmmsdkDef.h"

void OnSetInMode ()
{
	long nAxisNo = 1;			// Feedback Pulse 입력 모드를 설정할 축을 선택합니다.
	long nInputMode, nIsReverse;	// 입력 모드 정보.

	/* 설정되어있는 Feedback Pulse 입력 모드를 확인하여,
	입력 모드를 '4 채배 엔코더 입력 모드' 로 설정합니다. */

	// 입력 모드 및 입력 펄스 로직 설정 상태 확인
	if (cmmCfgGetInMode ( nAxisNo, &nInputMode, &nIsReverse ) == cmERR_NONE )
	{
		if ( nInputMode != cmIMODE_AB4X)
		{
			cmmCfgSetInMode ( nAxisNo, cmIMODE_AB4X, cmFALSE );
		}
	}
}
```

Visual Basic

```
Private Sub OnSetInMode ()

	Dim nAxisNo As Long				' Feedback Pulse 입력 모드를 설정할 축을 선택합니다.
	Dim nInputMode As Long, nIsReverse As Long	' 입력 모드 정보.

	nAxisNo = 1

	' 설정되어있는 Feedback Pulse 입력 모드를 확인하여,
	' 입력 모드를 '4 채배 엔코더 입력 모드' 로 설정합니다.

	' 입력 모드 및 입력 펄스 로직 설정 상태 확인
	If cmmCfgGetInMode ( nAxisNo, nInputMode, nIsReverse ) = cmERR_NONE Then

		If nInputMode <> cmIMODE_AB4X Then
			Call cmmCfgSetInMode ( nAxisNo, cmIMODE_AB4X, cmFALSE )
		End If
	End If

End Sub
```

Delphi

```
procedure OnSetInMode ();
var
        nAxisNo : LongInt;                      // Feedback Pulse 입력 모드를 설정할 축을 선택합니다.
        nInputMode, nIsReverse : LongInt;       // 입력 모드 정보.

begin
         nAxisNo := 1;

        { 설정되어있는 Feedback Pulse 입력 모드를 확인하여,
        입력 모드를 '4 채배 엔코더 입력 모드' 로 설정합니다. }

        // 입력 모드 및 입력 펄스 로직 설정 상태 확인
        if cmmCfgGetInMode ( nAxisNo, @nInputMode, @nIsReverse ) = cmERR_NONE then
        begin
                if nInputMode <> cmIMODE_AB4X then
                begin
                        cmmCfgSetInMode ( nAxisNo, cmIMODE_AB4X, cmFALSE );
                end;
        end;

end;
```

NAME

cmmCfgSetOutMode
cmmCfgGetOutMode
지령 펄스 신호 출력(出力) 모드 설정(設定) 및 반환 (返還)

INFORMATION

Environment
Configuration Functions
VC++/VB
BCB/Delphi/.NET
Level 2
다소 주의
부분적인 기능은 LX504a 제품에서만 지원됩니다. PARAMETER 항목을 자세히 참조합니다.

SYNOPSIS

- ❒ VT_I4 cmmCfgSetOutMode ([in] VT_I4 Axis, [in] VT_I4 OutputMode)
- ❒ VT_I4 cmmCfgGetOutMode ([in] VT_I4 Axis, [out] VT_PI4 OutputMode)

DESCRIPTION

cmmCfgSetOutMode() 함수는 Command 펄스의 출력 모드를 설정합니다. Command 신호에 대한 하드웨어적인 회로는 별도로 제공되는 "User's Guide" 매뉴얼의 "Part II Chapter2 의 2.2 절 모션컨트롤러 인터페이스 신호"단원을 참조하시기 바랍니다.

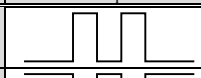
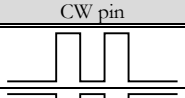
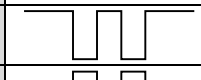
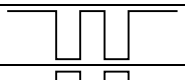
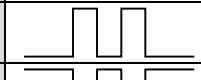
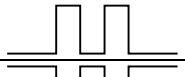
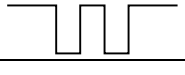
설정 매개변수의 반환의 용도로 제공되는 ***cmmCfgGetOutMode()*** 함수는 현재설정된 Command 펄스 출력모드를 반환합니다.

PARAMETER

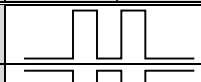
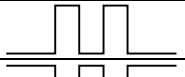
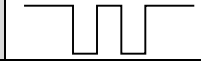
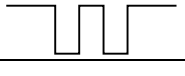
▶ Axis: 축번호. 축번호는 상수값으로 [cmX1] 부터 0 번째 축을 기준 축으로 임의의 축을 설정할 수 있습니다.

▶ OutputMode : Command 펄스 출력 모드의 설정값입니다. 출력 모드는 다음과 같이 10 가지 모드(Mode)를 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)

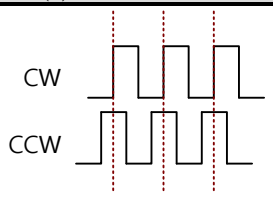
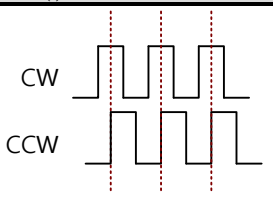
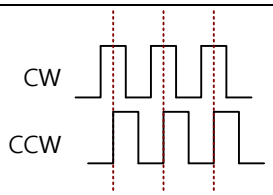
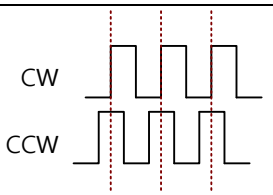
나) CW/CCW Mode

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

다) LX 504a 제품에서는 별도로 아래 두 가지 모드가 존재합니다. 아래 두 가지 기능은 기존 모션 보드 제품에서 지원하는 4 번과 5 번의 CW & CCW 커맨드 출력 모드를 4 번의 반대(Inverse)로 6 번이 되며, 5 번의 반대(Inverse) 로 7 번 모드로 사용할 수 있도록 지원합니다. 이 모드를 통해 모터의 회전 방향을 정방향에서 역방향으로 또는 그 반대로 상호 용이하게 변경할 수 있습니다.

Value	출력 형태			
	(+) 방향 운전 시		(-) 방향 운전 시	
	CW pin	CCW pin	CW pin	CCW pin
6	(Low)			(Low)
7	(High)			(High)

라) A/B Phase Mode (본 모드는 모든 커미조아 모션 제품에서 지원합니다)

Value	출력 형태	
	(+) 방향 운전 시	(-) 방향 운전 시
8		
9		

RETURN VALUE

□ cmmCfgSetOutMode() 및 cmmCfgGetOutMode() 함수의 반환값

Value	Meaning
음수	수행 실패. 자세한 내용은 '에러처리' 편을 참고합니다
cmERR_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, 스탭 드라이브에서는 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, 저희 모션 보드와 동일하게 설정하여 주십시오. 동일하게 설정이 되지 않았을 경우에는, 다음과 같은 현상이 발생할 수 있습니다.

1. 서보드라이브를 통해 동작하는 모터의 방향이 한방향으로만 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) 를 입력을 무시하게 됩니다. 이 현상을 확인하시려면, 서보드라이브의 Command Pulse Counter 와 저희 커미조아의 모션 보드의 Command Pulse Counter 와 의 연속적인 방향 전환 이동 상황에서 Pulse Counter 차이를 모니터링(Monitoring)함으로써, 현상을 확인할 수 있습니다.

이외에도 스탭 드라이브의 경우에는 드라이브 입력 부 회로의 사양에 따라서, Open Collecter 연결을 주로 사용하는데, 결선 방법과 상태에 따라서 일반적인 모터의 움직임에 이상 상황이 발생할 수 있습니다. 이때에는 해당 서보드라이브의 입력 부 사양을 첨부하여, 저희 ㈜ 커미조아의 고객(顧客) 지원 팀으로 문의해주시면 친절하게 상담하여 드리겠습니다.

EXAMPLE

C/C++

```
#include “Cmmsdk.h”
#include “CmmsdkDef.h”

void OnSetOutMode ()
{
	long nAxisNo = 1;			// Command Pulse 출력 모드를 설정할 축을 선택합니다.
	long nOutputMode;			// Comand Pulse 출력 모드 정보.

	/* 설정되어있는 Command Pulse 출력 모드를 확인하여, 4 번 모드로 설정합니다. */

	if (cmmCfgGetOutMode ( nAxisNo, &nOutputMode ) == cmERR_NONE )
	{
		if ( nOutputMode != 4 )
		{
			cmmCfgSetOutMode ( nAxisNo, 4 );
		}
	}
}
```

Visual Basic

```
Private Sub OnSetOutMode ()

	Dim nAxisNo As Long			‘ Command Pulse 출력 모드를 설정할 축을 선택합니다.
	Dim nOutputMode As Long			‘ Comand Pulse 출력 모드 정보.

	nAxisNo = 1

	‘ 설정되어있는 Command Pulse 출력 모드를 확인하여, 4 번 모드로 설정합니다.

	If cmmCfgGetOutMode ( nAxisNo, nOutputMode ) = cmERR_NONE Then

		If nOutputMode <> 4 Then
			Call cmmCfgSetOutMode ( nAxisNo, 4 )
		End If
	End If

End Sub
```

Delphi

```
procedure OnSetOutMode ();
var
	nAxisNo : LongInt;			// Command Pulse 출력 모드를 설정할 축을 선택합니다.
	nOutputMode : LongInt;			// Comand Pulse 출력 모드 정보.

begin
	nAxisNo := 1;

	// 설정되어있는 Command Pulse 출력 모드를 확인하여, 4 번 모드로 설정합니다.

	if cmmCfgGetOutMode ( nAxisNo, @nOutputMode ) = cmERR_NONE then
	begin
		if nOutputMode <> 4  then
		begin
			cmmCfgSetOutMode ( nAxisNo, 4 );
		end;
	end;

end;
```

NAME

cmmCfgSetInOutRatio
cmmCfgGetInOutRatio
– 입력(入力) 펄스와 출력(出力) 펄스의 분해능(分解能) 설정(設定) 및 반환(返還)

INFORMATION

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

SYNOPSIS

❒ VT_I4 cmmCfgSetInOutRatio

([in] VT_I4 Axis, [in] VT_R8 Ratio)

❒ VT_I4 cmmCfgGetInOutRatio

([in] VT_I4 Axis, [out] VT_PR8 Ratio)

DESCRIPTION

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.

PARAMETER

▶ Axis: 축번호. 축번호는 상수값으로 [cmX1] 부터 0 번째 축을 기준 축으로 임의의 축을 설정할 수 있습니다.

▶ Ratio : cmmCfgSetInOutRatio 함수의 인자이며, Feedback 펄스와 Command 펄스의 분해능 비를 지정합니다. 이 값은 아래와 같이 설정합니다.

Ratio = (Feedback 펄스 분해능)/(Command 펄스의 분해능)

▶ Ratio : cmmCfgGetInOutRatio 함수의 인자이며, Feedback 펄스와 Command 펄스의 분해능 비를 반환합니다.

RETURN VALUE

□ cmmCfgSetInOutRatio() 및 cmmCfgGetInOutRatio() 함수의 반환값

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

SEE ALSO

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

□ In/Out Ratio 는 cmmStGetPosition() 함수와 cmmStGetSpeed() 함수에서 카운터를 cmCNT_FEED 으로 설정한 경우에만 영향을 미칩니다.

EXAMPLE

C/C++

```
#include “Cmmsdk.h”
#include “CmmsdkDef.h”

void OnSetInOutRatio ()
{
	long nAxisNo = 0;			// 입력 펄스와 출력 펄스의 분해능 비율을 설정할 축을 선택합니다.
	double fInOutRatio;			// 분해능 비율 정보.

	/* 입력 펄스와 출력 펄스의 분해능 비율(In/Out Ratio)을 설정합니다. In/Out Ratio 설정은
	cmmCfgGetInOutRatio, cmmCfgSetInOutRatio 함수의 대상 카운터가 입력 펄스일 경우에만 영향을 미칩니다. */


	// 분해능 비율을 얻어와 분해능 비율을 ‘1’로 설정합니다.
	if (cmmCfgGetInOutRatio ( nAxisNo, &fInOutRatio ) != cmERR_NONE )
	{
		if ( fInOutRatio != 1.0f )
		{
			cmmCfgSetInOutRatio ( nAxisNo, 1.0f );
		}
	}
}
```

Visual Basic

```
Private Sub OnSetInOutRatio ()

	Dim nAxisNo As Long			‘ 입력 펄스와 출력 펄스의 분해능 비율을 설정할 축을 선택합니다.
	Dim fInOutRatio As Double		‘ 분해능 비율 정보.

	nAxisNo = 0

	‘ 입력 펄스와 출력 펄스의 분해능 비율(In/Out Ratio)을 설정합니다. In/Out Ratio 설정은
	‘cmmCfgGetInOutRatio, cmmCfgSetInOutRatio 함수의 대상 카운터가 입력 펄스일 경우에만 영향을 미칩니다.


	‘ 분해능 비율을 얻어와 분해능 비율을 ‘1’로 설정합니다.
	If cmmCfgGetInOutRatio ( nAxisNo, fInOutRatio ) <> cmERR_NONE Then

		If fInOutRatio <> 1 Then
			Call cmmCfgSetInOutRatio ( nAxisNo, 1 )
		End If
	End If

End Sub
```

Visual Delphi

```
procedure OnSetInOutRatio ();
var
	nAxisNo : LongInt;			// 입력 펄스와 출력 펄스의 분해능 비율을 설정할 축을 선택합니다.
	fInOutRatio : Double;			// 분해능 비율 정보.

begin
	nAxisNo := 0;

	{ 입력 펄스와 출력 펄스의 분해능 비율(In/Out Ratio)을 설정합니다. In/Out Ratio 설정은
	cmmCfgGetInOutRatio, cmmCfgSetInOutRatio 함수의 대상 카운터가 입력 펄스일 경우에만 영향을 미칩니다. }


	// 분해능 비율을 얻어와 분해능 비율을 ‘1’로 설정합니다.
	if cmmCfgGetInOutRatio ( nAxisNo, @fInOutRatio ) <> cmERR_NONE then
	begin
		if fInOutRatio <> 1 then
		begin
```

```
                    cmmCfgSetInOutRatio ( nAxisNo, 1 );
                end;
            end;

        end;
```

NAME

cmmCfgSetUnitDist
cmmCfgGetUnitDist
– 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 cmmCfgSetUnitDist

([in] VT_I4 Axis, [in] VT_R8 UnitDist)

❒ VT_I4 cmmCfgGetUnitDist

([in] VT_I4 Axis, [out] VT_PR8 UnitDist)

DESCRIPTION

cmmCfgSetUnitDist() 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.

PARAMETER

▶ Axis: 축번호. 축번호는 상수값으로 [cmX1] 부터 0 번째 축을 기준 축으로 임의의 축을 설정할 수 있습니다.

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

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

RETURN VALUE

□ cmmCfgSetUnitDist() 및 cmmCfgGetUnitDist() 함수의 반환값

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

SEE ALSO

□. 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 논리적거리와 논리적속도를 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.

$$D_u = P_r / L_r$$

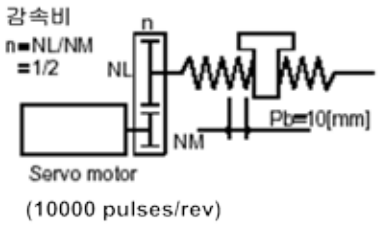
Pr : 모터 1 회전에 필요한 펄스 수 (모터의 Command 분해능)
Lr : 모터 1 회전 시에 이동되는 기구물의 거리

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 논리적거리 단위를 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

Unit distance (Du) = Pr/Lr = 10000/5 = 2000
Unit speed (Vu) = 2000

//따라서, Unit distance"와 "Unit speed"를 2000 으로 설정하면 다음과 같이 모든 이동함수에서 거리의 단위를 //mm 단위에 해당하는 값을 입력할 수 있습니다.

```
// 속도=100(mm/s), 가/감속도=1000(mm/s^2) 으로 설정한다. //
cmmCfgSetSpeedPattern(cmX1, cmSMODE_T, 100, 1000, 1000);

// (+)방향으로 20mm 이송한다 (실제는 20*2000 = 40000 펄스가 출력됨) //
cmmSxMove (0, 20);

// (-)방향으로 20mm 이송한다 (실제는 20*2000 = 40000 펄스가 출력됨) //
cmmSxMove (0, -20);
```

EXAMPLE

C/C++

```
#include "Cmmsdk.h"
#include "CmmsdkDef.h"

long nAxisNo = 1;              // Unit Distance 를 설정할 축을 선택합니다.

void OnSetUnitDist ()
{
    /*
    모터 1 회전 시 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

    따라서, 'Unit Distance' 와 'Unit Speed' 를 2000 으로 설정하면 모든 이송 함수에서 mm 거리단위에 대응하는
    값을 입력할 수 있습니다.
    */

    cmmCfgSetUnitDist ( nAxisNo, 2000 );      // Unit Distance 를 2000 으로 설정
    cmmCfgSetUnitSpeed ( nAxisNo, 2000 );     // Unit Speed 를 2000 으로 설정
```

```
}

void OnMove ()
{
	// 속도 = 100(mm/s), 가/감속도 = 1000(mm/s^2)으로 설정합니다.
	cmmCfgSetSpeedPattern ( nAxisNo, cmSMODE_T, 100, 1000, 1000 );

	// (+)방향으로 20mm 이송합니다. (실제는 20 * 2000 = 40000 펄스가 출력됩니다.)
	cmmSxMove ( nAxis, 20 );
}
```

Visual Basic

```
Dim nAxisNo As Long		' Unit Distance 를 설정할 축을 선택합니다.
nAxisNo = 1

Private Sub OnSetUnitDist ()

	' 모터 1 회전 시 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

	' 따라서, 'Unit Distance' 와 'Unit Speed' 를 2000 으로 설정하면 모든 이송 함수에서 mm 거리단위에 대응하는
	' 값을 입력할 수 있습니다.

	Call cmmCfgSetUnitDist ( nAxisNo, 2000 )		' Unit Distance 를 2000 으로 설정
	Call cmmCfgSetUnitSpeed ( nAxisNo, 2000 )	' Unit Speed 를 2000 으로 설정

End Sub

Private Sub OnMove ()

	' 속도 = 100(mm/s), 가/감속도 = 1000(mm/s^2)으로 설정합니다.
	Call cmmCfgSetSpeedPattern ( nAxisNo, cmSMODE_T, 100, 1000, 1000 )

	' (+)방향으로 20mm 이송합니다. (실제는 20 * 2000 = 40000 펄스가 출력됩니다.)
	Call cmmSxMove ( nAxis, 20 )

End Sub
```

Delphi

```
procedure OnSetUnitDist ();
begin
	{ 모터 1 회전 시 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

	따라서, 'Unit Distance' 와 'Unit Speed' 를 2000 으로 설정하면 모든 이송 함수에서 mm 거리단위에 대응하는
	값을 입력할 수 있습니다. }

	cmmCfgSetUnitDist (cmX1, 2000 );		// Unit Distance 를 2000 으로 설정
	cmmCfgSetUnitSpeed (cmX1, 2000 );		// Unit Speed 를 2000 으로 설정

end;

procedure OnMove ();
begin
	// 속도 = 100(mm/s), 가/감속도 = 1000(mm/s^2)으로 설정합니다.
	cmmCfgSetSpeedPattern ( cmX1, cmSMODE_T, 100, 1000, 1000 );

	// (+)방향으로 20mm 이송합니다. (실제는 20 * 2000 = 40000 펄스가 출력됩니다.)
	cmmSxMove ( cmX1, 20,0 );

end;
```

NAME

cmmCfgSetUnitSpeed
cmmCfgGetUnitSpeed
- 논리적 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 cmmCfgSetUnitSpeed

([in] VT_I4 Axis, [in] VT_R8 UnitSpeed)

❒ VT_I4 cmmCfgGetUnitSpeed

([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.

PARAMETER

▶ Axis: 축번호. 축번호는 상수값으로 [cmX1] 부터 0 번째 축을 기준 축으로 임의의 축을 설정할 수 있습니다.

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

▶ UnitSpeed : cmmCfgGetUnitSpeed 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. cmmCfgSetUnitSpeed() 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 fUnitDist 값을 3600/60, 즉 60 PPS 로 설정합니다(여기서 60 으로 나누는 것은 RPM 은 분당 회전수이므로 초당 3600/60 펄스를 출력해야 1 분에 3600 펄스가 나가기 때문입니다).
Ex 2) 1cm 이송에 필요한 펄스 수가 1000 펄스인 경우에 이동량의 단위를 cm/sec 로 하고자 한다면 fUnitDist 값을 1000 PPS 로 설정합니다.

EXAMPLE 1

C/C++

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

```
#include “Cmmsdk.h”
#include “CmmsdkDef.h”

void StartProgram()
{

        long nNumAxes = 0;
```

```
        if ( cmmLoadDll() != 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. */

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

        if ( cmmGnDeviceLoad(cmTRUE, &nNumAxes) != cmERR_NONE ){

        // MS VC++ 에서는 아래와 같이 에러 원인을 화면에 표시할 수 있습니다.
        cmmErrShowLast(Handle); //사용자 작성 윈도우의 핸들값입니다.

        // Borland C++ 계열에서는 아래와 같이 에러 원인을 화면에 표시할 수
        // 있습니다.
        //cmmErrShowLast(Form1->Handle);
        }
}

// 실제 이동 동작을 수행하는 가상 함수입니다.
void OnMove(void)
{
        // Set 10 pulses for unit distance
        // 이 예제에서는 1 회전에 필요한 펄스 수를 3600 펄스로
        // 가정하고 단위 거리를 1°로 설정한 것입니다.
        cmmCfgSetUnitDist(cmX1, 10);

        // Set 3600/60(=60) PPS for unit speed
        // 이 예제에서는 1 회전에 필요한 펄스 수를 3600 펄스로
        // 가정하고 단위 속도를 1rpm 으로 설정한 것입니다.
        cmmCfgSetUnitSpeed(cmX1, 3600./60);

        // Set trapezoidal speed mode //
        // Set speed as 100 rpm //
        //가속도와 감속도를 각각 200rpm/s 로 설정합니다. 이렇게 하면 작업속도가  100rpm 이므로 가속 및 감속
        시간은  각각 0.5 초가  걸립니다.
        cmmCfgSetSpeedPattern(cmX1,  cmSMODE_T, 100,200,200);

        // 모터를 720°회전한다. 실제로는 720*10 펄스가 출력됩니다. //
        cmmSxMove(cmX1, 720,cmFALSE);
}

void EndProgram(void)
{
        cmmGnDeviceUnload();
        cmmUnloadDll();
}
```

Visual Basic

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

        Dim nTotalAxis As Long
        Dim IRetVal As Long
        Dim IsLoaded As Long

         '=====================================================================
         ' cmmGnDeviceLoad 함수를 장치를 초기화합니다.
          IRetVal = cmmGnDeviceLoad(True, nTotalAxis)

          If IRetVal <> cmERR_NONE Then
             MsgBox ("cmmGnDeviceLoad has been failed")
          End If
```

```
        '=====================================================================

        '=====================================================================
        ' cmmGnDeviceIsLoaded 함수로 장치가 올바르게 불려졌는지를 확인할 수 있습니다.
         Call cmmGnDeviceIsLoaded(IsLoaded)

         If IsLoaded <> cmTRUE Then
           MsgBox ("디바이스 로드를 실패하였습니다")
           End
         End If
        '=====================================================================
 End Sub


'// 실제 이동 동작을 수행하는 가상 함수입니다.
Private Sub OnMove(void)

        '// Set 10 pulses for unit distance
        '// 이 예제에서는 1 회전에 필요한 펄스 수를 3600 펄스로
        '// 가정하고 단위 거리를 1°로 설정한 것입니다.
        Call cmmCfgSetUnitDist(cmX1, 10)

        '// Set 3600/60(=60) PPS for unit speed
        '// 이 예제에서는 1 회전에 필요한 펄스 수를 3600 펄스로
        '// 가정하고 단위 속도를 1rpm 으로 설정한 것입니다.
        Call cmmCfgSetUnitSpeed(cmX1, 3600# / 60)

        '// Set trapezoidal speed mode
        '// Set speed as 100 rpm
        '//가속도와 감속도를 각각 200rpm/s 로 설정합니다. 이렇게 하면 작업속도가
        ‘//100rpm 이므로 가속 및 감속 시간은  각각 0.5 초  걸립니다.
        Call cmmCfgSetSpeedPattern(cmX1, cmSMODE_T, 100, 200, 200)

        '// 모터를 720°회전한다. 실제로는 720*10 펄스가 출력됩니다.
        Call cmmSxMove(cmX1, 720, cmFALSE)
End Sub

Private Sub EndProgram()

        Call cmmGnDeviceUnload
        Call cmmUnloadDll

End Sub
```

```
Delphi

procedure OnSetUnitSpeed ();
begin
        { 1 회전에 필요한 펄스수가 3600 펄스일 때
        거리의 단위를 각도 (1°)로, 속도의 단위를 RPM 으로 설정합니다. }

        // 1 회전에 3600 펄스가 필요하므로 1° 회전을 하기 위해서는 10 펄스가 필요합니다.
        cmmCfgSetUnitDist (cmX1, 10 );

        // 단위 속도를 1rpm 으로 설정합니다. 3600(pulse) / 60(sec) = 60(pps)
        cmmCfgSetUnitSpeed (cmX1, 60 );

end;


procedure OnMove ();
begin
        { 가속도와 감속도를 각각 200rpm/s 로 설정합니다. 이렇게 하면 작업 속도가 100rpm 이므로
        가속 및 감속에는 0.5sec 가 소요됩니다. }
        cmmCfgSetSpeedPattern (cmX1, cmSMODE_T, 100, 200, 200 );

        // 모터를 720° 회전 시킵니다. 실제로는 720 * 10 pulse 가 출력됩니다.
        cmmSxMoveStart (cmX1, 720 );
end;
```

EXAMPLE 2

C/C++

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

```
#include “Cmmsdk.h”
#include “CmmsdkDef.h”

void StartProgram()
{
        long nNumAxes = 0;

        if ( cmmLoadDll() != 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. */

        OutputDebugString(“DLL 로드에 실패하였습니다”);
        // 이후 적절한 에러처리를 해주시기 바랍니다.

        }

        if ( cmmGnDeviceLoad(cmTRUE, &nNumAxes) != cmERR_NONE ){

        // MS VC++ 에서는 아래와 같이 에러 원인을 화면에 표시할 수 있습니다.
        cmmErrShowLast(Handle); //사용자가 제작 윈도우의 핸들입니다.

        // Borland C++ 계열에서는 아래와 같이 에러 원인을 화면에 표시할 수
        // 있습니다.
        // cmmErrShowLast(Form1->Handle);
        }
} /* void StartProgram() */

// 실제 이동 동작을 수행하는 가상 함수입니다.
void OnMove(void)
{

        // Set 1000 pulses for unit distance
        // 이 예제에서는 1cm 이동에 필요한 펄스 수를 1000 펄스로
        // 가정하고 단위 거리를 1cm 로 설정한 것입니다.
        cmmCfgSetUnitDist(cmX1, 1000);
        // Set 1000 PPS for unit speed
        // 이 예제에서는 1cm 이동에 필요한 펄스 수를 1000 펄스로
        // 펄스로 가정하고 단위 속도를 1rpm 로 설정한 것입니다.
        cmmCfgSetUnitSpeed(cmX1, 1000);

        // Set speed as 50 cm/sec/
        // Set constant speed mode

        cmmCfgSetSpeedPattern(cmX1, cmSMODE_C, 50,0,0);
        cmmSxMove(cmX1, 10); // 50 cm/sec 의 속도로 10cm 이동 . 실제로는 10*1000=10000 펄스가 출력됩니다.
}

void EndProgram(void)
{
        cmmGnDeviceUnload();
        cmmUnloadDll();
}
```

Visual Basic

```
'1cm 이동하는데 필요한 펄스수가 1000 펄스일 때 거리의 단위를 cm 로,
'속도의 단위를 cm/sec 로 설정하는 예제입니다.
```

```
Private Sub StartProgram()

        Dim nTotalAxis As Long
        Dim IRetVal As Long
        Dim IsLoaded As Long

        '=====================================================================
         ' cmmGnDeviceLoad 함수를 장치를 초기화합니다.
         IRetVal = cmmGnDeviceLoad(True, nTotalAxis)

         If IRetVal <> cmERR_NONE Then
            MsgBox ("cmmGnDeviceLoad has been failed")
         End If
        '=====================================================================

        '=====================================================================
        ' cmmGnDeviceIsLoaded 함수로 장치가 올바르게 불려졌는지를 확인할 수 있습니다.
         Call cmmGnDeviceIsLoaded(IsLoaded)

         If IsLoaded <> cmTRUE Then
           MsgBox ("디바이스 로드를 실패하였습니다")
           End
         End If
        '=====================================================================
  End Sub

'// 실제 이동 동작을 수행하는 가상 함수입니다.
Private Sub OnMove()

        '// Set 1000 pulses for unit distance
        '// 이 예제에서는 1cm 이동에 필요한 펄스 수를 1000 펄스로
        '// 가정하고 단위 거리를 1cm 로 설정한 것입니다.
        Call cmmCfgSetUnitDist(cmX1, 1000)

        '// Set 1000 PPS for unit speed
        '// 이 예제에서는 1cm 이동에 필요한 펄스 수를 1000 펄스로
        '// 펄스로 가정하고 단위 속도를 1rpm 로 설정한 것입니다.
        Call cmmCfgSetUnitSpeed(cmX1, 1000)

        '// Set speed as 50 cm/sec/ //
        '// Set constant speed mode //
        Call cmmCfgSetSpeedPattern(cmX1, cmSMODE_C, 50, 0, 0)

        Call cmmSxMove(cmX1, 10)
                 '50 cm/sec 의 속도로 10cm 이동
                '실제로는 10*1000=10000 펄스가 출력됩니다.

End Sub

Private Sub EndProgram()

        Call cmmGnDeviceUnload
        Call cmmUnloadDll

End Sub
```

NAME

cmmCfgSetSpeedRange
cmmCfgGetSpeedRange
– 모션 속도 제한 설정(設定) 및 반환(返還)

INFORMATION

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

SYNOPSIS

❒ VT_I4 cmmCfgSetSpeedRange

([in] VT_I4 Axis, [in] VT_R8 MaxPPS)

❒ VT_I4 cmmCfgGetSpeedRange

([in] VT_I4 Axis, [out] VT_PR8 MinPPS, [out] VT_PR8 MaxPPS)

DESCRIPTION

cmmCfgSetSpeedRange () 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 설정가능하며 기본적으로 설정되는 주파수 범위는 10Hz ~ 655,350Hz 입니다. 최저속도는 최대속도 설정에 따라서 자동으로 결정됩니다.
cmmCfgGetSpeedRange () 함수는 현재 설정되어 있는 속도의 최소 범위와 최대 범위를 반환합니다.

PARAMETER

▶ Axis: 축번호. 축번호는 상수값으로 [cmX1] 부터 0 번째 축을 기준 축으로 임의의 축을 설정할 수 있습니다.

▶ MaxPPS : cmmCfgSetSpeedRange 함수의 인자이며, 모션의 최고 속도를 PPS 로 설정합니다.

MaxPPS 값을 설정할 때는 Unit speed 설정에 관계없이 항상 **PPS** 단위로 설정하여야 합니다.

▶ MaxPPS : cmmCfgGetSpeedRange 함수의 인자이며, 모션의 최고 속도를 PPS 로 반환합니다.

▶ MinPPS : 모션의 최저 속도를 PPS 단위로 반환합니다. 사용자는 모션의 최저 속도를 참조만 할 수 있으며 사용자가 직접 설정할 수는 없습니다. 왜냐하면 최저속도는 최고 속도를 설정함에 따라서 자동으로 설정되기 때문입니다. 최저 속도와 최고 속도의 다음과 같습니다.

MinPPS = MaxPPS / 65,535
예를 들어서 MaxPPS 가 655,350 이면 MinPPS = 655,350 / 65,535 = 10 (PPS)이 됩니다.

EXAMPLE

C/C++

```
#include “Cmmsdk.h”
#include “CmmsdkDef.h”

void OnSetSpeedRange ()
{
        long nAxisNo = 1;                       // 모션 속도 제한 범위를 설정할 축을 선택합니다.
```

```
	double fMaxPPS, fMinPPS;		// 최저/최고 모션 속도 정보

	/* 모션 최고 속도 정보를 확인 후, 모션 최고 속도를 6.5MPPS 로 설정합니다.
	모션 최고 속도를 6.5MPPS 로 설정하면 모션 속도의 범위는 100 ~ 6553500 PPS 이 됩니다. */

	if(cmmCfgGetSpeedRange ( nAxisNo, &fMinPPS, &fMaxPPS ) == cmERR_NONE )
	{
		if ( fMaxPPS < 6553500 )
		{
			// 모션 최대 속도를 6553500 으로 설정합니다.
			cmmCfgSetSpeedRange ( nAxisNo, 6553500 );
		}
	}
}
```

Visual Basic

```
Private Sub OnSetSpeedRange ()

	Dim nAxisNo As Long				' 모션 속도 제한 범위를 설정할 축을 선택합니다.
	Dim fMaxPPS As Double, fMinPPS As Double		' 최저/최고 모션 속도 정보

	nAxisNo = 1

	' 모션 최고 속도 정보를 확인 후, 모션 최고 속도를 6.5MPPS 로 설정합니다.
	' 모션 최고 속도를 6.5MPPS 로 설정하면 모션 속도의 범위는 100 ~ 6553500 PPS 이 됩니다.
	If cmmCfgGetSpeedRange ( nAxisNo, fMinPPS, fMaxPPS ) = cmERR_NONE Then

		If fMaxPPS < 6553500 Then
			' 모션 최대 속도를 6553500 으로 설정합니다.
			Call cmmCfgSetSpeedRange ( nAxisNo, 6553500 )
		End If
	End If

End Sub
```

Delphi

```
procedure OnSetSpeedRange ();
var
	nAxisNo : LongInt;		// 모션 속도 제한 범위를 설정할 축을 선택합니다.
	fMaxPPS, fMinPPS : Double;	// 최저/최고 모션 속도 정보

begin
	nAxisNo := 1;

	{ 모션 최고 속도 정보를 확인 후, 모션 최고 속도를 6.5MPPS 로 설정합니다.
	모션 최고 속도를 6.5MPPS 로 설정하면 모션 속도의 범위는 100 ~ 6553500 PPS 이 됩니다. }
	if cmmCfgGetSpeedRange ( nAxisNo, @fMinPPS, @fMaxPPS ) = cmERR_NONE then
	begin
		if fMaxPPS < 6553500 then
		begin
			// 모션 최대 속도를 6553500 으로 설정합니다.
			cmmCfgSetSpeedRange ( nAxisNo, 6553500 );
		end;
	end;

end;
```

NAME

cmmCfgSetSpeedPattern
cmmCfgGetSpeedPattern
– 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 cmmCfgSetSpeedPattern

([in] VT_I4 Axis, [in] VT_I4 SpeedMode, [in] VT_R8 WorkSpeed, [in] VT_R8 Accel, [in] VT_R8 Decel)

❒ VT_I4 cmmCfgGetSpeedPattern

([in] VT_I4 Axis, [out] VT_PI4 SpeedMode, [out] VT_PR8 WorkSpeed, [out] VT_PR8 Accel, [out] VT_PR8 Decel)

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. 설정된 기준속도는 단축 제어의 경우 실제 모션 속도 지령 함수인 cmmSxSetSpeedRatio 와 같은 함수를 통해 그 비율로 모션 제어의 속도를 결정할 수 있습니다.

PARAMETER

▶ Axis: 축번호. 축번호는 상수값으로 [cmX1] 부터 0 번째 축을 기준 축으로 임의의 축을 설정할 수 있습니다.

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

Value	Meaning
0 또는 cmSMODE_C	CONSTANT 속도모드 => 가감속을 수행하지 않습니다.
1 또는 cmSMODE_T	TRAPEZOIDAL 속도모드 => 사다리꼴 가감속을 수행합니다.
2 또는 cmSMODE_S	S-CURVE 속도모드 => S-CURVE 가감속을 수행합니다.
-1 또는 cmSMODE_KEEP	이전 속도 모드는 그대로 설정합니다.

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

▶ SpeedMode : cmmCfgGetSpeedPattern 함수의 인자이며, 속도모드의 반환값입니다.

Value	Meaning
0 또는 cmSMODE_C	CONSTANT 속도모드 => 가감속을 수행하지 않습니다.
1 또는 cmSMODE_T	TRAPEZOIDAL 속도모드 => 사다리꼴 가감속을 수행합니다.
2 또는 cmSMODE_S	S-CURVE 속도모드 => S-CURVE 가감속을 수행합니다.

▶ WorkSpeed : cmmCfgSetSpeedPattern 함수의 인자이며, 작업 속도를 설정합니다.

▶ WorkSpeed : cmmCfgGetSpeedPattern 함수의 인자이며, 작업 속도를 반환합니다.

▶ Accel : cmmCfgSetSpeedPattern 함수의 인자이며, 가속도를 설정합니다.

▶ Accel : cmmCfgGetSpeedPattern 함수의 인자이며, 가속도를 반환합니다.

▶ Decel : cmmCfgSetSpeedPattern 함수의 인자이며, 감속도를 설정합니다.

▶ Decel : cmmCfgGetSpeedPattern 함수의 인자이며, 감속도를 반환합니다..

SEE ALSO

cmmSxSetSpeedRatio

RETURN VALUE

Value	Meaning
음수	수행 실패. 자세한 내용은 '에러처리' 편을 참고합니다
cmERR_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 cmmCfgSetUnitSpeed() 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 1]과 같이 Linear acceleration -> Working speed(constant) -> Linear deceleration 의 형태로 운용하는 모드입니다.

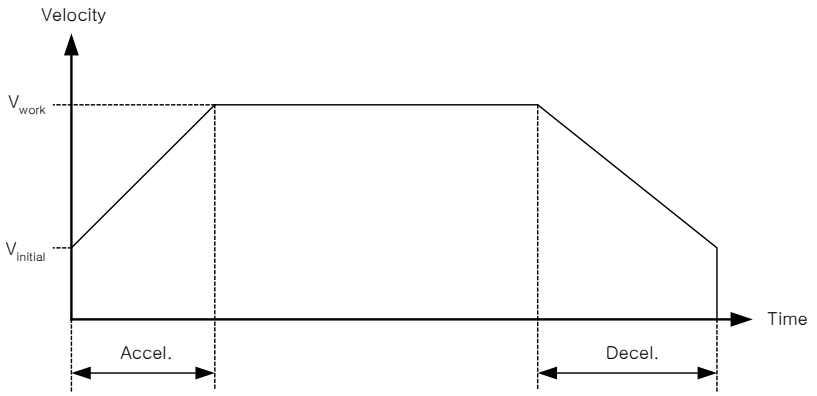


그림 7-1 Trapezoidal speed pattern

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.

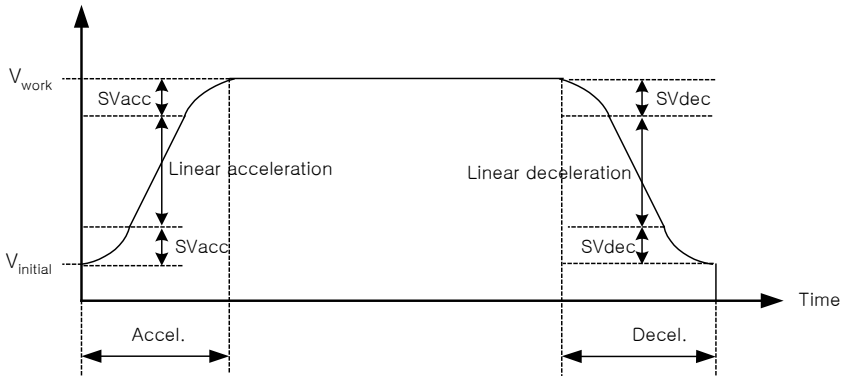


그림 7-2 S-curve speed pattern

※ 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. 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 “Cmmsdk.h”
#include “CmmsdkDef.h”

void OnSetSpeedPattern ()
{
        long nAxisNo = 0;                       // 모션 이송 기준 속도를 설정할 축을 선택합니다.
        long nSpeedMode;                        // 속도 패턴 정보.
        double fVel, fAcc, fDec;

        /* 0 번축의 속도 패턴을 S-Curve 로 설정하고,
        작업속도를 2000, 가속도를 10000, 감속도를 10000 으로 설정합니다.*/

        cmmCfgSetSpeedPattern ( nAxisNo,        // 대상 축 선택.
                                cmSMODE_S,      // 속도 모드 선택.
                                2000,           // 작업 속도
                                10000,          // 가속도
                                10000           // 감속도
                                );

        // 설정되어있는 속도 패턴 및 작업속도, 가속도, 감속도를 반환합니다.
        cmmCfgGetSpeedPattern ( nAxisNo, &nSpeedMode, &fVel, &fAcc, &fDec );
}
```

Visual Basic

```
Private Sub OnSetSpeedPattern ()

Dim nAxisNo As Long                          ' 모션 이송 기준 속도를 설정할 축을 선택합니다.
Dim nSpeedMode As Long                       ' 속도 패턴 정보.
Dim fVel As Double, fAcc As Double, fDec As Double

nAxisNo = 0

        ' 0 번축의 속도 패턴을 S-Curve 로 설정하고,
        ' 작업속도를 2000, 가속도를 10000, 감속도를 10000 으로 설정합니다.
        Call cmmCfgSetSpeedPattern ( nAxisNo, cmSMODE_S, 2000, 10000, 10000 )

        ' 설정되어있는 속도 패턴 및 작업속도, 가속도, 감속도를 반환합니다.
        Call cmmCfgGetSpeedPattern ( nAxisNo, nSpeedMode, fVel, fAcc, fDec )

End Sub
```

Delphi

```
procedure OnSetSpeedPattern ();
var
        nSpeedMode : LongInt;                        // 속도 패턴 정보.
        fVel, fAcc, fDec : Double;

begin
        { 0 번축의 속도 패턴을 S-Curve 로 설정하고,
        작업속도를 2000, 가속도를 10000, 감속도를 10000 으로 설정합니다. }
         cmmCfgSetSpeedPattern ( cmX1, cmSMODE_S, 2000, 10000, 10000 );

        // 설정되어있는 속도 패턴 및 작업속도, 가속도, 감속도를 반환합니다.
        cmmCfgGetSpeedPattern ( cmX1, @nSpeedMode, @fVel, @fAcc, @fDec );

end;
```

NAME

cmmCfgSetSpeedPattern_T
– 모션 이송 기준 속도 설정(設定)

INFORMATION

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

SYNOPSIS

❒VT_I4 cmmCfgSetSpeedPattern_T

([in] VT_I4 Axis, [in] VT_I4 SpeedMode, [in] VT_R8 WorkSpeed, [in] VT_R8 AccelTime,

[in] VT_R8 DecelTime)

DESCRIPTION

지정한 축에 대해 속도 모드, 작업 속도 및 가감속 시간을 설정 할 수 있습니다.
작업 속도는 각 모션 제어의 기준 속도로 설정되며, 가감속은 일반적인 속도 단위(PPS)가 아닌 시간(msec) 단위로 설정이 가능한 점에서 cmmCfgSetSpeedPattern 함수와 차이가 있습니다.
설정된 기준속도는 단축 제어의 경우 실제 모션 속도 지령 함수인 cmmSxSetSpeedRatio 와 같은 함수를 통해 그 비율로 모션 제어의 속도를 결정할 수 있습니다.

PARAMETER

▶ Axis: 축번호. 축번호는 상수값으로 [cmX1] 부터 0 번째 축을 기준 축으로 임의의 축을 설정할 수 있습니다.

▶ SpeedMode : 속도모드의 설정 값입니다. 아래와 같은 설정 값을 가집니다.

Value	Meaning
0 또는 cmSMODE_C	CONSTANT 속도모드 => 가감속을 수행하지 않습니다.
1 또는 cmSMODE_T	TRAPEZOIDAL 속도모드 => 사다리꼴 가감속을 수행합니다.
2 또는 cmSMODE_S	S-CURVE 속도모드 => S-CURVE 가감속을 수행합니다.
-1 또는 cmSMODE_KEEP	이전 속도 모드는 그대로 설정합니다.

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

▶ WorkSpeed : 작업 속도를 설정합니다.

▶ AccelTime : 가속 시간을 밀리초(msec) 단위로 설정합니다.

▶ DecelTime : 감속 시간을 밀리초(msec) 단위로 설정합니다.

SEE ALSO

cmmCfgSetSpeedPattern

RETURN VALUE

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

cmERR_NONE	수행 성공

REFERENCE

□ cmmCfgSetSpeedPattern 함수의 REFERENCE 참조

EXAMPLE

C/C++

```
#include “Cmmsdk.h”
#include “CmmsdkDef.h”

void OnSetSpeedPattern ()
{
	long nAxisNo = 0;				// 모션 이송 기준 속도를 설정할 축을 선택합니다.
	long nSpeedMode;				// 속도 패턴 정보.
	double fVel, fAccTime, fDecTime;

	/* 0 번축의 속도 패턴을 S-Curve 로 설정하고,
	작업속도를 2000, 가속 시간을 1 초(1000 ms), 감속 시간을 1 초(1000 ms)으로 설정합니다.*/

	 cmmCfgSetSpeedPattern_T ( nAxisNo,	// 대상 축 선택.
				cmSMODE_S,	// 속도 모드 선택.
				2000,		// 작업 속도
				1000,		// 가속 시간(ms)
				1000		// 감속 시간(ms)
				);

	// 설정되어있는 속도 패턴 및 작업속도, 가속 시간, 감속 시간을 반환합니다.
	cmmCfgGetSpeedPattern_T ( nAxisNo, &nSpeedMode, &fVel, &fAccTime, &fDecTime );
}
```

Visual Basic

```
Private Sub OnSetSpeedPattern ()

Dim nAxisNo As Long				‘ 모션 이송 기준 속도를 설정할 축을 선택합니다.
Dim nSpeedMode As Long				‘ 속도 패턴 정보.
Dim fVel As Double, fAccTime As Double, fDecTime As Double

nAxisNo = 0

	‘ 0 번축의 속도 패턴을 S-Curve 로 설정하고,
	‘작업속도를 2000, 가속 시간을 1 초(1000 ms), 감속 시간을 1 초(1000 ms)으로 설정합니다.
	Call cmmCfgSetSpeedPattern_T ( nAxisNo, cmSMODE_S, 2000, 1000, 1000 )

	‘ 설정되어있는 속도 패턴 및 작업속도, 가속 시간, 감속 시간을 반환합니다.
	Call cmmCfgGetSpeedPattern_T ( nAxisNo, nSpeedMode, fVel, fAccTime, fDecTime )

End Sub
```

Delphi

```
procedure OnSetSpeedPattern ();
var
	nSpeedMode : LongInt;				// 속도 패턴 정보.
	fVel, fAccTime, fDecTime : Double;

begin
	{ 0 번축의 속도 패턴을 S-Curve 로 설정하고,
	작업속도를 2000, 가속 시간을 1 초(1000 ms), 감속 시간을 1 초(1000 ms)으로 설정합니다. }
```

```
    cmmCfgSetSpeedPattern_T ( cmX1, cmSMODE_S, 2000, 1000, 1000 );

    // 설정되어있는 속도 패턴 및 작업속도, 가속 시간, 감속 시간을 반환합니다.
    cmmCfgGetSpeedPattern_T ( cmX1, @nSpeedMode, @fVel, @fAccTime, @fDecTime );

end;
```

NAME

cmmCfgGetSpeedPattern_T
- 모션 이송 기준 속도 설정(設定) 값 반환

INFORMATION

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

SYNOPSIS

❒VT_I4 cmmCfgGetSpeedPattern_T

([in] VT_I4 Axis, [out] VT_PI4 SpeedMode, [out] VT_PR8 WorkSpeed,

[out] VT_PR8 AccelTime, [out] VT_PR8 DecelTime)

DESCRIPTION

지정한 축에 대해 설정된 속도 모드, 작업 속도 및 가감속 시간을 반환 합니다.
작업 속도는 각 모션 제어의 기준 속도로 설정된 값이며, 가감속은 일반적인 속도 단위(PPS)가 아닌 시간(msec) 단위로 설정된 점에서 cmmCfgGetSpeedPattern 함수와 반환 값에 다소 차이가 있습니다.

PARAMETER

▶ Axis: 축번호. 축번호는 상수값으로 [cmX1] 부터 zero-based 로 설정 된 값입니다.

▶ SpeedMode : 속도모드의 설정 값입니다. 아래와 같은 설정 값을 가집니다.

Value	Meaning
0 또는 cmSMODE_C	CONSTANT 속도모드 => 가감속을 수행하지 않습니다.
1 또는 cmSMODE_T	TRAPEZOIDAL 속도모드 => 사다리꼴 가감속을 수행합니다.
2 또는 cmSMODE_S	S-CURVE 속도모드 => S-CURVE 가감속을 수행합니다.
-1 또는 cmSMODE_KEEP	이전 속도 모드는 그대로 설정합니다.

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

▶ WorkSpeed : 설정된 작업 속도를 반환 합니다.

▶ AccelTime : 설정된 가속 시간을 밀리초(msec) 단위로 반환 합니다.

▶ DecelTime : 설정된 감속 시간을 밀리초(msec) 단위로 반환 합니다.

SEE ALSO

cmmCfgGetSpeedPattern

RETURN VALUE

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

REFERENCE

□ cmmCfgSetSpeedPattern 함수의 REFERENCE 참조

EXAMPLE

C/C++

```
#include “Cmmsdk.h”
#include “CmmsdkDef.h”

void OnSetSpeedPattern ()
{
        long nAxisNo = 0;                       // 모션 이송 기준 속도를 설정할 축을 선택합니다.
        long nSpeedMode;                        // 속도 패턴 정보.
        double fVel, fAccTime, fDecTime;

        /* 0 번축의 속도 패턴을 S-Curve 로 설정하고,
        작업속도를 2000, 가속 시간을 1 초(1000 ms), 감속 시간을 1 초(1000 ms)으로 설정합니다.*/

         cmmCfgSetSpeedPattern_T ( nAxisNo,     // 대상 축 선택.
                                   cmSMODE_S,   // 속도 모드 선택.
                                   2000,        // 작업 속도
                                   1000,        // 가속 시간(ms)
                                   1000         // 감속 시간(ms)
                                   );

        // 설정되어있는 속도 패턴 및 작업속도, 가속 시간, 감속 시간을 반환합니다.
        cmmCfgGetSpeedPattern_T ( nAxisNo, &nSpeedMode, &fVel, &fAccTime, &fDecTime );
}
```

Visual Basic

```
Private Sub OnSetSpeedPattern ()

Dim nAxisNo As Long                             ‘ 모션 이송 기준 속도를 설정할 축을 선택합니다.
Dim nSpeedMode As Long                          ‘ 속도 패턴 정보.
Dim fVel As Double, fAccTime As Double, fDecTime As Double

nAxisNo = 0

        ‘ 0 번축의 속도 패턴을 S-Curve 로 설정하고,
        ‘작업속도를 2000, 가속 시간을 1 초(1000 ms), 감속 시간을 1 초(1000 ms)으로 설정합니다.
        Call cmmCfgSetSpeedPattern_T ( nAxisNo, cmSMODE_S, 2000, 1000, 1000 )

        ‘ 설정되어있는 속도 패턴 및 작업속도, 가속 시간, 감속 시간을 반환합니다.
        Call cmmCfgGetSpeedPattern_T ( nAxisNo, nSpeedMode, fVel, fAccTime, fDecTime )

End Sub
```

Delphi

```
procedure OnSetSpeedPattern ();
var
        nSpeedMode : LongInt;                           // 속도 패턴 정보.
        fVel, fAccTime, fDecTime : Double;

begin
        { 0 번축의 속도 패턴을 S-Curve 로 설정하고,
        작업속도를 2000, 가속 시간을 1 초(1000 ms), 감속 시간을 1 초(1000 ms)으로 설정합니다. }
         cmmCfgSetSpeedPattern_T ( cmX1, cmSMODE_S, 2000, 1000, 1000 );

        ‘ 설정되어있는 속도 패턴 및 작업속도, 가속 시간, 감속 시간을 반환합니다.
```

```
        cmmCfgGetSpeedPattern_T ( cmX1, @nSpeedMode, @fVel, @fAccTime, @fDecTime );

end;
```

NAME

cmmCfgSetMinAccTime
- 최소 가속 및 감속 시간 설정

INFORMATION

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

SYNOPSIS

❒VT_I4 cmmCfgSetMinAccTime

([in] VT_I4 Axis,[in] VT_R8 MinAccT, [in] VT_R8 MinDecT)

DESCRIPTION

cmmCfgSetMinAccTime() 함수는 해당 축에 대해 정격 속도 설정 외에 최소 가속 및 감속 시간을 추가로 설정해서 가감속에 필요한 최소 시간을 확보하기 위해 사용되어 집니다.

PARAMETER

▶ Axis: 축번호. 축번호는 상수값으로 [cmX1] 부터 0 번째 축을 기준 축으로 임의의 축을 설정할 수 있습니다.

▶ MinAccT : 보장되어야 하는 최소 가속 시간을 밀리초(msec) 단위로 설정합니다.

▶ MinDecT : 보장되어야 하는 최소 감속 시간을 밀리초(msec) 단위로 설정합니다.

SEE ALSO

cmmCfgGetMinAccTime

RETURN VALUE

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

EXAMPLE

C/C++

```
#include "Cmmsdk.h"
#include "CmmsdkDef.h"

void OnSetSpeedPatternEx ()
{
        long nAxisNo = 0;                       // 모션 이송 기준 속도를 설정할 축을 선택합니다.
        long nSpeedMode;                        // 속도 패턴 정보.
        double fVel, fAcc, fDec;
        double fMinAccTime, fMinDecTime;

        /* 0 번축의 속도 패턴을 S-Curve 로 설정하고,
        작업속도를 2000, 가속도를 10000, 감속도를 10000 으로 설정합니다.*/
```

```
        cmmCfgSetSpeedPattern ( nAxisNo,        // 대상 축 선택.
                                cmSMODE_S,      // 속도 모드 선택.
                                2000,           // 작업 속도
                                10000,          // 가속도
                                10000           // 감속도
                                );

        // 최소 가감속 시간을 각각 100 ms 로 설정합니다.
        cmmCfgSetMinAccTime ( nAxisNo, 100, 100 );

        // 설정된 최소 가감속 시간을 확인 합니다.
        cmmCfgGetMinAccTime( nAxisNo, &fMinAccTime, &fMinDecTime);
}
```

Visual Basic

```
Private Sub OnSetSpeedPatternEx ()

Dim nAxisNo As Long                         ' 모션 이송 기준 속도를 설정할 축을 선택합니다.
Dim nSpeedMode As Long                      ' 속도 패턴 정보.
Dim fVel As Double, fAcc As Double, fDec As Double
Dim fMinAccTime as Double, fMinDecTime as Double

nAxisNo = 0

        ' 0 번축의 속도 패턴을 S-Curve 로 설정하고,
        '작업속도를 2000, 가속 시간을 1 초(1000 ms), 감속 시간을 1 초(1000 ms)으로 설정합니다.
        Call cmmCfgSetSpeedPattern_T ( nAxisNo, cmSMODE_S, 2000, 10000, 10000 )

        ' 최소 가감속 시간을 각각 100 ms 로 설정합니다.
        Call cmmCfgSetMinAccTime ( nAxisNo, 100,100)

        ' 설정된 최소 가감속 시간을 확인 합니다.
        Call cmmCfgGetMinAccTime ( nAxisNo, fMinAccTime, fMinDecTime)

End Sub
```

Delphi

```
procedure OnSetSpeedPatternEx ();
var
        nAxisNo : LongInt;
        nSpeedMode : LongInt;                       // 속도 패턴 정보.
        fVel, fAcc, fDec : Double;
        fMinAccTime, fMinDecTime : Double;

begin
        { 0 번축의 속도 패턴을 S-Curve 로 설정하고,
        작업속도를 2000, 가속 시간을 1 초(1000 ms), 감속 시간을 1 초(1000 ms)으로 설정합니다. }
         cmmCfgSetSpeedPattern ( cmX1, cmSMODE_S, 2000, 10000, 10000 );

        // 최소 가감속 시간을 각각 100 ms 로 설정합니다.
        cmmCfgSetMinAccTime ( nAxisNo, 100, 100 );

        // 설정된 최소 가감속 시간을 확인 합니다.
        cmmCfgGetMinAccTime( nAxisNo, @fMinAccTime, @fMinDecTime);

end;
```

NAME

cmmCfgGetMinAccTime
– 최소 가속 및 감속 시간 설정 값 반환

INFORMATION

Environment
Configuration Functions
VC++/VB
BCB/Delphi/.NET
Level 4
위험 요소 없음

SYNOPSIS

❒VT_I4 cmmCfgGetMinAccTime

([in] VT_I4 Axis,[out] VT_PR8 MinAccT, [out] VT_PR8 MinDecT)

DESCRIPTION

cmmCfgGetMinAccTime() 함수는 해당 축에 대해 설정된 최소 가속 및 감속 시간을 확인 하기 위해 사용 됩니다.

PARAMETER

▶ Axis: 축번호. 축번호는 상수값으로 [cmX1] 부터 zero-based 로 설정됩니다.

▶ MinAccT : 최소 가속 시간을 밀리초(msec) 단위로 반환 합니다.

▶ MinDecT : 최소 감속 시간을 밀리초(msec) 단위로 반환 합니다.

SEE ALSO

cmmCfgSetMinAccTime

RETURN VALUE

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

EXAMPLE

C/C++

```
#include "Cmmsdk.h"
#include "CmmsdkDef.h"

void OnSetSpeedPatternEx ()
{
        long nAxisNo = 0;                       // 모션 이송 기준 속도를 설정할 축을 선택합니다.
        long nSpeedMode;                        // 속도 패턴 정보.
        double fVel, fAcc, fDec;
        double fMinAccTime, fMinDecTime;

        /* 0 번축의 속도 패턴을 S-Curve 로 설정하고,
        작업속도를 2000, 가속도를 10000, 감속도를 10000 으로 설정합니다.*/

        cmmCfgSetSpeedPattern ( nAxisNo,        // 대상 축 선택.
```

```
                                cmSMODE_S,      // 속도 모드 선택.
                                2000,           // 작업 속도
                                10000,          // 가속도
                                10000           // 감속도
                                );

        // 최소 가감속 시간을 각각 100 ms 로 설정합니다.
        cmmCfgSetMinAccTime ( nAxisNo, 100, 100 );

        // 설정된 최소 가감속 시간을 확인 합니다.
        cmmCfgGetMinAccTime( nAxisNo, &fMinAccTime, &fMinDecTime);
}
```

Visual Basic

```
Private Sub OnSetSpeedPatternEx ()

Dim nAxisNo As Long                             ' 모션 이송 기준 속도를 설정할 축을 선택합니다.
Dim nSpeedMode As Long                          ' 속도 패턴 정보.
Dim fVel As Double, fAcc As Double, fDec As Double
Dim fMinAccTime as Double, fMinDecTime as Double

nAxisNo = 0

        ' 0 번축의 속도 패턴을 S-Curve 로 설정하고,
        '작업속도를 2000, 가속 시간을 1 초(1000 ms), 감속 시간을 1 초(1000 ms)으로 설정합니다.
        Call cmmCfgSetSpeedPattern_T ( nAxisNo, cmSMODE_S, 2000, 10000, 10000 )

        ' 최소 가감속 시간을 각각 100 ms 로 설정합니다.
        Call cmmCfgSetMinAccTime ( nAxisNo, 100,100)

        ' 설정된 최소 가감속 시간을 확인 합니다.
        Call cmmCfgGetMinAccTime ( nAxisNo, fMinAccTime, fMinDecTime)

End Sub
```

Delphi

```
procedure OnSetSpeedPatternEx ();
var
        nAxisNo : LongInt;
        nSpeedMode : LongInt;                                   // 속도 패턴 정보.
        fVel, fAcc, fDec : Double;
        fMinAccTime, fMinDecTime : Double;

begin
        { 0 번축의 속도 패턴을 S-Curve 로 설정하고,
        작업속도를 2000, 가속 시간을 1 초(1000 ms), 감속 시간을 1 초(1000 ms)으로 설정합니다. }
         cmmCfgSetSpeedPattern ( cmX1, cmSMODE_S, 2000, 10000, 10000 );

        // 최소 가감속 시간을 각각 100 ms 로 설정합니다.
        cmmCfgSetMinAccTime ( nAxisNo, 100, 100 );

        // 설정된 최소 가감속 시간을 확인 합니다.
        cmmCfgGetMinAccTime( nAxisNo, @fMinAccTime, @fMinDecTime);

end;
```

NAME

cmmCfgSetActSpdCheck
cmmCfgGetActSpdCheck
– 실제 모션 속도 검출(檢出) 설정(設定) 및 반환 (返還)

INFORMATION

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

SYNOPSIS

❒ VT_I4 cmmCfgSetActSpdCheck

([in] VT_I4 IsEnable, [in] VT_I4 Interval)

❒ VT_I4 cmmCfgGetActSpdCheck

([out] VT_PI4 IsEnable, [out] VT_PI4 Interval)

DESCRIPTION

cmmCfgSetActSpdCheck() 함수는 실제 모션의 속도를 확인(確認)하는 기능을 Enable 시킵니다. 모션의 실제속도는 Feedback 펄스 수를 주기적으로 확인(確認)하여 펄스수의 변화량을 시간으로 나누어 계산됩니다. Feedback 속도 확인(確認) 기능은 필요한 경우에만 활성화하는 것이 좋습니다. Feedback 속도 확인(確認) 기능을 활성화하면, cmmStGetSpeed() 함수를 사용하여 입력 펄스 (Feedback Pulse)의 시간에 따른 변화량에 의하여, 실제 속도를 반환 받을 수 있습니다.

SEE ALSO 에 표기된 cmmCfgGetActSpdCheck 함수를 통해 현재 설정된 Feedback 속도 확인(確認) 기능의 환경설정 값을 읽어 들입니다.

PARAMETER

▶ IsEnable : cmmCfgSetActSpdCheck 함수의 인자이며, Feedback 속도 확인(確認)기능을 활성 또는 비활성화합니다.

▶ Interval : cmmCfgSetActSpdCheck 함수의 인자이며, Feedback 펄스의 수를 확인(確認)하는 주기를 msec 단위로 설정합니다.

▶ IsEnable : cmmCfgGetActSpdCheck 함수의 인자이며, Feedback 속도 확인(確認)기능의 활성여부를 반환합니다.

▶ Interval : cmmCfgGetActSpdCheck 함수의 인자이며, Feedback 펄스의 수를 확인(確認)하는 주기를 msec 단위로 반환합니다.

SEE ALSO

cmmStGetSpeed

REFERENCE

□ 모션의 실제속도는 Feedback 펄스의 수를 확인(確認)하는 주기를 msec 단위로 설정하면 설정된 주기마다 Feedback 펄스 수를 주기적으로 확인(確認)하여 펄스수의 변화량을 시간으로 나누어 계산됩니다. Feedback 펄스의 주파수는 다음과 같은 식에 의해 계산됩니다.

$$V_a = \frac{\Delta C}{\Delta T} \times \frac{1}{R_u} \times R_{io}$$

여기서,

V_a : Feedback 펄스를 통하여 계측되는 모션의 실제 속도

ΔC : 확인(確認) 주기 동안에 변화된 Feedback counter 값

ΔT : 확인(確認)주기, 사용자가 지정한 Interval 이 msec 단위로 설정되기 때문에 ΔT 는 Interval/1000 를 의미합니다.

R_u : Unit Speed, cmmCfgSetUnitSpeed() 함수를 참조합니다.

R_{io} : In/Out Ratio, cmmCfgSetInOutRatio() 함수를 참조합니다.

NAME

cmmCfgSetSoftLimit
cmmCfgGetSoftLimit
– 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 cmmCfgSetSoftLimit

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

❒ VT_I4 cmmCfgGetSoftLimit

([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 리미트를 설정하는 것입니다. 소프트웨어적인 Limit 은 Command pulse 카운터의 절대값이 지정한 +/- Limit 값보다 같거나 크게 되면 모션을 자동 정지(停止)하도록 합니다.

PARAMETER

▶ Axis : 축번호. 축 번호는 0 부터 시작합니다.

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

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

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

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

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

▶ LimitP : cmmCfgGetSoftLimit 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.

EXAMPLE

C/C++

```
#include “Cmmsdk.h”
#include “CmmsdkDef.h”

long nAxisNo = 1;                       // Software Limit 을 설정할 축을 선택합니다.

void OnSetSoftLimit ()
{
        long nIsEnable;                 // Software Limit 정보.
        double fElNeg, fElPos;

        /* Software Limit 설정 상태를 확인하고, Software Limit 기능이 비활성 상태이면 활성 상태로 설정합니다. */
        if (cmmCfgGetSoftLimit ( nAxisNo, &nIsEnable, &fElNeg, &fElPos ) == cmERR_NONE )
        {
                if ( nIsEnable != cmTRUE)
                {
                        // Software Limit 기능을 활성화 하고, -EL 은 -100000, +EL 은 100000 으로 설정합니다.
                        cmmCfgSetSoftLimit ( nAxisNo, cmTRUE, -100000, 100000 );
                }
        }
}

void OnMove ()
{
        /* 현재 위치 : 50000 이라 가정, 이송 거리 : 100000 으로 이송 명령을 주면 SW Limit 이송 제한 범위에 걸려
         150000 까지 Command pulse 를 출력하지 못하고 100000 위치에서 자동으로 정지합니다. */
        cmmSxMoveStart ( nAxisNo, 100000 );
}
```

Visual Basic

```
Private Sub OnSetSoftLimit ()

        Dim nIsEnable As Long                                   ‘ Software Limit 정보.
        Dim fElNeg As Double, fElPos As Double

        ‘ Software Limit 설정 상태를 확인하고, Software Limit 기능이 비활성 상태이면 활성 상태로 설정합니다.

        If cmmCfgGetSoftLimit (cmX1, nIsEnable, fElNeg, fElPos ) = cmERR_NONE Then

                If nIsEnable <> cmTRUE Then
                        ‘ Software Limit 기능을 활성화 하고, -EL 은 -100000, +EL 은 100000 으로 설정합니다.
                        Call cmmCfgSetSoftLimit (cmX1, cmTRUE, -100000, 100000 )
                End If
        End If
End Sub


Private Sub OnMove ()

        ‘ 현재 위치 : 50000 이라 가정, 이송 거리 : 100000 으로 이송 명령을 주면 SW Limit 이송 제한 범위에 걸려
        ‘ 150000 까지 Command pulse 를 출력하지 못하고 100000 위치에서 자동으로 정지합니다.
        Call cmmSxMoveStart (cmX1, 100000 )

End Sub
```

Delphi

```
procedure OnSetSoftLimit ();
var
        nIsEnable : LongInt;                            // Software Limit 정보.
        fElNeg, fElPos : Double;

begin
        // Software Limit 설정 상태를 확인하고, Software Limit 기능이 비활성 상태이면 활성 상태로 설정합니다.

        if cmmCfgGetSoftLimit (cmX1, @nIsEnable, @fElNeg, @fElPos ) = cmERR_NONE then
        begin
                if nIsEnable <> cmTRUE then
                        // Software Limit 기능을 활성화 하고, -EL 은 -100000, +EL 은 100000 으로 설정합니다.
                        cmmCfgSetSoftLimit (cmX1, cmTRUE, -100000, 100000 );
        end;
end;

procedure OnMove ();
begin
        { 현재 위치 : 50000 이라 가정, 이송 거리 : 100000 으로 이송 명령을 주면 SW Limit 이송 제한 범위에 걸려
         150000 까지 Command pulse 를 출력하지 못하고 100000 위치에서 자동으로 정지합니다. }
        cmmSxMoveStart (cmX1, 100000 );

end;
```

NAME

cmmCfgSetRingCntr
cmmCfgGetRingCntr
- 링 카운터(Ring-Counter) 기능 설정(設定) 및 반환(返還)

INFORMATION

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

SYNOPSIS

❒ VT_I4 cmmCfgSetRingCntr ([in] VT_I4 Axis, [in] VT_I4 TargCntr, [in] VT_I4 IsEnable, [in] VT_R8 CntMax)

❒ VT_I4 cmmCfgGetRingCntr ([in] VT_I4 Axis, [in] VT_I4 TargCntr, [out] VT_PI4 IsEnable, [out] VT_PR8 CntMax)

DESCRIPTION

cmmCfgSetRingCntr 함수는 링 카운터 기능의 활성/비활성 상태 설정 및 링 카운터 범위를 설정합니다. 해당 모션 축(Axis) 의 Command 또는 Feedback 카운터를 대상으로 링 카운터를 설정합니다..

cmmCfgGetRingCntr 함수는 링 카운터 기능의 설정 상태를 반환합니다.

PARAMETER

▶ Axis : 축(채널) 번호. 축 번호는 0 부터 시작합니다.

▶ TargCntr : 링 카운터 기능 대상 카운터를 설정합니다.

Value	Meaning
0 (cmCNT_COMM)	Command Counter
1 (cmCNT_FEED)	Feedback Counter

▶ IsEnable : cmmCfgSetRingCntr 함수의 인자이며, 링 카운터 기능 활성/비활성 상태를 설정합니다.

Value	Meaning
0 (cmFALSE)	링 카운터 기능을 사용하지 않습니다.
1 (cmTRUE)	링 카운터 기능을 사용합니다.

▶ IsEnable : cmmCfgGetRingCntr 함수의 인자이며, 링 카운터 기능 활성/비활성 상태를 반환합니다.

▶ CntMax : cmmCfgSetRingCntr 함수의 인자이며, 링 카운터 범위를 설정합니다. 링 카운터 기능이 활성화되면 지정한 카운터는 0~CntMax 사이의 값에서만 카운트 됩니다. 지정하는 카운터 값의 단위는 “Unit distance”에 의해 정의되는 논리적 거리 단위입니다

▶ CntMax : cmmCfgGetRingCntr 함수의 인자이며, 링 카운터 범위를 반환합니다.

RETURN VALUE

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

REFERENCE

링 카운트 기능 미지원 제품 안내.

이 기능은 COMI-LX502 제품에서는 지원하지 않습니다.

링 카운트 기능 사용 시 주의 사항 안내.

링 카운트 기능 활성화 시 소프트웨어 리밋 설정 환경이 무효화 됩니다.

즉, 링카운트 기능 과 소프트웨어 리밋 기능을 같이 사용할 수 없습니다. 소프트웨어 리밋 기능이 설정되어 있던 상태라도 링 카운트 기능 사용 후라면, 소프트웨어 리밋 기능을 재설정 하셔야 합니다.

EXAMPLE

C/C++

```
#include "Cmmsdk.h"
#include "CmmsdkDef.h"

long nAxisNo = 1;              // 링 카운터 기능을 설정할 축을 선택합니다.

void OnSetRingCounter ()
{
        long nIsEnable;                  // 링 카운터 설정 정보.
        double fCntMax;

        /* 커맨드 카운터를 대상으로 링카운터 기능을 활성화합니다.
        40000 펄스가 되면 커맨드 카운터가 다시 '0'부터 카운트 됩니다.*/
        cmmCfgSetRingCntr ( nAxisNo,                    // 대상 축 설정
                            cmCNT_COMM,                 // 링 카운터 대상 카운터 설정
                            cmTRUE,             // 링 카운터 기능 활성/비활성 설정
                            40000                       // 링 카운터 범위 설정
                            );

        // 해당 축의 링카운터 설정 상태를 반환합니다.
        cmmCfgGetRingCntr ( nAxis, cmCNT_COMM, &nIsEnable, &fCntMax );
}


void OnMove ()
{
        /* 링 카운터 범위가 '0 ~ 40000'으로 설정되어 있으므로 Command Count 값이 40000 이 되면
        자동으로 0 에서 Command Count 를 계수 합니다. */

cmmSxVMoveStart ( nAxisNo,1 );
}
```

Visual Basic

```
Dim nAxisNo As Long          ' 링 카운터 기능을 설정할 축을 선택합니다.
nAxisNo = 1

Private Sub OnSetRingCounter ()

        Dim nIsEnable As Long            ' 링 카운터 설정 정보.
        Dim fCntMax As Double

        ' 커맨드 카운터를 대상으로 링카운터 기능을 활성화합니다.
        ' 40000 펄스가 되면 커맨드 카운터가 다시 '0'부터 카운트 됩니다.
        Call cmmCfgSetRingCntr ( nAxisNo, cmCNT_COMM, cmTRUE, 40000 )

        ' 해당 축의 링카운터 설정 상태를 반환합니다.
        Call cmmCfgGetRingCntr ( nAxis, cmCNT_COMM, nIsEnable, fCntMax )

End Sub


Private Sub OnMove ()

        ' 링 카운터 범위가 '0 ~ 40000'으로 설정되어 있으므로 Command Count 값이 40000 이 되면
        ' 자동으로 0 에서 Command Count 를 계수 합니다.

        Call cmmSxVMoveStart ( nAxisNo ,1)

End Sub
```

Delphi

```
procedure OnSetRingCounter ();
var
        nIsEnable : LongInt;   // 링 카운터 설정 정보.
        fCntMax : Double;

begin
        { 커맨드 카운터를 대상으로 링카운터 기능을 활성화합니다.
        40000 펄스가 되면 커맨드 카운터가 다시 '0'부터 카운트 됩니다. }
        cmmCfgSetRingCntr (cmX1, cmCNT_COMM, cmTRUE, 40000 );

        // 해당 축의 링카운터 설정 상태를 반환합니다.
        cmmCfgGetRingCntr (cmX1, cmCNT_COMM, @nIsEnable, @fCntMax );

end;


procedure OnMove ();
begin
        { 링 카운터 범위가 '0 ~ 40000'으로 설정되어 있으므로 Command Count 값이 40000 이 되면
        자동으로 0 에서 Command Count 를 계수 합니다. }

        cmmSxVMoveStart (cmX1,1 );

end;
```

NAME

cmmCfgSetVelCorrRatio
cmmCfgGetVelCorrRatio
– 작업 속도 보정에 대한 속도 산출 비례 설정(設定) 및 반환(返還)

INFORMATION

Environment
Configuration Functions
VC++/VB
BCB/Delphi/.NET
Level 2
다소 주의
특별한 경우가 아니면 이 함수는 사용하지 않습니다.

SYNOPSIS

❒ VT_I4 cmmCfgSetVelCorrRatio

([in] VT_I4 Axis, [in] VT_R8 CorrRatio)

❒ VT_I4 cmmCfgGetVelCorrRatio

([in] VT_I4 Axis, [out] VT_PR8 CorrRatio)

DESCRIPTION

이 함수는 작업속도 보정 시에 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 CMMSDK 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.

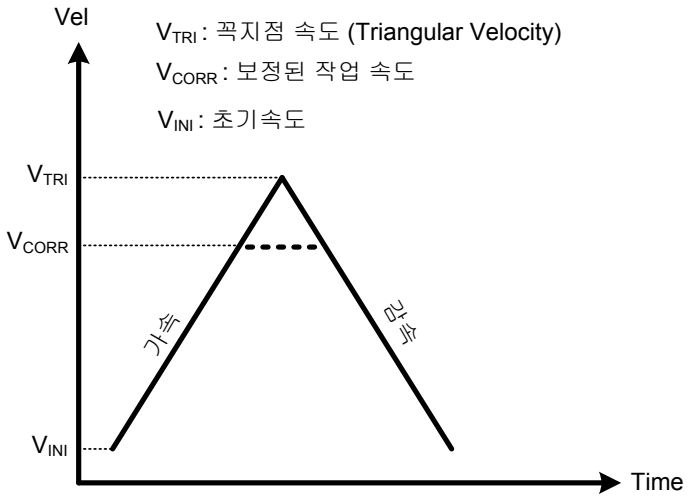


그림 7-3 꼭지점속도와 보정속도

보정속도는 꼭지점 속도에 속도보정비 값을 곱하여 산출합니다.

cmmCfgSetVelCorrRatio() 함수는 속도 보정비를 설정하는 함수이며, 기본적으로는 92(%)가 적용됩니다

PARAMETER

▶ Axis: 축번호. 축번호는 상수값으로 [cmX1] 부터 0 번째 축을 기준 축으로 임의의 축을 설정할 수 있습니다.

▶ fCorrRatio : cmmCfgSetVelCorrRatio 함수의 인자이며, 수정하고자 하는 속도 보정 비율값을 % 단위로 전달합니다.

▶ fCorrRatio : cmmCfgGetVelCorrRatio 함수의 인자이며, 수정하고자 하는 속도 보정 비율값을 % 단위로 반환합니다.

SEE ALSO

cmmCfgGetVelCorrRatio

RETURN VALUE

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

REFERENCE

□ 속도 보정비는 기본적으로 92(%)가 적용되고 있습니다. 따라서 특별한 경우가 아니면 이 함수를 사용할 필요가 없습니다.

NAME

cmmCfgSetFilterAB
cmmCfgGetFilterAB
– EA/EB, PA/PB 신호 노이즈 필터 기능 설정(設定) 및 반환(返還)

INFORMATION

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

SYNOPSIS

❒ VT_I4 cmmCfgSetFilterAB

([in] VT_I4 Channel, [in] VT_I4 Target, [in] VT_I4 IsEnable)

❒ VT_I4 cmmCfgGetFilterAB

([in] VT_I4 Channel, [in] VT_I4 Target, [out] VT_PI4 IsEnabled)

DESCRIPTION

cmmCfgSetFilterAB() 함수는 EA/EB(Encoder Feedback) 신호와 PA/PB(Manual Pulsar) 신호의 입력 회로에 입력 필터 적용을 위해 설정 혹은 해당 설정 매개변수를 반환 하는 함수입니다. 필터를 적용하게 되면 펄스의 폭이 308 ns 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 3.25 MHz 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 매개변수(媒介變數)를 통해서 함수의 적용 대상을 구분합니다.

cmmCfgSetFilterAB() 함수는 EA/EB 입력회로 또는 PA/PB 입력회로에 필터를 적용하는 지에 대하여 현재 장치에 설정된 상태를 읽어 들이는 함수입니다.

PARAMETER

▶ Channel : 축(채널) 번호. 축번호는 상수값으로 [cmX1] 부터 0 번째 채널을 기준 채널로 임의의 채널을 설정할 수 있습니다.

▶ Target : 함수의 적용 대상을 결정합니다. 이 매개 변수(媒介變數)에 적용 가능한 값은 다음과 같습니다.

Value	Meaning
0 (cmAB_ENC)	이 함수의 적용 대상이 EA/EB 신호임을 의미합니다.
1 (cmAB_PULSAR)	이 함수의 적용 대상이 PA/PB 신호임을 의미합니다.

▶ IsEnable : cmmCfgSetFilterAB 함수의 인자이며, 필터로직의 적용 여부를 설정합니다.

Value	Meaning
0 **[초기값]**	Filter disable
1	Filter enable

▶ IsEnable : cmmCfgGetFilterAB 함수의 인자이며, 필터로직의 적용 여부를 반환합니다.

Value	Meaning
0 **[초기값]**	Filter disable
1	Filter enable

RETURN VALUE

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

REFERENCE

□ 필터 로직 적용이 Enable 되면 3.25 MHz 이상의 주파수를 가지는 펄스는 노이즈로 간주되어서 카운트되지 않습니다.

EXAMPLE

C/C++

```
#include "Cmmsdk.h"
#include "CmmsdkDef.h"

void OnSetNoiseFilterAB ()
{
	long nAxisNo = 1;			// Noise Filter 기능을 적용할 대상 축 선택.
	long nFilterEnable;			// Noise Filter 기능 활성 상태 정보.


	/* EA/EB, PA/PB 신호 입력 회로에 노이즈 필터 기능 활성 여부를 확인 후,
	비활성 상태이면 활성 상태로 설정합니다. */

	// EA/EB 신호 입력 회로에 노이즈 필터 기능 적용
	if (cmmCfgGetFilterAB ( nAxisNo, cmAB_ENC, &nFilterEnable ) == cmERR_NONE )
	{
		if ( nFilterEnable != cmTRUE )
		{
			// EA/EB Noise Filter Enable
			cmmCfgSetFilterAB ( nAxisNo, cmAB_ ENC, cmTRUE );
		}
	}

	// PA/PB 신호 입력 회로에 노이즈 필터 기능 적용
	if (cmmCfgGetFilterAB ( nAxisNo, cmAB_ PULSAR, &nFilterEnable ) == cmERR_NONE )
	{
		if ( nFilterEnable != cmTRUE )
		{
			// PA/PB Noise Filter Enable
			cmmCfgSetFilterAB ( nAxisNo, cmAB_PULSAR, cmTRUE );
		}
	}
}
```

Visual Basic

```
Private Sub OnSetNoiseFilterAB ()

	Dim nAxisNo As Long			' Noise Filter 기능을 적용할 대상 축 선택.
	Dim nFilterEnable As Long		' Noise Filter 기능 활성 상태 정보.

	nAxisNo = 1


	' EA/EB, PA/PB 신호 입력 회로에 노이즈 필터 기능 활성 여부를 확인 후,
	' 비활성 상태이면 활성 상태로 설정합니다.

	' EA/EB 신호 입력 회로에 노이즈 필터 기능 적용
	If cmmCfgGetFilterAB ( nAxisNo, cmAB_ENC, nFilterEnable ) = cmERR_NONE Then

		If nFilterEnable <> cmTRUE Then
			' EA/EB Noise Filter Enable
```

```
                    Call cmmCfgSetFilterAB ( nAxisNo, cmAB_ENC, cmTRUE )
                End If
        End If

        ' PA/PB 신호 입력 회로에 노이즈 필터 기능 적용
        If cmmCfgGetFilterAB ( nAxisNo, cmAB_PULSAR, nFilterEnable ) = cmERR_NONE Then

                If nFilterEnable <> cmTRUE Then
                        ' PA/PB Noise Filter Enable
                        Call cmmCfgSetFilterAB ( nAxisNo, cmAB_PULSAR, cmTRUE )
                End If
        End If

End Sub
```

Delphi

```
procedure OnSetNoiseFilterAB ();
var
        nAxisNo : LongInt;                      // Noise Filter 기능을 적용할 대상 축 선택.
        nFilterEnable : LongInt;                // Noise Filter 기능 활성 상태 정보.

begin
        nAxisNo := 1;


        { EA/EB, PA/PB 신호 입력 회로에 노이즈 필터 기능 활성 여부를 확인 후,
        비활성 상태이면 활성 상태로 설정합니다. }

        // EA/EB 신호 입력 회로에 노이즈 필터 기능 적용
        if cmmCfgGetFilterAB ( nAxisNo, cmAB_ENC, @nFilterEnable ) = cmERR_NONE then
        begin
                if nFilterEnable <> cmTRUE then
                begin
                        // EA/EB Noise Filter Enable
                        cmmCfgSetFilterAB ( nAxisNo, cmAB_ENC, cmTRUE );
                end;
        end;

        // PA/PB 신호 입력 회로에 노이즈 필터 기능 적용
        if cmmCfgGetFilterAB ( nAxisNo, cmAB_PULSAR, @nFilterEnable ) = cmERR_NONE then
        begin
                if nFilterEnable <> cmTRUE then
                begin
                        // PA/PB Noise Filter Enable
                        cmmCfgSetFilterAB ( nAxisNo, cmAB_PULSAR, cmTRUE );
                end;
        end;

end;
```

NAME

cmmCfgSetCtrlMode
cmmCfgGetCtrlMode
– 각 축의 제어 모드 설정(設定) 및 반환(返還)

INFORMATION

Environment
Configuration Functions
VC++/VB
BCB/Delphi/.NET
Level 2
다소 주의
전체 이송 목표 좌표의 기준을 변경하는 설정입니다.

SYNOPSIS

❒ VT_I4 cmmCfgSetCtrlMode

([in] VT_I4 Axis, [in] VT_I4 CtrlMode)

❒ VT_I4 cmmCfgGetCtrlMode

([in] VT_I4 Axis, [out] VT_PI4 CtrlMode)

DESCRIPTION

cmmCfgSetCtrlMode() 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.

cmmCfgGetCtrlMode() 함수는 각 축의 제어모드에 대하여 현재 장치에 설정되어 있는 값을 읽어 들이는 함수입니다.

PARAMETER

▶ Axis: 축번호. 축번호는 상수값으로 [cmX1] 부터 0 번째 축을 기준 축으로 임의의 축을 설정할 수 있습니다.

▶ CtrlMode : cmmCfgSetCtrlMode 함수의 인자이며, 제어모드를 설정합니다. 이 값의 의미는 다음과 같습니다.

Value	Meaning
0 (cmCTRL_OPEN) **[초기값]**	이 값은 제어모드를 Open loop 제어 모드로 설정합니다. 이 모드에서는 이송명령의 목표 좌표가 항상 커맨드 위치를 기준으로 설정됩니다.
1 (cmCTRL_SEMI_C)	이 값은 제어모드를 Semi-closed loop 제어 모드로 설정합니다. 이 모드에서는 이송명령의 목표 좌표가 항상 FEEDBACK 위치를 기준으로 설정됩니다. 예를 들어서 10000 의 좌표로 이송하라는 명령이 하달되었을 때 커맨드 위치는 무시하고, 피드백 위치가 10000 이 되도록 이송의 목표좌표가 설정됩니다.
2 (cmCTRL_FULL_C)	이 값은 향후 버전을 위해서 예약된 값이며, 현재는 사용되지 않는 값입니다.

▶ CtrlMode : cmmCfgGetCtrlMode 함수의 인자이며, 제어모드를 반환합니다. 반환값의 의미는 다음과 같습니다.

Value	Meaning
0 (cmCTRL_OPEN) **[초기값]**	Open loop 제어 모드입니다. 이 모드에서는 이송명령의 목표 좌표가 항상 커맨드 위치를 기준으로 설정됩니다.
1 (cmCTRL_SEMI_C)	Semi-closed loop 제어 모드입니다. 이 모드에서는 이송명령의 목표 좌표가 항상 FEEDBACK 위치를 기준으로 설정됩니다.

RETURN VALUE

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

SEE ALSO

cmmSxMoveTo, cmmSxMoveToStart, cmmMxMoveTo, cmmMxMoveToStart, cmmIxLineTo, cmmIxLineToStart

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 이해(理解)를 돕기 위하여 예를 들어보겠습니다. 현재 커맨드 위치가 5000 , 피드백 위치가 4000 인 상태에서 cmmSxMoveTo() 함수를 사용하여 10000 의 위치로 이송하라는 명령을 하달하였을 때

- Open loop 제어모드에서는 현재 위치에서 5000 만큼 (+) 방향으로 이송
- Semi-Closed Loop 제어모드에서는 현재 위치에서 6000 만큼 (+) 방향으로 이송

을 하는 서로 다른 이송 결과를 나타낼 수 있습니다.

□ Semi-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 "Cmmsdk.h"
#include "CmmsdkDef.h"

void OnSetCtrlMode ()
{
	long nAxisNo = 1;			// 제어 모드를 설정할 축을 선택합니다.
	long nCtrlMode;				// 제어 모드 정보.

	/* 설정되어 있는 제어 모드를 확인하여 Semi-Closed 제어 모드로 설정합니다.
	이 제어 모드에서는 이송 명령의 목표 좌표가 항상 피드백 위치를 기준으로 설정됩니다. 단, 절대 좌표 이송
	명령에서만 영향을 미치고, 상대 좌표 이송 명령에서는 Open Loop 제어 모드로 운용됩니다. */


	// 해당 축의 제어 모드 설정 상태를 확인하여, Semi-Closed Loop 제어 모드로 설정합니다.
	if (cmmCfgGetCtrlMode ( nAxisNo, nCtrlMode ) == cmERR_NONE )
	{
		if ( nCtrlMode != cmCTRL_SEMI_C )
		{
			// Semi-Closed Loop 제어 모드로 설정
			cmmCfgSetCtrlMode ( nAxisNo, cmCTRL_OPEN );
		}
	}
}
```

Visual Basic

```
Private Sub OnSetCtrlMode ()

	Dim nAxisNo As Long		' 제어 모드를 설정할 축을 선택합니다.
	Dim nCtrlMode As Long		' 제어 모드 정보.
```

```
    nAxisNo = 1

    ‘ 설정되어 있는 제어 모드를 확인하여 Semi-Closed 제어 모드로 설정합니다.
    ‘ 이 제어 모드에서는 이송 명령의 목표 좌표가 항상 피드백 위치를 기준으로 설정됩니다. 단, 절대 좌표 이송
    ‘ 명령에서만 영향을 미치고, 상대 좌표 이송 명령에서는 Open Loop 제어 모드로 운용됩니다.

    ‘ 해당 축의 제어 모드 설정 상태를 확인하여, Semi-Closed Loop 제어 모드로 설정합니다.
    If cmmCfgGetCtrlMode ( nAxisNo, nCtrlMode ) = cmERR_NONE Then

        If nCtrlMode <> cmCTRL_SEMI_C Then
            ‘ Semi-Closed Loop 제어 모드로 설정
            Call cmmCfgSetCtrlMode ( nAxisNo, cmCTRL_OPEN )
        End If
    End If

End Sub
```

```
Delphi

procedure OnSetCtrlMode ();
var
    nAxisNo : LongInt;          // 제어 모드를 설정할 축을 선택합니다.
    nCtrlMode : LongInt;        // 제어 모드 정보.

begin
    nAxisNo := 1;

    { 설정되어 있는 제어 모드를 확인하여 Semi-Closed 제어 모드로 설정합니다.
    ‘이 제어 모드에서는 이송 명령의 목표 좌표가 항상 피드백 위치를 기준으로 설정됩니다. 단, 절대 좌표 이송
     명령에서만 영향을 미치고, 상대 좌표 이송 명령에서는 Open Loop 제어 모드로 운용됩니다. }


    // 해당 축의 제어 모드 설정 상태를 확인하여, Semi-Closed Loop 제어 모드로 설정합니다.
    if cmmCfgGetCtrlMode ( nAxisNo, @nCtrlMode ) = cmERR_NONE then
    begin
        if nCtrlMode <> cmCTRL_SEMI_C then
        begin
            // Semi-Closed Loop 제어 모드로 설정
            cmmCfgSetCtrlMode ( nAxisNo, cmCTRL_OPEN );
        end;
    end;

end;
```

NAME

cmmCfgSetSeqMode
cmmCfgGetSeqMode
– 시퀀스(Sequence) 모드 설정(設定) 및 반환(返還)

INFORMATION

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

SYNOPSIS

❒ VT_I4 cmmCfgSetSeqMode

([in] VT_I4 SeqMode)

❒ VT_I4 cmmCfgGetSeqMode

([out] VT_PI4 SeqMode)

DESCRIPTION

cmmCfgSetSeqMode () 함수는 현재 이송이 진행되고 있는 축에 새로운 이송 명령이 하달되었을 때 이의 처리를 어떻게 할 것인지에 대한 모드를 설정하는 것입니다. 이러한 경우에 CMMSDK 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.

cmmCfgGetSeqMode () 함수는 각 축의 시퀀스 모드에 대하여 현재 장치에 설정되어 있는 값을 읽어 들이는 함수입니다.

PARAMETER

▶ SeqMode : cmmCfgSetSeqMode 함수의 인자이며, 현재 이송이 진행되고 있는 축에 새로운 이송 명령이 하달되었을 때 이의 처리를 어떻게 할 것인지에 대한 시퀀스(Sequence) 모드를 설정하는 것입니다. 이 값의 의미는 다음과 같습니다.

Value	Meaning
0 (cmSEQM_SKIP_RUN) **[초기값]**	현재 이송이 진행되고 있는 축에 새로운 이송 명령이 하달되면 -5170(cmERR_MOT_SEQ_SKIPPED) 에러값과 함께 곧바로 반환됩니다. 다양한 에러코드의 확인 본 매뉴얼의 부록편에 명시되어 있습니다.
1 (cmSEQM_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.

▶ SeqMode : cmmCfgGetSeqMode 함수의 인자이며, 시퀀스(Sequence) 모드를 반환합니다. 이 값의 의미는 다음과 같습니다.

Value	Meaning
0 (cmSEQM_SKIP_RUN) **[초기값]**	현재 이송이 진행되고 있는 축에 새로운 이송 명령이 하달되면 -5170(cmERR_MOT_SEQ_SKIPPED) 에러값과 함께 곧바로 반환하는 모드입니다.
1 (cmSEQM_WAIT_RUN)	현재 이송이 진행되고 있는 축에 새로운 이송 명령이 하달되면 이송 함수 내부에서 루프를 돌면서 이전 이송이 완료되기를 기다리다가 이송이 완료되면 현재 하달된 이송 명령을 수행하는 모드입니다.

RETURN VALUE

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

REFERENCE

시퀀스(Sequence) 모드가 cmSEQM_SKIP_RUN 으로 설정된 경우[2]에 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 "Cmmsdk.h"
#include "CmmsdkDef.h"

void OnSetSeqMode ()
{
	long nSeqMode;			// 시퀀스 모드 정보.

	// 설정되어있는 시퀀스 모드를 확인 후, cmSEQM_SKIP_RUN 모드로 설정합니다.
	if (cmmCfgGetSeqMode ( &nSeqMode ) == cmERR_NONE )
	{
		if ( nSeqMode != cmSEQM_SKIP_RUN )
		{
			cmmCfgSetSeqMode ( cmSEQM_SKIP_RUN )
		}
	}
}


void OnMove ()
{
	SxVMoveStart (cmX1 ,1);			// 속도 이송 명령 수행

	if ( cmmSxMoveStart (cmX1, 100000 ) != cmERR_NONE )
	{
		// -5170 에러 코드 반환
		// 시퀀스 모드가 cmSEQM_SKIP_RUN 모드로 설정되어 있으므로
		// 이송 명령이 완료되지 않은 상태에서 다른 이송 명령은 실행되지 않습니다.
	}
}
```

Visual Basic

```
Private Sub OnSetSeqMode ()

	Dim nSeqMode As Long			' 시퀀스 모드 정보.

	' 설정되어있는 시퀀스 모드를 확인 후, cmSEQM_SKIP_RUN 모드로 설정합니다.
	If cmmCfgGetSeqMode ( nSeqMode ) = cmERR_NONE Then

		If nSeqMode <> cmSEQM_SKIP_RUN Then
		Call cmmCfgSetSeqMode ( cmSEQM_SKIP_RUN )
```

[2] 시퀀스(Sequence) 모드에 대한 초기(기본) 설정은 cmSEQM_SKIP_RUN 입니다.

```
        End If
End If

End Sub


Private Sub OnMove ()

        Call SxVMoveStart (cmX1 ,1)                     ' 속도 이송 명령 수행

        If cmmSxMoveStart (cmX1, 100000 ) <> cmERR_NONE Then
                ' -5170 에러 코드 반환
                ' 시퀀스 모드가 cmSEQM_SKIP_RUN 모드로 설정되어 있으므로
                ' 이송 명령이 완료되지 않은 상태에서 다른 이송 명령은 실행되지 않습니다.
        End If

End Sub
```

```
Delphi

procedure OnSetSeqMode ();
var
        nSeqMode : LongInt;                 // 시퀀스 모드 정보.

begin
        // 설정되어있는 시퀀스 모드를 확인 후, cmSEQM_SKIP_RUN 모드로 설정합니다.
        if cmmCfgGetSeqMode ( @nSeqMode ) = cmERR_NONE then
        begin
                if nSeqMode <> cmSEQM_SKIP_RUN then
                begin
                        cmmCfgSetSeqMode ( cmSEQM_SKIP_RUN );
                end;
        end;

end;


procedure OnMove ();
begin
        SxVMoveStart (cmX1,1 );                         ' 속도 이송 명령 수행

        if cmmSxMoveStart (cmX1, 100000 ) <> cmERR_NONE then
        begin
                { -5170 에러 코드 반환
                시퀀스 모드가 cmSEQM_SKIP_RUN 모드로 설정되어 있으므로
                이송 명령이 완료되지 않은 상태에서 다른 이송 명령은 실행되지 않습니다. }
        end;

end;
```

NAME

cmmCfgSetManExtLimit
cmmCfgGetManExtLimit
– 사용자 정의 한계(Limit) 신호 제어 및 설정(設定) 반환(返還)

INFORMATION

- Environment
  Configuration Functions
- VC++/VB
  BCB/Delphi/.NET
- Level 2
- LX504a 전용 기능

SYNOPSIS

❒ VT_I4 cmmCfgSetManExtLimit

([in] VT_I4 Axis, [in] VT_I4 IsSetELP, [in] VT_I4 IsEnable, [in] VT_I4 ManState)

❒ VT_I4 cmmCfgGetManExtLimit

([in] VT_I4 Axis, [in] VT_I4 IsGetELP, [out] VT_PI4 IsEnable, [out] VT_PI4 ManState)

DESCRIPTION

cmmCfgSetManExtLimit 함수는 -/+ EL(External Limit) 입력에 리밋 센서 입력을 무시하고 수동(소프트웨어적)으로 ON 또는 OFF 가 입력되도록 할 수 있는데 이 기능에 대한 설정을 하는 함수입니다. 단, 이 함수는 COMI-LX504a 제품에서만 적용 가능한 함수입니다. COMI-LX504a 이외의 제품에서는 이 기능을 지원하지 않습니다.

cmmCfgGetManExtLimit 함수는 수동 Limit 입력 기능에 대하여 현재 장치에 설정되어 있는 값을 읽어 들이는 함수입니다.

PARAMETER

▶ Axis: 축번호. 축번호는 상수값으로 [cmX1] 부터 0 번째 축을 기준 축으로 임의의 축을 설정할 수 있습니다.

▶ IsSetELP : 이 기능의 설정 대상을 결정하는 매개 변수(媒介變數)입니다. 이 기능은 (-)EL 과 (+)EL 에 따로 따로 적용할 수 있습니다.

Value	Meaning
0 (cmFALSE)	적용대상이 (-)EL 임을 의미합니다.
1 (cmTRUE)	적용대상이 (+)EL 임을 의미합니다.

▶ IsGetELP : 이 기능의 설정 대상을 결정하는 매개 변수(媒介變數)입니다. 이 기능은 (-)EL 과 (+)EL 에 따로 따로 적용할 수 있습니다.

Value	Meaning
0 (cmFALSE)	적용대상이 (-)EL 임을 의미합니다.
1 (cmTRUE)	적용대상이 (+)EL 임을 의미합니다.

▶ IsEnable : cmmCfgSetManExtLimit 함수의 인자이며, 수동 Limit 입력 기능을 사용할 것인지를 결정하는 매개 변수(媒介變數)입니다. 이 값의 의미는 다음과 같습니다.

Value	Meaning
0 (cmFALSE) **[초기값]**	수동 입력 기능을 사용하지 않습니다.
1 (cmTRUE)	수동 입력 기능을 사용합니다. 따라서 이 모드에서는 리밋 센서의 상태는 무시되며, ManState 매개 변수(媒介變數)의 값이 리밋 센서의 입력의 상태를 대신합니다.

▶ IsEnable : cmmCfgGetManExtLimit 함수의 인자이며, 수동 Limit 입력 기능의 사용여부를 반환합니다.

Value	Meaning
0 (cmFALSE) **[초기값]**	수동 입력 기능을 사용하지 않고 있습니다.
1 (cmTRUE)	수동 입력 기능을 사용하고 있습니다.

▶ ManState : cmmCfgSetManExtLimit 함수의 인자이며, 수동 리밋 입력 상태(OFF/ON)를 설정합니다. IsEnable 매개 변수(媒介變數)가 1 로 설정되어 있으면 리밋 센서의 상태는 무시되고 (–)EL 또는 (+)EL 입력의 상태는 항상 ManState 매개 변수(媒介變數)에서 지정한 상태로 인식됩니다. 단, IsEnable 매개 변수(媒介變數)가 0 이면 이 값은 무시됩니다.

Value	Meaning
0	(-)EL 또는 (+)EL 의 상태가 센서의 상태와 상관없이 OFF 로 인식됩니다.
1	(-)EL 또는 (+)EL 의 상태가 센서의 상태와 상관없이 ON 으로 인식됩니다.

▶ ManState : cmmCfgGetManExtLimit 함수의 인자이며, 수동 리밋 입력 상태(OFF/ON)를 반환합니다.

Value	Meaning
0	(-)EL 또는 (+)EL 의 상태가 센서의 상태와 상관없이 OFF 입니다.
1	(-)EL 또는 (+)EL 의 상태가 센서의 상태와 상관없이 ON 입니다.

RETURN VALUE

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

SEE ALSO

cmmStReadMioStatuses

REFERENCE

이 함수는 COMI-LX504a 제품에 대해서만 사용 가능한 함수입니다.

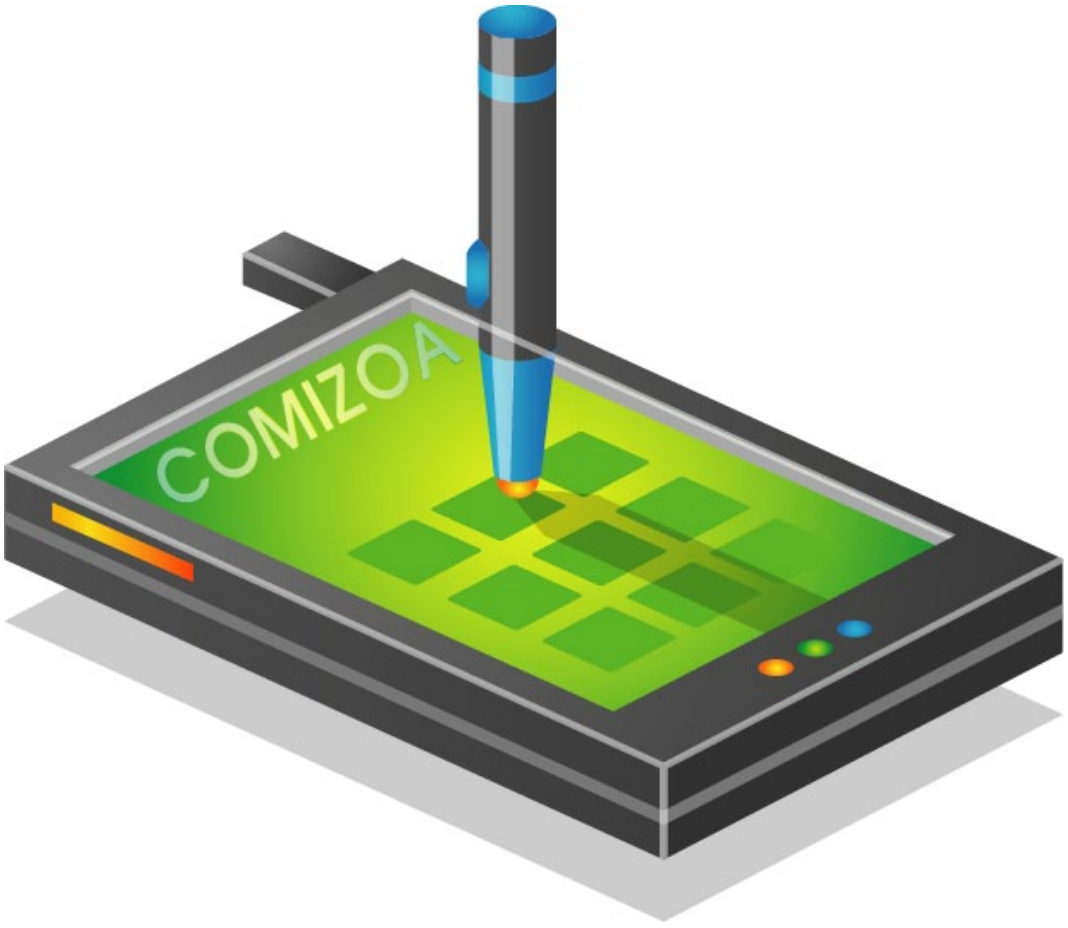
Chapter

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, 모션제어, 기본 보간 제어, 원점복귀(原點復歸)등의 내용으로 구성되어 있습니다. 단축 제어는 단일 축을 단일 축을 독립적으로 제어하는 작업을 의미합니다. 다축(多軸) 제어는 다수의 복수(複數) 축을 제어하는 제어하는 것을 의미하며, 보간제어는 직선 보간과 원호 보간(補間) 기능(機能)으로 구성되어 있습니다. 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 cmmSxSetSpeedRatio ([in] VT_I4 Axis, [in] VT_I4 SpeedMode, [in] VT_R8 VelRatio, [in] VT_R8 AccRatio, [in] VT_R8 DecRatio) 단축(單軸) 구동 시 해당 축에 대한 속도방식(速度方式) 및 속도 비율(速度比率)을 설정합니다.
❒ VT_I4 cmmSxGetSpeedRatio ([in] VT_I4 Axis, [out] VT_PI4 SpeedMode, [out] VT_PR8 VelRatio, [out] VT_PR8 AccRatio, [out] VT_PR8 DecRatio) 단축(單軸) 구동 시 해당 축에 대한 설정된 속도방식(速度方式) 및 속도 비율(速度比率)을 반환합니다.
❒ VT_I4 cmmSxMoveStart ([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 cmmSxMove ([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 cmmSxMoveToStart ([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 cmmSxMoveTo ([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 cmmSxVMoveStart ([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 cmmSxStop ([in] VT_I4 Axis, [in] VT_I4 IsWaitComplete, [in] VT_I4 IsBlocking) 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 cmmSxStopEmg ([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 cmmSxIsDone ([in] VT_I4 Axis, [out] VT_PI4 IsDone) 단축(單軸) 이송의 완료(完了) 를 확인(確認)합니다.
❒ VT_I4 cmmSxWaitDone ([in] VT_I4 Axis, [in] VT_I4 IsBlocking) 단축(單軸) 이송의 완료(完了) 시점까지 대기(待機)합니다.
❒ VT_I4 cmmSxGetTargetPos([in] VT_I4 Channel, [out] VT_PR8 Position) 단축(單軸) 구동 시 대상 축에 대한 이송 목표 위치(상대 혹은 절대 좌표)를 반환합니다.
❒ VT_I4 cmmSxOptSetIniSpeed ([in] VT_I4 Axis, [in] VT_R8 IniSpeed) 단축(單軸) 모션의 초기속도(初期 速度)를 설정(設定) 합니다.
❒ VT_I4 cmmSxOptGetIniSpeed ([in] VT_I4 Axis, [out] VT_PR8 IniSpeed) 단축(單軸) 모션의 초기속도(初期 速度)의 설정(設定)을 반환(返還)합니다.
❒ VT_I4 cmmSxSetCorrection ([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 cmmSxGetCorrection ([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.

❒ VT_I4 cmmSxOptSetSyncMode ([in] VT_I4 Axis, [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 cmmSxOptGetSyncMode ([in] VT_I4 Axis, [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 cmmSxOptSetSyncOut ([in] VT_I4 Axis, [in] VT_I4 Mode, [in] VT_I4 DoChan_local, [in] VT_I4 DoLogic) 모션 속도 구간 별 디지털 출력(出力)을 설정합니다. 지정한 축의 속도구간(速度區間)에서 머신비젼(Machine Vision) 등의 동기 시작 신호(信號)로 사용될 수 있습니다.
❒ VT_I4 cmmSxOptGetSyncOut ([in] VT_I4 Axis, [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 속도 구간에서 머신비젼(Machine Vision) 등의 동기 시작 신호로 사용 되는 설정에 대해 반환합니다.
❒ VT_I4 cmmSxOptSetRdpOffset ([in] VT_I4 Axis, [in] VT_R8 OffsetDist) 감속시작의 상대 (相對) 위치를 설정합니다. RDP(Ramping Down Point) 의 오프셋(Offset) 에 대해 설정합니다.
❒ VT_I4 cmmSxOptGetRdpOffset ([in] VT_I4 Axis, [out] VT_PR8 OffsetDist) 감속시작의 상대 (相對) 위치를 반환합니다. RDP(Ramping Down Point) 의 오프셋(Offset) 을 반환합니다.

8.1.2 함수 설명

NAME

cmmSxSetSpeedRatio
cmmSxGetSpeedRatio
– 단축(單軸)이송 속도 비율 지정

INFORMATION

- Single Axis Control
- VC++/VB
- BCB/Delphi/.NET
- Level 3
- ☺ 다소 주의

SYNOPSIS

❒ VT_I4 cmmSxSetSpeedRatio

([in] VT_I4 Axis, [in] VT_I4 SpeedMode, [in] VT_R8 VelRatio, [in] VT_R8 AccRatio, [in] VT_R8 DecRatio)

❒ VT_I4 cmmSxGetSpeedRatio

([in] VT_I4 Axis, [out] VT_PI4 SpeedMode, [out] VT_PR8 VelRatio, [out] VT_PR8 AccRatio, [out] VT_PR8 DecRatio)

DESCRIPTION

단축 구동시 해당 축에 대한 속도 모드 및 속도 비율을 설정하거나 설정된 값을 반환합니다.

PARAMETER

▶ Axis: 축번호. 축번호는 상수값으로 [cmX1] 부터 0 번째 축을 기준 축으로 임의의 축을 설정할 수 있습니다.

▶ SpeedMode : cmmSxSetSpeedRatio 함수의 인자이며, 속도모드의 설정값입니다. 다음과 같은 설정값을 가집니다.

Value	Meaning
0 또는 cmSMODE_C	CONSTANT 속도모드 => 가감속을 수행하지 않습니다.
1 또는 cmSMODE_T	TRAPEZOIDAL 속도모드 => 사다리꼴 가감속을 수행합니다.
2 또는 cmSMODE_S	S-CURVE 속도모드 => S-CURVE 가감속을 수행합니다.
-1 또는 cmSMODE_KEEP	이전 속도 모드를 그대로 유지합니다. 즉, 속도모드를 변경하지 않습니다.

▶ SpeedMode : cmmSxGetSpeedRatio 함수의 인자이며, 속도모드의 설정값입니다. 다음과 같은 설정값을 가집니다.

Value	Meaning
0 또는 cmSMODE_C	CONSTANT 속도모드 => 가감속을 수행하지 않습니다.
1 또는 cmSMODE_T	TRAPEZOIDAL 속도모드 => 사다리꼴 가감속을 수행합니다.
2 또는 cmSMODE_S	S-CURVE 속도모드 => S-CURVE 가감속을 수행합니다.

▶ VelRatio : cmmSxSetSpeedRatio 함수의 인자이며, 작업 속도 비율(Ratio) 값입니다. 이 값의 단위는 %입니다.

▶ VelRatio : cmmSxGetSpeedRatio 함수의 인자이며, 작업 속도 비율(Ratio)을 반환합니다. 이 값의 단위는 %입니다.

▶ AccRatio : cmmSxSetSpeedRatio 함수의 인자이며, 가속도 비율(Ratio) 값입니다. 이 값의 단위는 %입니다.

▶ AccRatio : cmmSxGetSpeedRatio 함수의 인자이며, 가속도 비율(Ratio)을 반환합니다. 이 값의 단위는 %입니다.

▶ DecRatio : cmmSxSetSpeedRatio 함수의 인자이며, 감속도 비율(Ratio) 값입니다. 이 값의 단위는 %입니다.

▶ DecRatio : cmmSxGetSpeedRatio 함수의 인자이며, 감속도 비율(Ratio) 을 반환합니다. 이 값의 단위는 %입니다.

RETURN VALUE

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

속도 비율(Ratio) 의 정확한 의미를 알고 싶습니다.

속도 설정은 비율로 설정이 됩니다. 비율에 의한 속도값은 기준 값에 배수(Multiplication)가 되거나 제법(Division) 이 된 속도 값을 의미합니다. 기준이 되는 속도 값은 cmmCfgSetSpeedPattern 으로 설정됩니다.

㈜ 커미조아 CMMSDK 에서는 기준 속도(Standard Speed) 개념을 이용하고 있습니다. 전체 모션 속도는 기준속도의 비율로 설정이 가능하며, 이것은 cmmCfgSetSpeedPattern 함수에 의해서 설정된 기준 속도를 의미합니다. ㈜ 커미조아의 CMMSDK 는 기준속도의 값을 비율(Ratio) 로 설정할 수 있는 커다란 이점을 가지고 있습니다.

SEE ALSO

cmmCfgSetSpeedPattern, cmmCfgGetSpeedPattern

EXAMPLE

C/C++

```
#include "Cmmsdk.h"
#include "CmmsdkDef.h"

void OnSetSpeed ()
{
	long nAxisNo = 1;	// 모션 이송 속도를 설정할 대상 축을 선택합니다.

	/* 단축 모션의 작업 속도 비율을 설정합니다. cmmCfgSetSpeedPattern ()함수를 통해서
	설정된 모션 이송 속도를 기준으로 속도 비율이 적용됩니다. */

	cmmCfgSetSpeedPattern ( nAxisNo, cmSMODE_T, 1000, 10000, 10000 );

	cmmSxSetSpeedRatio (	nAxisNo,		// 대상 축 선택
				cmSMODE_T,		// 속도 모드 선택
				50,			// 작업 속도 비율. 1000 * 0.5 = 500 pps
				80,			// 가속도 비율. 10000 * 0.8 = 8000 pps²
				80			// 감속도 비율. 10000 * 0.8 = 8000 pps²
				);
}
```

Visual Basic

```
Private Sub OnSetSpeed (void)

	Dim nAxisNo As Long

	nAxisNo = 1		' 모션 이송 속도를 설정할 대상 축을 선택합니다.

	' 단축 모션의 작업 속도 비율을 설정합니다. cmmCfgSetSpeedPattern ()함수를 통해서
	' 설정된 모션 이송 속도를 기준으로 속도 비율이 적용됩니다.

	Call cmmCfgSetSpeedPattern ( nAxisNo, cmSMODE_T, 1000, 10000, 10000 )
```

```
        Call cmmSxSetSpeedRatio( nAxisNo,cmSMODE_T,50,80,80 )

End Sub
```

```
Delphi

procedure OnSetSpeed ();
var
        nAxisNo : LongInt;
begin
        nAxisNo := 1;           // 모션 이송 속도를 설정할 대상 축을 선택합니다.

        // 단축 모션의 작업 속도 비율을 설정합니다. cmmCfgSetSpeedPattern ()함수를 통해서
        // 설정된 모션 이송 속도를 기준으로 속도 비율이 적용됩니다.

        cmmCfgSetSpeedPattern ( nAxisNo, cmSMODE_T, 1000, 10000, 10000 );

        cmmSxSetSpeedRatio ( nAxisNo,  cmSMODE_T, 50, 80, 80);

end;
```

NAME

cmmSxMove
cmmSxMoveStart
– 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 cmmSxMove ([in] VT_I4 Axis, [in] VT_R8 Distance, [in] VT_I4 IsBlocking)

❒ VT_I4 cmmSxMoveStart ([in] VT_I4 Axis, [in] VT_R8 Distance)

DESCRIPTION

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

PARAMETER

▶ Axis: 축번호. 축번호는 상수값으로 [cmX1] 부터 0 번째 축을 기준 축으로 임의의 축을 설정할 수 있습니다.

▶ 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
cmFALSE	블록(Blocking)을 하지 않습니다. 따라서 해당 모션이 완료되는 동안에도 윈도우 이벤트를 처리합니다.
cmTRUE	블록(Blocking)을 합니다. 따라서 해당 모션이 완료되는 동안에는 윈도우 이벤트가 처리되지 않습니다.

SEE ALSO

cmmSxMoveTo, cmmSxMoveToStart

REFERENCE

□ 논리적 거리 단위는 cmmCfgSetUnitDist() 함수에 의해 결정됩니다.

□cmmSxMoveStart 함수를 사용하는 경우에는 cmmSxIsDone() 함수나 cmmSxWaitDone() 함수를 사용하여 모션의 완료를 확인(確認)할 수 있습니다.

□ cmmSxMove 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.

RETURN VALUE

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

EXAMPLE

C/C++

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

#include "Cmmsdk.h"
#include "CmmsdkDef.h"

/*****************************************************************
* OnProgramInitial : 이 함수는 가상의 함수로서 프로그램 초기화 루틴이
* 적용되는 부분을 의미합니다.
*****************************************************************/
void OnProgramInitial()
{
        long m_nNumAxes;
        cmmLoadDll();
        if(cmmGnDeviceLoad(cmTRUE, &m_nNumAxes) != cmERR_NONE)
        {
                //Handle 은 사용자가 생성한 폼의 핸들 값입니다.
                cmmErrShowLast(Handle);
                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.
        cmmCfgSetSpeedPattern(cmX1, cmSMODE_S, m_fVwork, m_fAcc, m_fDec );
}

/************************************************************
* DoMotion : 작업명령시에 호출되는 가상의 함수 입니다.
************************************************************/
void DoMotion()
{
        //cmmSetCfgSetSpeedPattern 으로 설정된 첫번째 축(Axis)의 속도모드를
        //그대로 유지 하면서 cmmCfgSetSpeedPattern 에서 설정된 작업속도,
        //가속도, 감속도의 비율로 지정한 90%로 설정합니다.
        cmmSxSetSpeedRatio(cmX1, cmSMODE_KEEP, 90, 90, 90);

        //블록 모드를 FALSE 로 하여서 UI 메시지(Message) 처리가 가능합니다.
        cmmSxMove(cmX1, 5000, cmFALSE);//Move 5000
        if(!m_bAbortMotion) //Stop 버튼이 눌리지 않았는지 확인(確認)
                cmmSxMove(cmX1, -5000, cmFALSE);//Move -5000
}
```

Visual Basic

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

Private Sub Form_Load()
   Dim nTotalAxis As Long
   Dim IRetVal As Long
 '================================================================
 ' cmmGnDeviceLoad 함수로 장치를 초기화합니다.
  IRetVal = cmmGnDeviceLoad(True, nTotalAxis)

  If IRetVal <> cmERR_NONE Then
     MsgBox ("cmmGnDeviceLoad has been failed")
  End If
 '================================================================
End Sub

Private Sub CfgSpeed(nTotalAxis As Long)

Dim i As Integer
 '=======================================================
   ' 이 함수에서 cmmCfgSetSpeedPattern 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 cmmCfgSetSpeedPattern(i, cmSMODE_S, 1000, 2000, 2000)
   Next
End Sub

Private Sub btnLeft_Click()
   Dim nResult As Long

   ' 이 함수로 인해 설정된 기준속도의 비율(%) 로 모션 동작을 수행합니다.
   Call cmmSxSetSpeedRatio(cmX1, cmSMODE_S, 100#, 100#, 100#)

   ' 왼쪽 버튼을 누르게 되면, 입력된 거리의 방향으로 이송합니다.
   nResult = cmmSxMove(cmX1, 5000, cmFALSE)
End Sub

Private Sub btnRight_Click()
 Dim nResult As Long

   ' 이 함수로 인해 설정된 기준속도의 비율(%) 로 모션 동작을 수행합니다.
   Call cmmSxSetSpeedRatio(cmX1, cmSMODE_S, 100#, 100#, 100#)
```

```
   ' 오른쪽 버튼을 누르게 되면, 입력된 거리의 반대 방향으로 이송합니다.
   nResult = cmmSxMove(cmX1, -5000, cmFALSE)
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;
begin

        // Load CMMSDK(DLL) Library
        if ( cmmGnDeviceLoad(cmTRUE,@g_nAxis) <> cmERR_NONE ) then
        begin
                // 마지막에 발생한 에러를 화면에 표시합니다.
                // 함수 인자로는 Form 의 Handle 이 전달됩니다.
                cmmErrShowLast(Handle);
                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 := cmSMODE_S;
      // 설정된 기준 속도를 실제 SDK 함수에 전달합니다.

      cmmCfgSetSpeedPattern(
      cmX1,             // 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.

end;


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

procedure btnPositiveClick();
var
    fWorkSpeedRatio : Double;
    fAccelSpeedRatio : Double;
    fDecelSpeedRatio : Double;
begin
        ////////////////////////////////////////////////////////////
        // 저희 COMIZOA 에서는 다음과 같은 형태의 함수를 제공합니다.
        // 가) 지령 펄스 출력과 위치 검출기를 통한 펄스 값 입력의 일치를 위한 제어
        // 설명 : 위 함수는 지정된 지령 위치의 펄스 출력을 내보내고, 위치 검출기
        // 를 통해 지령된 위치까지 이송 후에 함수가 반환됩니다.
        // cmmSxMove [상대좌표]
        // cmmSxMoveTo [절대좌표]
```

```
        // 나) 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
        //                      cmmSxIsDone 혹은 cmmSxWaitDone 함수가 있습니다.
        // cmmSxMoveStart [상대좌표]
        // cmmSxMoveToStart [절대좌표]
        ////////////////////////////////////////////////////////////

    // 기준속도 대비 실제 구동속도를 비율로  구합니다.
    // 여기서 기준 속도란 cmmCfgSetSpeedPattern 함수를 통해 설정된
          // 속도를 의미하며, 아래의 cmmSxSetSpeedRatio 함수는
    // 단축(Single Axis)를 대상으로 축의 속도를 기준 속도 대비
          // Percent(%) 단위로 입력 받아 설정하는 함수입니다.
    fAccelSpeedRatio := 100;
    fDecelSpeedRatio := 100;
    fWorkSpeedRatio := 100;

    cmmSxSetSpeedRatio(
                        cmX1,     //현재 활성화 축을 알려주는 가상함수
       cmSMODE_KEEP,     // 이전에 설정된 가감속 모드를 그대로 사용합니다.
       fWorkSpeedRatio,
       fAccelSpeedRatio,
       fDecelSpeedRatio );

          cmmSxMove
          (
          cmX1,
          5000, // (-)일 경우 반대 방향으로 이동합니다.
          cmFALSE
          );
end;
```

NAME

cmmSxMoveTo
cmmSxMoveToStart
- 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.cmmSxMoveTo ([in] VT_I4 Axis, [in] VT_R8 Position, [in] VT_I4 IsBlocking)

❒ VT_I4 cmmSxMoveToStart ([in] VT_I4 Axis, [in] VT_R8 Position)

DESCRIPTION

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

PARAMETER

▶ Axis: 축번호. 축번호는 상수값으로 [cmX1] 부터 0 번째 축을 기준 축으로 임의의 축을 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 : 완료될 때까지 기다리는 동안 윈도우 메시지를 블록(Blocking)할 것인지를 결정합니다. 단, 쓰레드내에서 실행할 때는 이 값을 1(cmTRUE)로 설정해주어야 합니다.

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

SEE ALSO

cmmSxMove, cmmSxMoveStart

REFERENCE

□ 논리적 거리 단위는 cmmCfgSetUnitDist() 함수에 의해 결정됩니다.

□ cmmSxMoveToStart() 함수를 사용하는 경우에는 cmmSxIsDone() 함수나 cmmSxWaitDone() 함수를 사용하여 모션의 완료를 확인(確認)할 수 있습니다.

□ cmmSxMoveTo() 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.

RETURN VALUE

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

EXAMPLE

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

```
C/C++
#include "Cmmsdk.h"
#include "CmmsdkDef.h"

/****************************************************************
* OnProgramInitial : 이 함수는 가상의 함수로서 프로그램 초기화 루틴이
* 적용되는 부분을 의미합니다.
****************************************************************/

void OnProgramInitial()
{
	long m_nNumAxes;
	cmmLoadDll();
	if(cmmGnDeviceLoad(cmTRUE, &m_nNumAxes) != cmERR_NONE)
	{
		//Handle 은 사용자가 생성한 폼의 핸들 값입니다.
		cmmErrShowLast(Handle);
		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.
	cmmCfgSetSpeedPattern(cmX1, cmSMODE_S, m_fVwork, m_fAcc,  m_fDec );
}

/************************************************************
* DoMotion : 작업명령시에 호출되는 가상의 함수 입니다.
*************************************************************/
void DoMotion()
{
	if(cmmSxMoveTo(cmX1, 1000.0, cmFALSE) != cmERR_NONE){
		cmmErrShowLast(Handle);
		return ;
	}
	if(cmmSxMoveTo(cmX1, 0, cmFALSE) != cmERR_NONE){
		cmmErrShowLast(Handle);
		return ;
	}

	//cmmSxMoveTo()함수 대신에 cmmSxMoveToStart() 함수를 사용하려면
	//아래와 같이 코드를 작성한다.
	//if(cmmSxMoveToStart(cmX1, 1000.0) != cmERR_NONE){
	//		cmmErrShowLast(Handle);
	//		return ;
	//}
	//if(cmmSxWaitDone(cmX1, cmFALSE) != cmERR_NONE){
	//		cmmErrShowLast(Handle);
	//		return ;
	//}
	//
	//if(cmmSxMoveToStart(cmX1, 0.0) != cmERR_NONE){
	//		cmmErrShowLast(Handle);
	//		return ;
	//}
	//if(cmmSxWaitDone(cmX1, cmFALSE) != cmERR_NONE){
	//		cmmErrShowLast(Handle);
	//		return ;
	//}
}
```

```
Visual Basic

'=============================================================
'cmmGnDeviceLoad 함수로 장치를 초기화 합니다.
'=============================================================
Private Sub Form_Load()

   Dim nTotalAxis As Long
   Dim IRetVal As Long
   Dim Hwnd As Long

'=============================================================
 ' cmmGnDeviceLoad 함수로 장치를 초기화합니다.
 IRetVal = cmmGnDeviceLoad(True, nTotalAxis)

 If IRetVal <> cmERR_NONE Then
    MsgBox ("cmmGnDeviceLoad has been failed")
 End If
  '============================================================

End Sub

Private Sub CfgSpeed(nTotalAxis As Long)
   Dim i As Integer
   '============================================================
   ' 이 함수에서 cmmCfgSetSpeedPattern 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 cmmCfgSetSpeedPattern(cmX1, cmSMODE_S, 1000, 2000,  2000)
  Next
End Sub

Private Sub btnLeft_Click()
  Dim nResult As Long

  ' 이 함수로 인해 설정된 기준속도의 비율(%) 로 모션 동작을 수행합니다.
  Call cmmSxSetSpeedRatio(cmX1, cmSMODE_S, 100#, 100#, 100#)

  ' 왼쪽 버튼을 누르게 되면, 입력된 거리의 반대 방향으로 이송합니다.
  nResult = cmmSxMoveTo( cmX1, 1000, cmFALSE)
End Sub

Private Sub btnRight_Click()
  Dim nResult As Long

  ' 이 함수로 인해 설정된 기준속도의 비율(%) 로 모션 동작을 수행합니다.
  Call cmmSxSetSpeedRatio(cmX1, cmSMODE_S, 100#, 100#, 100#)

  ' 오른쪽 버튼을 누르게 되면, 입력된 거리의 정 방향으로 이송합니다.
  nResult = cmmSxMoveTo(cmX1, 0, cmFALSE)
End Sub
```

Delphi

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

procedure OnCreate();
var
        g_nAxis : LongInt;
begin
        // Load CMMSDK(DLL) Library
        if ( cmmGnDeviceLoad(cmTRUE,@g_nAxis) <> cmERR_NONE ) then
        begin
                // 마지막에 발생한 에러를 화면에 표시합니다.
                // 함수 인자로는 Form 의 Handle 이 전달됩니다.
                cmmErrShowLast(Handle);
                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 := cmSMODE_S;
         // 설정된 기준 속도를 실제 SDK 함수에 전달합니다.

       cmmCfgSetSpeedPattern(
       cmX1,              // 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.

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.
        // 가) 지령 펄스 출력과 위치 검출기를 통한 펄스 값 입력의 일치를 위한 제어
        // 설명 : 위 함수는 지정된 지령 위치의 펄스 출력을 내보내고,
        // 위치 검출기를 통해 지령된 위치까지 이송 후에 함수가 반환됩니다.
        // cmmSxMove [상대좌표]
        // cmmSxMoveTo [절대좌표]

        // 나) 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
        // cmmSxIsDone 혹은 cmmSxWaitDone 함수가 있습니다.

        // cmmSxMoveStart [상대좌표]
        // cmmSxMoveToStart [절대좌표]
        ////////////////////////////////////////////////////////////

    //cmmSxSetSpeedRation 를 통하여서 기준 속도와의 비에 따른 실제 구동
    // 속도를 설정 합니다.
    // 여기서 기준 속도란 cmmCfgSetSpeedPattern 함수를 통해 설정된
    // 속도를 의미하며, 아래의 cmmSxSetSpeedRatio 함수는
    // 단축(Single Axis)를 대상으로 축의 속도를
    // 기준 속도 대비 Percent(%) 단위로 입력 받아 설정하는 함수입니다.
   fAccelSpeedRatio := 100;
   fDecelSpeedRatio := 100;
   fWorkSpeedRatio := 100;

   cmmSxSetSpeedRatio(cmX1, cmSMODE_KEEP, fWorkSpeedRatio, fAccelSpeedRatio, fDecelSpeedRatio);

   cmmSxMoveTo(cmX1, 1000, cmFALSE );
   cmmSxMoveTo(cmX1, 0, cmFALSE );

end;
```

NAME

cmmSxVMoveStart
– 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 cmmSxVMoveStart ([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.

PARAMETER

▶ Axis: 축번호. 축번호는 상수값으로 [cmX1] 부터 0 번째 축을 기준 축으로 임의의 축을 설정할 수 있습니다.

▶ 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
cmERR_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 "Cmmsdk.h"
#include "CmmsdkDef.h"

/*************************************************************
* onprograminitial : 이 함수는 가상의 함수로서 프로그램 초기화 루틴이
* 적용되는 부분을 의미합니다.
*************************************************************/
void OnProgramInitial()
{
```

```
        long m_nNumAxes;
        cmmLoadDll();
        if(cmmGnDeviceLoad(cmTRUE, &m_nNumAxes) != cmERR_NONE)
        {
                //Handle 은 사용자가 생성한 폼의 핸들 값입니다.
                cmmErrShowLast(Handle);
                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.
        cmmCfgSetSpeedPattern(cmX1, cmSMODE_S, m_fVwork, m_fAcc, m_fDec );
}

/************************************************************
* OnPlusButtonDown() : (+)Move 버튼이 눌렸을 때 호출되는 가상의 함수
* 이 함수에서 (+)방향으로 V-Move 를 시작합니다.
************************************************************/
void OnPlusButtonDown ()
{
        cmmSxVMoveStart(cmX1, cmDIR_P);//Positive dir V-MOVE
}

/************************************************************
* OnPlusButtonUp() : (-)Move 버튼이 올라올 때 호출되는 가상의 함수
* 이 함수에서는 V-Move 를 종료합니다.
************************************************************/
void OnPlusButtonUp ()
{
        cmmSxStop(cmX1, cmFALSE, cmFALSE);
}

/************************************************************
* OnMinusButtonDown(): (-)Move 버튼이 눌렸을 때 호출되는 가상의 함수
* 이 함수에서 (+)방향으로 V-Move 를 시작합니다.
************************************************************/
void OnMinusButtonDown()
{
        cmmSxVMoveStart(cmX1, cmDIR_N); // Negative dir V-MOVE
}
/************************************************************
* OnMinusButtonUp() : (-)Move 버튼이 올라올 때 호출되는 가상의 함수
* 이 함수에서는 V-Move 를 종료합니다.
************************************************************/
void OnMinusButtonUp()
{
        cmmSxStop(cmX1, cmFALSE, cmFALSE);
}
```

Visual Basic

```
'============================================================
'cmmGnDeviceLoad 함수로 장치를 초기화 합니다.
'============================================================
Private Sub Form_Load()
   Dim nTotalAxis As Long
   Dim IRetVal As Long
   Dim Hwnd As Long
'============================================================
 ' cmmGnDeviceLoad 함수로 장치를 초기화합니다.
  IRetVal = cmmGnDeviceLoad(True, nTotalAxis)

  If IRetVal <> cmERR_NONE Then
     MsgBox ("cmmGnDeviceLoad has been failed")
```

```
  End If
  '=============================================================
End Sub

Private Sub CfgSpeed(nTotalAxis As Long)
  ‘=============================================================
  ‘ 이 함수에서 cmmCfgSetSpeedPattern 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 cmmCfgSetSpeedPattern(cmX1, cmSMODE_S, 1000, 2000, 2000)

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 cmmSxVMoveStart(cmX1, cmDIR_N)

End Sub
```

Delphi

```
// * 이 함수는 폼이 생성될때 이벤트에 의해 불려지며 , 장치를 로드하는 함수입
// * 니다.
procedure OnCreate();
var
      g_nAxis : LongInt;
begin

      // Load CMMSDK(DLL) Library
      if ( cmmGnDeviceLoad(cmTRUE,@g_nAxis) <> cmERR_NONE ) then
      begin
            // 마지막에 발생한 에러를 화면에 표시합니다.
            // 함수 인자로는 Form 의 Handle 이 전달됩니다.
            cmmErrShowLast(Form1.Handle);
            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 := cmSMODE_S;
        // 설정된 기준 속도를 실제 SDK 함수에 전달합니다.

    cmmCfgSetSpeedPattern(
            cmX1,          // 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.

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.
        // 가) 지령 펄스 출력과 위치 검출기를 통한 펄스 값 입력의 일치를 위한 제어
        // 설명 : 위 함수는 지정된 지령 위치의 펄스 출력을 내보내고,
        // 위치 검출기를 통해 지령된 위치까지 이송 후에 함수가 반환됩니다.
        // cmmSxMove [상대좌표]
        // cmmSxMoveTo [절대좌표]
        // 나) 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
        //        cmmSxIsDone 혹은 cmmSxWaitDone 함수가 있습니다.
        // cmmSxMoveStart [상대좌표]
        // cmmSxMoveToStart [절대좌표]
        ////////////////////////////////////////////////////////////

         //cmmSxSetSpeedRation 를 통하여서 기준 속도와의 비에 따른 실제 구동
         // 속도를 설정 합니다.
         // 여기서 기준 속도란 cmmCfgSetSpeedPattern 함수를 통해 설정된
         // 속도를 의미하며, 아래의 cmmSxSetSpeedRatio 함수는
         // 단축(Single Axis)를 대상으로 축의 속도를
         // 기준 속도 대비 Percent(%) 단위로 입력 받아 설정하는 함수입니다.
   fAccelSpeedRatio := 100;
   fDecelSpeedRatio := 100;
   fWorkSpeedRatio := 100;

   cmmSxSetSpeedRatio(cmX1 ,cmSMODE_KEEP, fWorkSpeedRatio,  fAccelSpeedRatio, fDecelSpeedRatio);

   cmmSxVMoveStart
   (
      cmX1,
      cmDIR_N
   );

end;
```

NAME

cmmSxStop
cmmSxStopEmg
– 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 cmmSxStop ([in] VT_I4 Axis, [in] VT_I4 IsWaitComplete, [in] VT_I4 IsBlocking)

❒ VT_I4 cmmSxStopEmg ([in] VT_I4 Axis)

DESCRIPTION

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

PARAMETER

▶ Axis: 축번호. 축번호는 상수값으로 [cmX1] 부터 0 번째 축을 기준 축으로 임의의 축을 설정할 수 있습니다.

▶ IsWaitComplete : 이 동작이 완료될때까지 함수를 반환할 것인지를 결정합니다.

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

Value	Meaning
cmFALSE	블록(Blocking)을 하지 않습니다. 따라서 해당 모션이 완료되는 동안에도 윈도우 이벤트를 처리합니다.
cmTRUE	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
cmERR_NONE	수행 성공

EXAMPLE

```
//* cmmSxVMoveStart 예제를 참고하여 주시기 바랍니다.
```

NAME

cmmSxIsDone
– 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 cmmSxIsDone ([in] VT_I4 Axis, [out] VT_PI4 IsDone)

DESCRIPTION

단일 축에 대하여 모션 완료를 확인(確認)합니다.

PARAMETER

▶ Axis: 축번호. 축번호는 상수값으로 [cmX1] 부터 0 번째 축을 기준 축으로 임의의 축을 설정할 수 있습니다.

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

RETURN VALUE

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

SEE ALSO

cmmSxWaitDone

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 nIsDone=0;
cmmSxMoveStart(cmX1, 1000);

while (1){
        cmmSxIsDone(cmX1, &nIsDone);
        if(nIsDone == cmTRUE) break;
        else{
                ...
        }
}
```

Visual Basic

```
cmmSxMoveStart(cmX1, 1000, cmFALSE)

Do Until IsDone
   ' 지정된 2 축에 대한 모션 완료 여부를 판단합니다.
   Call cmmSxIsDone(cmX1, IsDone)
...
```

Delphi

```
  // IsDone 은 모션 완료를 검사하기 위한 가상의 변수 입니다.
cmmSxMoveStart(cmX1, 1000, cmFALSE);

While (cmTRUE) do
Begin
        cmmSxIsDone( cmX1, @IsDone);
        if ( IsDone = cmTrue ) then break;
...
end;
```

NAME

cmmSxWaitDone
– 단축(單軸) 모션 완료대기(完了待機)

INFORMATION

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

SYNOPSIS

❒ VT_I4 cmmSxWaitDone ([in] VT_I4 Axis, [in] VT_I4 IsBlocking)

DESCRIPTION

단일 축에 대하여 모션이 완료(完了)될 때까지 기다립니다.

PARAMETER

▶ Axis: 축번호. 축번호는 상수값으로 [cmX1] 부터 0 번째 축을 기준 축으로 임의의 축을 설정할 수 있습니다.

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

Value	Meaning
cmFALSE	블록(Blocking)을 하지 않습니다. 따라서 해당 모션이 완료되는 동안에도 윈도우 이벤트를 처리합니다.
cmTRUE	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
cmERR_NONE	수행 성공

SEE ALSO

cmmSxIsDone

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++

```
if(cmmSxMoveStart(cmX1, 5000.0) != cmERR_NONE){
        cmmErrShowLast(Handle);
        return ;
}
//모션이 완료 될 때 까지 기다립니다.
if(cmmSxWaitDone(cmX1, cmFALSE) != cmERR_NONE){
        cmmErrShowLast(Handle);
        return ;
}
```

Visual Basic

```
If(cmmSxMoveStart(cmX1, 1000) <> cmERR_NONE) Then
        Call cmmErrShowLast(Handle) '// 핸들은 사용자 윈도우의 핸들입니다.
        Exit Sub
End If
'Wait till motion done
If(cmmSxWaitDone(cmX1, cmFALSE) <> cmERR_NONE) Then
        Call cmmErrShowLast(Handle)
        Exit Sub
End If
```

Delphi

```
if(cmmSxMoveStart(cmX1, 1000) <> cmERR_NONE) then begin
        cmmErrShowLast(Handle);
        exit
end;
//Wait till motion done //
if(cmmSxWaitDone(cmX1, cmFALSE) <> cmERR_NONE) then begin
        cmmErrShowLast(Handle);
        exit;
end;
```

NAME

cmmSxGetTargetPos
– 대상 축에 대한 이송 목표 좌표 확인

INFORMATION

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

SYNOPSIS

❒ VT_I4 cmmSxGetTargetPos([in] VT_I4 Channel, [out] VT_PR8 Position)

DESCRIPTION

대상 축에 대한 이송 목표 위치(상대 혹은 절대 좌표)를 반환합니다.

PARAMETER

▶ 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 : 이 매개변수를 통하여 대상 축에 설정된 이송 목표 위치(상대 혹은 절대 좌표)를 반환 합니다.

RETURN VALUE

Value	Meaning
음수	수행 실패. 자세한 내용은 '에러처리' 편을 참고합니다.
0(cmERR_NONE)	수행 성공.

EXAMPLE

C/C++

```
#include "Cmmsdk.h"
#include "CmmsdkDef.h"

void OnGetSxTargetPos ()
{
	double fTargetPos;  // 이송 명령의 목표 위치 정보.
	double fGetPos;     // 현재 위치 정보.

	if (cmmSxGetTargetPos ( cmX1, &fTargetPos ) == cmERR_NONE )
	{
		cmmStGetPosition ( cmX1, cmCNT_COMM, &fGetPos );
		if ( fTargetPos == fGetPos )
		{
			// 단축 이송 명령에 대한 이송 완료 확인.
		}
	}
}
```

Visual Basic

```
Private Sub OnGetSxTargetPos ()

        Dim fTargetPos As Double            '이송 명령의 목표 위치 정보.
        Dim fGetPos As Double               ' 현재 위치 정보.

        If cmmSxGetTargetPos ( cmX1, fTargetPos ) = cmERR_NONE Then

                Call cmmStGetPosition ( cmX1, cmCNT_COMM, fGetPos )

                If fTargetPos = fGetPos Then
                        ' 단축 이송 명령에 대한 이송 완료 확인.
                End If
        End If

End Sub
```

```
Delphi

procedure OnGetSxTargetPos ();
var
        fTargetPos : Double;            // 이송 명령의 목표 위치 정보.
        fGetPos : Double;               // 현재 위치 정보.

begin
        if cmmSxGetTargetPos ( cmX1, @fTargetPos ) = cmERR_NONE then
        begin
                cmmStGetPosition ( cmX1, cmCNT_COMM, @fGetPos );

                if fTargetPos = fGetPos then
                begin
                        // 단축 이송 명령에 대한 이송 완료 확인.
                end;
        end;

end;
```

NAME

cmmSxOptSetIniSpeed
cmmSxOptGetIniSpeed
– 단축(單軸) 모션 초기 속도(初期速度) 설정

INFORMATION

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

SYNOPSIS

- ❒ VT_I4 cmmSxOptSetIniSpeed ([in] VT_I4 Axis, [in] VT_R8 IniSpeed)
- ❒ VT_I4 cmmSxOptGetIniSpeed ([in] VT_I4 Axis, [out] VT_PR8 IniSpeed)

DESCRIPTION

모션의 초기 속도를 설정하거나 설정값을 얻어옵니다.

PARAMETER

▶ Axis: 축번호. 축번호는 상수값으로 [cmX1] 부터 0 번째 축을 기준 축으로 임의의 축을 설정할 수 있습니다.

▶ IniSpeed : cmmSxOptSetIniSpeed 함수의 인자이며, 초기 속도를 설정하기 위한 매개변수입니다.

▶ IniSpeed : cmmSxOptGetIniSpeed 함수의 인자이며, 초기 속도를 반환하기 위한 매개변수입니다.

RETURN VALUE

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

EXAMPLE

C/C++

```
#include "Cmmsdk.h"
#include "CmmsdkDef.h"

void OnSetInitSpeed ()
{
	long nAxisNo = 1;	// 초기 속도를 설정할 대상 축을 선택합니다.
	double fIniSpeed;	// 초기 속도 정보.

	/* 해당 축의 초기 속도 값을 확인 후, 초기 속도를 '100'으로 설정합니다.
	if (cmmSxOptGetIniSpeed ( nAxisNo, &fIniSpeed ) == cmERR_NONE )
	{
		if ( fIniSpeed != 100 )
		{
			// 해당 축의 초기 속도 값을 '100'으로 설정합니다.
			cmmSxOptSetIniSpeed ( nAxisNo, 100 );
		}
	}
}
```

Visual Basic

```
Private Sub OnSetInitSpeed ()

        Dim nAxisNo As Long             ' 초기 속도를 설정할 대상 축을 선택합니다.
        Dim fIniSpeed As Long           ' 초기 속도 정보.

        nAxisNo = 1

        ' 해당 축의 초기 속도 값을 확인 후, 초기 속도를 '100'으로 설정합니다.
        If cmmSxOptGetIniSpeed ( nAxisNo, fIniSpeed ) = cmERR_NONE Then

                If fIniSpeed <> 100 Then
                        ' 해당 축의 초기 속도 값을 '100'으로 설정합니다.
                        Call cmmSxOptSetIniSpeed ( nAxisNo, 100 )
                End If
        End If

End Sub
```

Delphi

```
procedure OnSetInitSpeed ();
var
        nAxisNo : LongInt;    // 초기 속도를 설정할 대상 축을 선택합니다.
        fIniSpeed : Double;   // 초기 속도 정보.

begin
        // 해당 축의 초기 속도 값을 확인 후, 초기 속도를 '100'으로 설정합니다.
        if cmmSxOptGetIniSpeed (cmX1, @fIniSpeed ) = cmERR_NONE then
        begin
                if fIniSpeed <> 100 then
                begin
                        // 해당 축의 초기 속도 값을 '100'으로 설정합니다.
                        cmmSxOptSetIniSpeed ( nAxisNo, 100 );
                end;
        end;

end;
```

NAME

cmmSxSetCorrection
cmmSxGetCorrection
– 백래쉬 / 슬립 보정 설정

INFORMATION

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

SYNOPSIS

❒ VT_I4 cmmSxSetCorrection

([in] VT_I4 Axis, [in] VT_I4 CorrMode, [in] VT_R8 CorrAmount, [in] VT_R8 CorrVel,

[in] VT_I4 CntrMask)

❒ VT_I4 cmmSxGetCorrection

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

[out] VT_PI4 CntrMask)

DESCRIPTION

cmmSxSetCorrection() 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.
cmmSxGetCorrection() 함수는 백래쉬/슬립 보정에 대한 현재 설정값을 읽어들이는 함수입니다.

PARAMETER

▶ Axis: 축번호. 축번호는 상수값으로 [cmX1] 부터 0 번째 축을 기준 축으로 임의의 축을 설정할 수 있습니다.

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

BIT No.	Meaning
0 (cmCORR_DIS)	보정기능을 비활성화합니다.
1 (cmCORR_BACK)	보정모드를 백래쉬 보정모드로 설정합니다.
2 (cmCORR_SLIP)	보정모드를 슬립 보정모드로 설정합니다.

▶ CorrMode : cmmSxGetCorrection 함수의 인자이며, 반환값은 다음과 같습니다.

BIT No.	Meaning
0 (cmCORR_DIS)	보정기능이 비활성화 상태입니다.
1 (cmCORR_BACK)	보정모드가 백래쉬 보정모드로 동작중입니다.
2 (cmCORR_SLIP)	보정모드가 슬립 보정모드로 동작중입니다.

▶ CorrAmount : cmmSxSetCorrection 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 : cmmSxGetCorrection 함수의 인자이며, 보정 펄스의 수를 반환합니다.

▶ CorrVel : cmmSxSetCorrection 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 : cmmSxGetCorrection 함수의 인자이며, 보정펄스의 주파수를 반환합니다.

▶ CntrMask : : cmmSxSetCorrection 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 : cmmSxGetCorrection 함수의 인자이며, 보정펄스가 출력되는 동안에 각 카운터의 동작여부를 비트별로 반환합니다. 각 비트가 나타내느 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
cmERR_NONE	수행 성공

EXAMPLE 1

C/C++

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) //
cmmSxSetCorrection (cmX1, cmCORR_BACK, 1000, 1000, 0x0);

////////////////////////////////////////////////////////////
// 이전에 (-)방향 이동을 수행하였다면 아래에서 백래쉬 보정을 수행합니다.
// 1000 PPS 의 속도로 (+)1000 펄스를 출력한 후에 지정한 SxMove()가 수행됩니다.
cmmSxMove(cmX1, 10000);

// 이동방향이 이전과 동일하므로 아래에서는 백래쉬보정을 하지 않습니다.
cmmSxMove(cmX1, 10000);
```

```
// 이동방향이 이전과 동일하므로 아래에서는 백래쉬보정을 하지 않습니다.
cmmSxMove(cmX1, 10000);

////////////////////////////////////////////////////////////
// 이동방향이 전환되므로 아래에서 백래쉬 보정을 수행합니다. 1000 PPS 의 속도로 (-)1000 펄스를 출력한
// 후에 지정한 SxMove()가 수행됩니다.
cmmSxMove(cmX1, -10000);

// 이동방향이 이전과 동일하므로 아래에서는 백래쉬보정을 하지 않습니다.
cmmSxMove(cmX1, -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.

'// 백래쉬보정 모드로 설정 (보정펄스수:1000, 보정펄스출력속도:1000 PPS)
Call cmmSxSetCorrection(cmX1, cmCORR_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 cmmSxMove(cmX1, 10000)

'// 이동방향이 이전과 동일하므로 아래에서는 백래쉬보정을 하지 않습니다.
Call cmmSxMove(cmX1, 10000)

'// 이동방향이 이전과 동일하므로 아래에서는 백래쉬보정을 하지 않습니다.
Call cmmSxMove(cmX1, 10000)

'////////////////////////////////////////////////////////////
'// 이동방향이 전환되므로 아래에서 백래쉬 보정을 수행합니다. 1000 PPS 의
'속도로(-)1000 펄스를 출력한 후에 지정한 SxMove()가 수행됩니다.
Call cmmSxMove(cmX1, -10000)
'// 이동방향이 이전과 동일하므로 아래에서는 백래쉬보정을 하지 않습니다.
Call cmmSxMove(cmX1, -10000)
```

Delphi

```
procedure OnSetSxCorrection ();
begin

    { 해당 축에 대해 백래쉬 보정 모드를 설정 합니다.
    ( 보정 펄스 수 : 1000, 보정 펄스 출력 속도 : 1000 PPS ) }
    cmmSxSetCorrection (cmX1, cmCORR_BACK, 1000, 1000, $1 );

end;


procedure OnMove ();
begin

    { 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. }
    cmmSxMove (cmX1, 10000, cmFALSE );

    // 이송 방향이 이전과 동일하므로 아래에서는 백래쉬 보정을 하지 않습니다.
    cmmSxMove (cmX1, 10000, cmFALSE );

    { 이송 방향이 전환되므로 백래쉬 보정을 수행합니다.
    1000PPS 의 속도로 (-)1000 펄스를 출력 한 후에 지정한 SxMove()가 수행됩니다. }
    cmmSxMove (cmX1, -10000, cmFALSE )

end;
```

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++

```
// 슬립보정 모드로 설정 (보정펄스수:1000, 보정펄스출력속도:1000 PPS) //
cmmSxSetCorrection (cmX1, cmCORR_SLIP, 1000, 1000, 0);

// (+)1000 펄스의 보정펄스가 출력된 후에 SxMove()가 수행됩니다
cmmSxMove(cmX1, 10000);

// (+)1000 펄스의 보정펄스가 출력된 후에 SxMove()가 수행됩니다
cmmSxMove(cmX1, 10000);

// (-)1000 펄스의 보정펄스가 출력된 후에 SxMove()가 수행됩니다
cmmSxMove(cmX1, -10000);

// (-)1000 펄스의 보정펄스가 출력된 후에 SxMove()가 수행됩니다
cmmSxMove(cmX1, -10000);
```

Visual Basic

```
'// 슬립보정 모드로 설정 (보정펄스수:1000, 보정펄스출력속도:1000 PPS)
Call cmmSxSetCorrection(cmX1, cmCORR_SLIP, 1000, 1000, 0)

'// (+)1000 펄스의 보정펄스가 출력된 후에 SxMove()가 수행됩니다
Call cmmSxMove(cmX1, 10000)

'// (+)1000 펄스의 보정펄스가 출력된 후에 SxMove()가 수행됩니다
Call cmmSxMove(cmX1, 10000)

'// (-)1000 펄스의 보정펄스가 출력된 후에 SxMove()가 수행됩니다
Call cmmSxMove(cmX1, -10000)

'// (-)1000 펄스의 보정펄스가 출력된 후에 SxMove()가 수행됩니다
Call cmmSxMove(cmX1, -10000)
```

Delphi

```
// 슬립보정 모드로 설정 (보정펄스수:1000, 보정펄스출력속도:1000 PPS)
cmmSxSetCorrection (cmX1, cmCORR_SLIP, 1000, 1000, 0);

// (+)1000 펄스의 보정펄스가 출력된 후에 SxMove()가 수행됩니다
cmmSxMove(cmX1, 10000);

// (+)1000 펄스의 보정펄스가 출력된 후에 SxMove()가 수행됩니다
cmmSxMove(cmX1, 10000);

// (-)1000 펄스의 보정펄스가 출력된 후에 SxMove()가 수행됩니다
cmmSxMove(cmX1, -10000);

// (-)1000 펄스의 보정펄스가 출력된 후에 SxMove()가 수행됩니다
cmmSxMove(cmX1, -10000);

end;
```

NAME

cmmSxOptSetSyncMode
cmmSxOptGetSyncMode
– 다른 축 동기 구동 설정

INFORMATION

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

SYNOPSIS

❒ VT_I4 cmmSxOptSetSyncMode

([in] VT_I4 Axis, [in] VT_I4 Mode, [in] VT_I4 RefAxis [in] VT_I4 Condition)

❒ VT_I4 cmmSxOptGetSyncMode

([in] VT_I4 Axis, [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

▶ Axis: 축번호. 축번호는 상수값으로 [cmX1] 부터 0 번째 축을 기준 축으로 임의의 축을 설정할 수 있습니다.

▶ Mode : cmmSxOptSetSyncMode 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 : cmmSxOptGetSyncMode 함수의 인자이며, 동기모드의 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 : cmmSxOptSetSyncMode 함수의 인자이며, 내부 동기신호를 발생할 때 참조할 축 번호 또는 마스크를 지정합니다.

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 : cmmSxOptGetSyncMode 함수의 인자이며, 내부 동기신호를 발생할 때 참조하는 축 번호 또는 마스크를 반환합니다.

▶ Condition : cmmSxOptSetSyncMod 함수의 인자이며, 이 값은 Mode 매개 변수(媒介變數)가 "1"로 지정되었을 때에만 의미를 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 : cmmSxOptGetSyncMod 함수의 인자이며, 반환하는 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
cmERR_NONE	수행 성공

NAME

cmmSxOptSetSyncOut
cmmSxOptGetSyncOut
– 속도(速度) 구간 별 디지털 출력(出力) 설정

INFORMATION

Single Axis Control

VC++/VB

BCB/Delphi/.NET

Level 3

다소 주의

함수의 전달 인자 중 'DoChan_local' 은 전역 채널이 아닙니다. 이 의미는 개별 모션 보드 상의 디지털 출력 채널을 의미하므로, 로컬 채널임을 반드시 주의합니다.

SYNOPSIS

❒ VT_I4 cmmSxOptSetSyncOut

([in] VT_I4 Axis, [in] VT_I4 Mode, [in] VT_I4 DoChan_local, [in] VT_I4 DoLogic)

❒ VT_I4 cmmSxOptGetSyncOut

([in] VT_I4 Axis, [out] VT_PI4 Mode, [out] VT_PI4 DoChan_local, [out] VT_PI4 DoLogic)

DESCRIPTION

cmmSxOptSetSyncOut() 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. cmmSxOptGetSyncOut() 함수는 지정한 축의 각 속도 구간에서 고속 디지털 출력의 발생에 대한 설정을 반환합니다.

PARAMETER

▶ Axis: 축번호. 축번호는 상수값으로 [cmX1] 부터 0 번째 축을 기준 축으로 임의의 축을 설정할 수 있습니다.

▶ Mode : cmmSxOptSetSyncOut 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 : cmmSxOptGetSyncOut 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.

▶ DoChannel_local : cmmSxOptSetSyncOut 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 CMMSDK 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 매개 변수(媒介變數)에 사용될 수 있는 번호는 장치의 순서에 관계없이 0 ~ 5 가 되는 것입니다.

▶ DoChannel_local : cmmSxOptGetSyncOut 함수의 인자이며, 범용 디지털 출력으로 사용되는 디지털 출력 채널을 반환합니다.

▶ DoLogic : cmmSxOptSetSyncOut 함수의 인자이며, 디지털 출력 채널의 로직을 설정합니다.

Value	Meaning
cmLOGIC_A	A 접점 방식 => 평상시 Open, 감지되면 Close 되는 스위치 방식
cmLOGIC_B	B 접점 방식 => 평상시 Close, 감지되면 Open 되는 스위치 방식

▶ DoLogic : cmmSxOptGetSyncOut 함수의 인자이며, 디지털 출력 채널의 로직을 반환합니다.

Value	Meaning
cmLOGIC_A	A 접점 방식 => 평상시 Open, 감지되면 Close 되는 스위치 방식
cmLOGIC_B	B 접점 방식 => 평상시 Close, 감지되면 Open 되는 스위치 방식

RETURN VALUE

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

NAME

cmmSxOptSetRdpOffset
cmmSxOptGetRdpOffset
- 감속(減速) 시작 상대 위치 설정

INFORMATION

Single Axis Control

VC++/VB

BCB/Delphi/.NET

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 cmmSxOptSetRdpOffset ([in] VT_I4 Axis, [in] VT_R8 OffsetDist)

❒ VT_I4 cmmSxOptGetRdpOffset ([in] VT_I4 Axis, [out] VT_PR8 OffsetDist)

DESCRIPTION

cmmSxOptSetRdpOffset() 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

▶ Axis: 축번호. 축번호는 상수값으로 [cmX1] 부터 0 번째 축을 기준 축으로 임의의 축을 설정할 수 있습니다.

▶ OffsetDist : cmmSxOptSetRdpOffset 함수의 인자이며, RdpOffset 을 적용할 거리를 설정합니다. 이 거리는 논리적 거리 단위 입니다.

▶ OffsetDist : cmmSxOptGetRdpOffset 함수의 인자이며, RdpOffset 이 적용된 거리를 반환합니다. 이 거리는 논리적 거리 단위 입니다.

RETURN VALUE

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

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 Move와 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 cmmCfgSetSpeedPattern() 함수를 사용하여야 합니다.

8.2.1 함수 요약

Summary of Functions
❒ VT_I4 cmmMxMove ([in] VT_I4 NumAxes, [in] VT_PI4 AxisList, [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 cmmMxMoveStart ([in] VT_I4 NumAxes, [in] VT_PI4 AxisList, [in] VT_PI4 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 cmmMxMoveTo ([in] VT_I4 NumAxes, [in] VT_PI4 AxisList, [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 cmmMxMoveToStart ([in] VT_I4 NumAxes, [in] VT_PI4 AxisList, [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 cmmMxVMoveStart ([in] VT_I4 NumAxes, [in] VT_PI4 AxisList, [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 cmmMxStop ([in] VT_I4 NumAxes, [in] VT_PI4 AxisList, [in] VT_I4 IsWaitComplete, [in] VT_I4 IsBlocking) 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 cmmMxStopEmg ([in] VT_I4 NumAxes, [in] VT_PI4 AxisList) 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 cmmMxIsDone ([in] VT_I4 NumAxes, [in] VT_PI4 AxisList, [out] VT_PI4 IsDone) 다축(多軸) 이송의 완료(完了) 를 확인(確認)합니다.
❒ VT_I4 cmmMxWaitDone ([in] VT_I4 NumAxes, [in] VT_PI4 AxisList, [in] VT_I4 IsBlocking) 다축(多軸) 이송의 완료(完了) 시점까지 대기(待機)합니다.

8.2.2 함수 설명

NAME

cmmMxMove
cmmMxMoveStart
– 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 cmmMxMove

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

❒ VT_I4 cmmMxMoveStart

([in] VT_I4 NumAxes, [in] VT_PI4 AxisList, [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.
cmmMxMove() 함수는 지정한 모든 축의 모션이 완료되기 전까지 반환되지 않으며, cmmMxMoveStart() 함수는 모션을 시작시킨 후에 바로 반환됩니다.

PARAMETER

▶ NumAxes : 동시에 작업을 수행할 대상 축의 수

▶ AxisList : 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 일치해야 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
cmFALSE	블록(Blocking)을 하지 않습니다. 따라서 해당 모션이 완료되는 동안에도 윈도우 이벤트를 처리합니다.
cmTRUE	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
cmERR_NONE	수행 성공

REFERENCE

□ cmmMxMoveStart 함수를 사용하는 경우에는 cmmMxIsDone() 함수나 cmmMxWaitDone() 함수를 사용하여 모션의 완료를 확인(確認)할 수 있습니다.
□ cmmMxMove 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.

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

```
C/C++
#include "Cmmsdk.h"
#include "CmmsdkDef.h"

/*************************************************************
* OnProgramInitial : 이 함수는 가상의 함수로서 프로그램 초기화 루틴이
* 적용되는 부분을 의미합니다.
*************************************************************/
void OnProgramInitial()
{
	long m_nNumAxes;
	cmmLoadDll();
	if(cmmGnDeviceLoad(cmTRUE, &m_nNumAxes) != cmERR_NONE)
	{
		//Handle 은 사용자가 생성한 폼의 핸들 값입니다.
		cmmErrShowLast(Handle);
		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.
	cmmCfgSetSpeedPattern(cmX1, cmSMODE_S, m_fVwork, m_fAcc, m_fDec );
	cmmCfgSetSpeedPattern(cmY1, cmSMODE_S, m_fVwork, m_fAcc, m_fDec );
	cmmCfgSetSpeedPattern(cmZ1, cmSMODE_S, m_fVwork, m_fAcc, m_fDec );
}

/*************************************************************
* DoMotion : 작업명령시에 호출되는 가상의 함수 입니다.
*************************************************************/
void DoMotion()
{
	long nAxisList[3] = {cmX1, cmY1, cmZ1};//움직일 축의 목록입니다.
	double fDistList[3] = {5000, 5000, 5000};//각축의 이동 거리입니다.

	//cmmSetCfgSetSpeedPattern 으로 설정된 축(Axis)의 속도모드를
	//그대로 유지 하면서 cmmCfgSetSpeedPattern 에서 설정된 작업속도,
	//가속도, 감속도의 100%로 설정합니다.
	cmmSxSetSpeedRatio(cmX1, cmSMODE_KEEP, 100, 100, 100);
	cmmSxSetSpeedRatio(cmY1, cmSMODE_KEEP, 100, 100, 100);
	cmmSxSetSpeedRatio(cmZ1, cmSMODE_KEEP, 100, 100, 100);

	//세개의 축을 상대거리 5000 만큼 이동 시킵니다.
	cmmMxMove(3, nAxisList, fDistList, cmFALSE);

	//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.
	cmmMxMove(3, nAxisList, fDistList, cmFALSE);

	//위의 cmmMxMove()함수 대신에 cmmMxMoveStart() 함수를 사용하려면 아래코드를 사용합니다.
	//cmmMxMoveStart(3, nAxisList, fDistList);
	//cmmMxWaitDone(3, AxisList, cmFALSE);
	//fDistList[0] = -5000; fDistList[1] = -5000; fDistList[2] = -5000;
	//cmmMxMoveStart(3, nAxisList, fDistList);
	//cmmMxWaitDone(3, AxisList, cmFALSE);
}
```

```
Visual Basic

‘=================================================================
‘cmmGnDeviceLoad 함수로 장치를 초기화 합니다.
‘=================================================================

Private Sub Form_Load()
   Dim nTotalAxis As Long
   Dim IRetVal As Long
 '=================================================================
 ' cmmGnDeviceLoad 함수로 장치를 초기화합니다.
 IRetVal = cmmGnDeviceLoad(cmTRUE, nTotalAxis)

 If IRetVal <> cmERR_NONE Then
    MsgBox ("cmmGnDeviceLoad has been failed")
 End If
  '=================================================================
End Sub

Private Sub CfgSpeed(nTotalAxis As Long)

	 Dim i As Integer
	'=================================================================
	' 이 함수에서 cmmCfgSetSpeedPattern 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 cmmCfgSetSpeedPattern(i, cmSMODE_S, 10000, 20000, 20000)
   Next

End Sub

Private Sub btnMove_Click()

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

   AxisList(0) = 0
   AxisList(1) = 1

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

   ' 이 함수로 인해 설정된 기준속도의 비율(%) 로 모션 동작을 수행합니다.
   Call cmmSxSetSpeedRatio(cmX1, cmSMODE_S, 100#, 100#, 100#)
   Call cmmSxSetSpeedRatio(cmY1, cmSMODE_S, 100#, 100#, 100#)


   ' 각 인자는 순서대로
   ' 대상축의 갯수, 대상 축의 배열, 대상 거리의 배열, 블록 여부입니다.
    Call cmmMxMove( 2, AxisList(0), DistanceList(0), cmFALSE)

End Sub
```

```
Delphi

// * 이 함수는 폼이 생성될때 이벤트에 의해 불려지며 , 장치를 로드하는 함수입
// * 니다.
procedure OnCreate();
var
        g_nAxis : LongInt;
begin

        // Load CMMSDK(DLL) Library
        if ( cmmGnDeviceLoad(cmTRUE,@g_nAxis) <> cmERR_NONE ) then
        begin
                // 마지막에 발생한 에러를 화면에 표시합니다.
                // 함수 인자로는 Form 의 Handle 이 전달됩니다.
                cmmErrShowLast(Handle);
                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 := cmSMODE_S;
          // 설정된 기준 속도를 실제 SDK 함수에 전달합니다.


cmmCfgSetSpeedPattern(
      cmX1,// 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.

      cmmCfgSetSpeedPattern(
        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.

end;

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

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

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

begin
        AxisList[0] := cmX1;
        AxisList[1] := cmY1;
        DistanceList[0] := 1000;
        DistanceList[1] := 2000;

        //cmmSxSetSpeedRation 를 통하여서 기준 속도와의 비에 따른 실제 구동
        // 속도를 설정 합니다.
        // 여기서 기준 속도란 cmmCfgSetSpeedPattern 함수를 통해 설정된
        // 속도를 의미하며, 아래의 cmmSxSetSpeedRatio 함수는
        // 단축(Single Axis)를 대상으로 축의 속도를
        // 기준 속도 대비 Percent(%) 단위로 입력 받아 설정하는 함수입니다.

          fAccelSpeedRatio := 100;
          fDecelSpeedRatio := 100;
          fWorkSpeedRatio := 100;

          cmmSxSetSpeedRatio(cmX1,cmSMODE_KEEP,fWorkSpeedRatio,  fAccelSpeedRatio, fDecelSpeedRatio);
          cmmSxSetSpeedRatio(cmY1,cmSMODE_KEEP,fWorkSpeedRatio,  fAccelSpeedRatio, fDecelSpeedRatio);

        cmmMxMove(AxisList[0], @AxisList, @DistanceList, cmFALSE);

end;
```

NAME

cmmMxMoveTo
cmmMxMoveToStart
– 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 cmmMxMoveTo

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

❒ VT_I4 cmmMxMoveToStart

([in] VT_I4 NumAxes, [in] VT_PI4 AxisList, [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. cmmMxMoveTo() 함수는 지정한 모든 축의 모션이 완료되기 전까지 반환되지 않으며, cmmMxMoveToStart() 함수는 모션을 시작시킨 후에 바로 반환됩니다.

PARAMETER

▶ NumAxes : 동시에 작업을 수행할 대상 축의 수

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

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

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

Value	Meaning
cmFALSE	블록(Blocking)을 하지 않습니다. 따라서 해당 모션이 완료되는 동안에도 윈도우 이벤트를 처리합니다.
cmTRUE	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
cmERR_NONE	수행 성공

REFERENCE

□ 논리적 거리 단위는 cmmCfgSetUnitDist() 함수에 의해 결정됩니다.

□ cmmMxMoveToStart() 함수를 사용하는 경우에는 cmmMxIsDone() 함수나 cmmMxWaitDone() 함수를 사용하여 모션의 완료를 확인(確認)할 수 있습니다.

□ cmmMxMoveTo() 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.

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 cmmMxMoveTo 를 사용하여 X1,Y1,Z1 축을 절대좌표 (1000,1000,1000) 지점으로 이동한 후 다시 절대좌표 (0,0,0) 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 "Cmmsdk.h"
#include "CmmsdkDef.h"

/*************************************************************
* OnProgramInitial : 이 함수는 가상의 함수로서 프로그램 초기화 루틴이
* 적용되는 부분을 의미합니다.
*************************************************************/
void OnProgramInitial()
{
	long m_nNumAxes;
	cmmLoadDll();
	if(cmmGnDeviceLoad(cmTRUE, &m_nNumAxes) != cmERR_NONE)
	{
		//Handle 은 사용자가 생성한 폼의 핸들 값입니다.
		cmmErrShowLast(Handle);
		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.
	cmmCfgSetSpeedPattern(cmX1, cmSMODE_S, m_fVwork, m_fAcc, m_fDec );
	cmmCfgSetSpeedPattern(cmY1, cmSMODE_S, m_fVwork, m_fAcc, m_fDec );
	cmmCfgSetSpeedPattern(cmZ1, cmSMODE_S, m_fVwork, m_fAcc, m_fDec );
}

/************************************************************
* DoMotion : 작업명령시에 호출되는 가상의 함수 입니다.
************************************************************/
void DoMotion()
{
	long nAxisList[3] = {cmX1, cmY1, cmZ1};//움직일 축의 목록입니다.
	double fPosList[3] = {1000, 1000, 1000};//각축의 이동할 거리입니다.

	//cmmSetCfgSetSpeedPattern 으로 설정된 축(Axis)의 속도모드를
	//그대로 유지 하면서 cmmCfgSetSpeedPattern 에서 설정된 작업속도,
	//가속도, 감속도를 100%로 설정합니다.
	cmmSxSetSpeedRatio(cmX1, cmSMODE_KEEP, 100, 100, 100);
	cmmSxSetSpeedRatio(cmY1, cmSMODE_KEEP, 100, 100, 100);
	cmmSxSetSpeedRatio(cmZ1, cmSMODE_KEEP, 100, 100, 100);

	//세개의 축을 절대좌표 5000 으로 이동 시킵니다.
	cmmMxMoveTo(3, AxisList, fPosList, cmFALSE);

	//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.
	cmmMxMoveTo(3, AxisList, fPosList, cmFALSE);

}
```

```
Visual Basic

'=============================================================
'cmmGnDeviceLoad 함수로 장치를 초기화 합니다.
'=============================================================

Private Sub Form_Load()

	Dim nTotalAxis As Long
	Dim IRetVal As Long
	 '=============================================================
	 ' cmmGnDeviceLoad 함수로 장치를 초기화합니다.
	  IRetVal = cmmGnDeviceLoad(cmTRUE, nTotalAxis)

	  If IRetVal <> cmERR_NONE Then
	     MsgBox ("cmmGnDeviceLoad has been failed")
	  End If
	 '=============================================================

End Sub

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

	'=============================================================
	 ' 이 함수에서 cmmCfgSetSpeedPattern 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 cmmCfgSetSpeedPattern(i, cmSMODE_S, 10000, 20000, 20000)
        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 AxisList(2) As Long
        Dim nRetVal As Long

        AxisList(0) = 0
        AxisList(1) = 1

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

        ' 이 함수로 인해 설정된 기준속도의 비율(%) 로 모션 동작을 수행합니다.
        Call cmmSxSetSpeedRatio(cmX1, cmSMODE_S, 100#, 100#, 100#)
        Call cmmSxSetSpeedRatio(cmY1, cmSMODE_S, 100#, 100#, 100#)


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

        nRetVal = cmmMxMoveTo(2, AxisList(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
        g_nAxis : LongInt;
begin

        // Load CMMSDK(DLL) Library
        if ( cmmGnDeviceLoad(cmTRUE,@g_nAxis) <> cmERR_NONE ) then
        begin
                // 마지막에 발생한 에러를 화면에 표시합니다.
                // 함수 인자로는 Form 의 Handle 이 전달됩니다.
                cmmErrShowLast(Form1.Handle);
                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 := cmSMODE_S;
                        // 설정된 기준 속도를 실제 SDK 함수에 전달합니다.

                       cmmCfgSetSpeedPattern(
                       cmX1,// 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.

                       cmmCfgSetSpeedPattern(
                       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.

end;

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

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

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

begin
        AxisList[0] := cmX1;
        AxisList[1] := cmY1;
        DistanceList[0] := 1000;
        DistanceList[1] := 2000;

        //cmmSxSetSpeedRation 를 통하여서 기준 속도와의 비에 따른 실제 구동
        // 속도를 설정 합니다.
        // 여기서 기준 속도란 cmmCfgSetSpeedPattern 함수를 통해 설정된
        // 속도를 의미하며, 아래의 cmmSxSetSpeedRatio 함수는
        // 단축(Single Axis)를 대상으로 축의 속도를
        // 기준 속도 대비 Percent(%) 단위로 입력 받아 설정하는 함수입니다.
        fAccelSpeedRatio := 100;
        fDecelSpeedRatio := 100;
        fWorkSpeedRatio := 100;

        cmmSxSetSpeedRatio(cmX1,cmSMODE_KEEP,fWorkSpeedRatio,  fAccelSpeedRatio, fDecelSpeedRatio);

        cmmSxSetSpeedRatio(cmY1,cmSMODE_KEEP,fWorkSpeedRatio, fAccelSpeedRatio, fDecelSpeedRatio);

        cmmMxMoveTo( 2, @AxisList, @DistanceList, cmFALSE);

end;
```

NAME

cmmMxVMoveStart
- 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 cmmMxVMoveStart ([in] VT_I4 NumAxes, [in] VT_PI4 AxisList, [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.

PARAMETER

▶ NumAxes : 동시에 작업을 수행할 대상 축의 수

▶ AxisList : 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 일치해야 합니다. 모션의 방향을 지시하는 값은 다음과 같습니다.

Value	Meaning
0 또는 cmDIR_N	(-) 방향
1 또는 cmDIR_P	(+) 방향

RETURN VALUE

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

EXAMPLE

```
C/C++
#include "Cmmsdk.h"
#include "CmmsdkDef.h"

/****************************************************************
* OnProgramInitial : 이 함수는 가상의 함수로서 프로그램 초기화 루틴이
* 적용되는 부분을 의미합니다.
****************************************************************/
void OnProgramInitial()
{
        long m_nNumAxes;
        cmmLoadDll();
```

```
	if(cmmGnDeviceLoad(cmTRUE, &m_nNumAxes) != cmERR_NONE)
	{
		//Handle 은 사용자가 생성한 폼의 핸들 값입니다.
		cmmErrShowLast(Handle);
		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.
	cmmCfgSetSpeedPattern(cmX1, cmSMODE_S, m_fVwork, m_fAcc, m_fDec );
	cmmCfgSetSpeedPattern(cmY1, cmSMODE_S, m_fVwork, m_fAcc, m_fDec );
	cmmCfgSetSpeedPattern(cmZ1, cmSMODE_S, m_fVwork, m_fAcc, m_fDec );
}

/************************************************************
* OnDoMotion() : 작업명령시에 호출되는 가상의 함수
* 이 함수에서는 X1, Y1, Z1 축에 대하여 Velocity Move 를 시작합니다.
************************************************************/
void OnDoMotion()
{
	long nAxisList[3] = {cmX1, cmY1, cmZ1};
	long nDirList[3] = {cmDIR_P, cmDIR_P, cmDIR_P}; //Positive dir

	//cmmSetCfgSetSpeedPattern 으로 설정된 축(Axis)의 속도모드를
	//그대로 유지 하면서 설정된 작업속도, 가속도,
	//감속도의 100%로 설정합니다.
	cmmSxSetSpeedRatio(cmX1, cmSMODE_KEEP, 100, 100, 100);
	cmmSxSetSpeedRatio(cmY1, cmSMODE_KEEP, 100, 100, 100);
	cmmSxSetSpeedRatio(cmZ1, cmSMODE_KEEP, 100, 100, 100);

	// Start V-Move of X1&Y1&Z1 //
	if(cmmMxVMoveStart(3, nAxisList, nDirList) != cmERR_NONE){
		cmmErrShowLast(Handle);
		return;
	}
}
```

```
Visual Basic

Private Sub CfgSpeed(nTotalAxis As Long)

  Dim i As Integer
  '============================================================
  ' 이 함수에서 cmmCfgSetSpeedPattern 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 cmmCfgSetSpeedPattern(i, cmSMODE_S, 10000, 20000, 20000)
  Next

End Sub

Private Sub btnMove_Click()

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

  AxisList(0) = 0
```

```
    AxisList(1) = 1

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

    ' 이 함수로 인해 설정된 기준속도의 비율(%) 로 모션 동작을 수행합니다.
    Call cmmSxSetSpeedRatio(cmX1, cmSMODE_S, 100#, 100#, 100#)
    Call cmmSxSetSpeedRatio(cmY1, cmSMODE_S, 100#, 100#, 100#)


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


End Sub
```

```
Delphi

Procedure btnSetSpeedClick();
var
    fAccelSpeed : Double;
    fDecelSpeed : Double;
    fWorkSpeed : Double;
    nSMODE : LongInt;

begin
        fAccelSpeed := 30000;
        fDecelSpeed := 30000;
        fWorkSpeed := 10000;
        nSMODE := cmSMODE_S;

        // cmX1 을 위해 설정된 기준 속도를 실제 SDK 함수에 전달합니다.
        cmmCfgSetSpeedPattern(
        cmX1,               // 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.

       // cmY1 을 위해 설정된 기준 속도를 실제 SDK 함수에 전달합니다.
        cmmCfgSetSpeedPattern(
        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.
end;

procedure FormCreate();
var
        g_nAxis : LongInt;
begin

        // Load CMMSDK(DLL) Library
        if ( cmmGnDeviceLoad(cmTRUE,@g_nAxis) <> cmERR_NONE ) then
        begin
                // 마지막에 발생한 에러를 화면에 표시합니다.
                // 함수 인자로는 Form 의 Handle 이 전달됩니다.
                cmmErrShowLast(Handle);
                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;

  AxisList : 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;

      AxisList[0] := cmX1;
      AxisList[1] := cmY1;

      ////////////////////////////////////////////////////
      // Slider Bar 를 통해 각 속도를 기준속도 대비의 비율로  구합니다.
      // 여기서 기준 속도란 cmmCfgSetSpeedPattern 함수를 통해 설정된
      // 속도를 의미하며, 아래의 cmmSxSetSpeedRatio 함수는
      // 단축(Single Axis)를 대상으로 축의 속도를
      // 기준 속도 대비 Percent(%) 단위로 입력 받아 설정하는 함수입니다.
      fAccelSpeedRatio := 100;
      fDecelSpeedRatio := 100;
      fWorkSpeedRatio := 100;

      cmmSxSetSpeedRatio(
        cmX1,             // X1 축
        cmSMODE_KEEP,     // 이전 가감속 모드를 그대로 설정
        fWorkSpeedRatio,
        fAccelSpeedRatio,
        fDecelSpeedRatio);

      cmmSxSetSpeedRatio(
        cmY1,             // Y1 축
        cmSMODE_KEEP,     // 이전 가감속 모드를 그대로 설정
        fWorkSpeedRatio,
        fAccelSpeedRatio,
        fDecelSpeedRatio);

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

end;
```

NAME

cmmMxStop
cmmMxStopEmg
- 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)(비상 정지(停止)를 주의하십시요. 기구물의 손상이나 안전사고에 원인이 될 수 있습니다.

SYNOPSIS

❒ VT_I4 cmmMxStop

([in] VT_I4 NumAxes, [in] VT_PI4 AxisList, [in] VT_I4 IsWaitComplete, [in] VT_I4 IsBlocking)

❒ VT_I4 cmmMxStopEmg

([in] VT_I4 NumAxes, [in] VT_PI4 AxisList)

DESCRIPTION

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

PARAMETER

▶ NumAxes : 동시에 작업을 수행할 대상 축의 수

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

▶ IsWaitComplete : 완료될때까지 기다리는지 여부.

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

Value	Meaning
cmFALSE	블록(Blocking)을 하지 않습니다. 따라서 해당 모션이 완료되는 동안에도 윈도우 이벤트를 처리합니다.
cmTRUE	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
cmERR_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 nAxes[4]={0, 1, 2, 3};
	GetDlgItem(IDC_btnStop)->EnableWindow(FALSE);
	cmmMxStop(4, nAxes, TRUE, FALSE);
	GetDlgItem(IDC_btnStop)->EnableWindow(TRUE);
}
```

Visual Basic

```
Private Sub btnStop_Click()
	Dim nRetVal As Long

	Dim AxisList(2) As Long

	AxisList(0) = 0
	AxisList(1) = 1

	' 각 인자에 대한 설명을 드립니다.
	' cmmMxStop( 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
	' cmmMxStop 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 = cmmMxStop(2, AxisList(0), True, GetIsBlocking())

End Sub
```

Delphi

```
// * Description :
// * 이 함수는 버튼 이벤트에 의해 모션 동작을 정지(停止)하는 함수입니다.
// *
procedure btnStopClick();
var
	AxisList : Array[0..1] of LongInt;
```

```
        gnTargetAxis : LongInt;
begin
        AxisList[0] := cmX1;
        AxisList[1] := cmY1;
        gnTargetAxis := 2;

        // 정지(停止) 함수의 원형은 cmmSxStop([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.
        // cmFALSE 시에는 윈도우 메시지를 함수 내부에서 처리해주게 됩니다.
        //  보다 자세한 설명은 매뉴얼을 참고해주시기 바랍니다.

        cmmMxStop(gnTargetAxis, @AxisList, cmTRUE,  cmFALSE);
end;
```

NAME

cmmMxIsDone
– 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 cmmMxIsDone ([in] VT_I4 NumAxes, [in] VT_PI4 AxisList, [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.

PARAMETER

▶ NumAxes : 동시에 작업을 수행할 대상 축의 수

▶ ChannelList : 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
cmFALSE	모션작업이 완료되지 않음
cmTRUE	모션작업이 완료됨

RETURN VLAUE

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

SEE ALSO

cmmMxWaitDone

EXAMPLE

C/C++

```
long nIsDone;
long nAxisList[2] = {cmX1, cmY1};
double fDistList[2] = {1000, 1000};

if(cmmMxMove(2, nAxisList, fDistList, cmFALSE) != cmERR_NONE){
                    //Handle 은 사용자가 생성한 폼의 핸들 값입니다.
                    cmmErrShowLast(Handle);
                    return;
}

while (1){
```

```
        cmmMxIsDone(2, nAxisList, &nIsDone);
        if(nIsDone == cmTRUE) break;
        else{
                // 다축 모션이 종료되지 않은 경우입니다. 적절한 처리를 합니다.
        }
}
```

Visual Basic

```
Dim nAxisList(2) As Long
Dim fDistList(2) As Double

‘ 대상 축 설정
nAxisList(0) = cmX1
nAxisList(1) = cmY1

‘ 대상 축에 대한 이송 거리 설정
fDistList(0) = 1000
fDistList(1) = 1000

If(cmmMxMove(2, nAxisList(0), fDistList(0), cmFALSE) <> cmERR_NONE) Then
        Call cmmErrShowLast(Handle) ‘에러처리
        Exit Sub
End If

While(cmmMxIsDone(2, nAxisList(0), cmTRUE) <> cmERR_NONE)
        Call cmmErrShowLast(Handle) ‘에러처리
        Exit Sub
End If
```

Delphi

```
// 대상 축 설정
nAxisList[0] := cmX1;
nAxisList[1] := cmY1;

// 대상 축에 대한 이송 거리 설정
fDistList[0] := 1000;
fDistList[1] := 1000;

if(cmmMxMove(2, @nAxisList, @fDistList) <> cmERR_NONE) then begin
        cmmErrShowLast(Handle);
        // 에러 발생에 대한 적절한 처리를 수행합니다.
end;

while(cmmMxIsDone(2, @nAxisList, @IsDone) <> cmERR_NONE) do begin

        // 여기서 IsDone 이 cmTRUE 이면 Loop 를 탈출하면 됩니다.
        …
end;

if(cmmErrGetLastCode() <> cmERR_NONE) then begin
        cmmErrShowLast(Handle);
        // 에러 발생에 대한 적절한 처리를 수행합니다.
end;
```

NAME

cmmMxWaitDone
– 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 cmmMxWaitDone ([in] VT_I4 NumAxes, [in] VT_PI4 AxisList, [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.

PARAMETER

▶ 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
cmFALSE	블록(Blocking)을 하지 않습니다. 따라서 해당 모션이 완료되는 동안에도 윈도우 이벤트를 처리합니다.
cmTRUE	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
cmERR_NONE	수행 성공

SEE ALSO

cmmMxIsDone

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 nAxisList[2] = {cmX1, cmY1};
double fDistList[2] = {1000, 1000};

if(cmmMxMoveStart(2, nAxisList, fDistList) != cmERR_NONE){
        //Handle 은 사용자가 생성한 폼의 핸들 값입니다.
        cmmErrShowLast(Handle);
        return ;
}

//모션이 완료 될 때 까지 기다립니다.
if(cmmMxWaitDone(2, nAxisList, cmFALSE) != cmERR_NONE){
        cmmErrShowLast(Handle);
        return ;
}
```

Visual Basic

```
Dim nAxisList(1)
Dim fDistList(1)

‘ 대상 축 설정
nAxisList(0) = cmX1
nAxisList(1) = cmY1

‘ 대상 축에 대한 이송 거리 설정
fDistList(0) = 1000
fDistList(1) = 1000

if(cmmMxMove(2, nAxisList(0), fDistList(0), cmFALSE) <> cmERR_NONE) then
        call cmmErrShowLast(Form1.Hwnd) ‘에러처리
        exit sub
end if

‘Wait till motion done
if(cmmMxWaitDone(2, nAxisList(0), cmFALSE) <> cmERR_NONE) then
        call cmmErrShowLast(Form1.Hwnd) ’에러처리
        exit sub
end if
```

Delphi

```
// 대상 축 설정
nAxisList[0] := cmX1;
nAxisList[1] := cmY1;

// 대상 축에 대한 이송 거리 설정
fDistList[0] := 1000;
fDistList[1] := 1000;
if(cmmMxMove(2, @nAxisList, @fDistList, cmFALSE) <> cmERR_NONE) then begin
        cmmErrShowLast(self.Handle);
        // 적절한 에러처리를 수행합니다.
end;
// Wait till motion done //
if(cmmMxWaitDone(2, @nAxisList, cmFALSE) <> cmERR_NONE) then begin
        cmmErrShowLast(self.Handle); // 에러처리
        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 모션보드는 통합된 모션 축을 대상으로 총 64 개의 축(Axes) 을 대상으로 보간제어 기능을 제공합니다. 당사의 모션보드가 제공하는 보간제어의 사양표는 다음과 같습니다.

<table>
<tr><th>보간 제어 형태</th><th>축 구성 범위</th><th colspan="2">축 구성 제한 사항
(해당 축 범위 이내)</th></tr>
<tr><td>직선 보간</td><td>64 축(Axes) 이내</td><td colspan="2">제한 없음</td></tr>
<tr><td rowspan="4">원호 보간</td><td rowspan="4">각 모션 보드 내의 2 축</td><td>LX502</td><td>0~1 Axes</td></tr>
<tr><td>LX504</td><td>0~3 Axes</td></tr>
<tr><td rowspan="2">LX508</td><td>0~3 Axes</td></tr>
<tr><td>4~7 Axes</td></tr>
<tr><td rowspan="4">확장 보간제어</td><td rowspan="4">각 모션 보드 내의 3 축</td><td>LX502</td><td>지원안함</td></tr>
<tr><td>LX504</td><td>0~3 Axes</td></tr>
<tr><td rowspan="2">LX508</td><td>0~3 Axes</td></tr>
<tr><td>4~7 Axes</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 "8 축" 모션보드에서의 직선보간

8 축 모션보드는 Axis 0 ~ 3 의 4 개의 축이 축그룹 1 로 구분되고, Axis 4 ~ 7 의 4 개의 축이 축그룹 2 로 구분됩니다. CMMSDK에서는 동일 그룹에 속하는 축들간의 "직선보간" 구동은 물론 서로 다른 축그룹에 속한 축들간의 "직선보간" 구동에 아무런 제약없이 사용하실 수 있습니다. 또한 임의의 보드에 속한 축그룹 간에 상관없이 "직선보간" 구동이 가능합니다. 즉 예를들어 0 번 Axis 와 63 번 Axis 간에 "직선보간" 구동이 가능합니다.

8.3.2 "8 축" 모션보드에서의 원호보간

동일 그룹에 속하는 축들간의 "원호보간" 구동은 아무런 제약없이 사용하실 수 있습니다. 그러나 서로 다른 축그룹에 속한 축들간의 "원호보간" 구동은 허용되지 않습니다. 예를 들어, Axis 0 과 Axis 4 의 두 축간의 원호보간은 오동작하게 됩니다.

8.3.3 함수 요약

"기본보간제어"에 관련된 함수들은 모두 "Ix..."의 형식으로 이름이 구성되었으며 그 리스트는 다음과 같습니다.

Summary of Functions
❒ VT_I4 cmmIxMapAxes ([in] VT_I4 MapIndex, [in] VT_I4 MapMask1, [in] VT_I4 MapMask2) 보간(補間) 대상 축 그룹을(Group) 설정합니다. 최대 32Bit 데이터형으로 표현할 수 있는 두 가지의 매개변수를 사용할 수 있기 때문에 총 64 축을 대상으로 보간 대상 축을 지정할 수 있습니다.
❒ VT_I4 cmmIxSetSpeedPattern ([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(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 cmmIxGetSpeedPattern ([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(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 cmmIxSetSpeedPattern_T ([in] VT_I4 MapIndex, [in] VT_I4 SpeedMode, [in] VT_R8 Vel, [in] VT_R8 AccelTime, [in] VT_R8 DecelTime) 보간(補間) 이송 속도(移送速度)를 설정합니다. 가속 및 감속도 설정 시 펄스 단위가 아닌 시간 단위(msec) 로 설정 합니다.
❒ VT_I4 cmmIxGetSpeedPattern_T ([in] VT_I4 MapIndex,[out] VT_PI4 SpeedMode, [out] VT_PR8 Vel, [out] VT_PR8 AccelTime, [out] VT_PR8 DecelTime) 보간(補間) 이송 속도(移送速度)를 설정합니다. 가속 및 감속도 설정 시 펄스 단위가 아닌 시간 단위(msec) 로 설정 합니다.
❒ VT_I4 cmmIxLine ([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 cmmIxLineStart ([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 cmmIxLineTo ([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 cmmIxLineToStart ([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 cmmIxArcA ([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 cmmIxArcAStart ([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 cmmIxArcATo ([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 cmmIxArcAToStart ([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 cmmIxArcP ([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 cmmIxArcPStart ([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 cmmIxArcPTo ([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 cmmIxArcPToStart ([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 cmmIxArc3P ([in] VT_I4 MapIndex, [in] VT_R8 P2X, [in] VT_R8 P2Y, [in] VT_R8 P3X, [in] VT_R8 P3Y, [in] VT_R8 EndAngle, [in] VT_I4 IsBlocking)
보간(補間) 제어에 원호의 중심점 및 종점 좌표를 사용하는 cmmIxArcP(상대 좌표) 또는 cmmIxArcP (절대 좌표) 함수와는 달리 현재 좌표(1 차)를 포함해서 2 차 및 3 차 좌표의 총 3 개의 (X, Y) 순서쌍 좌표를 사용해서 원호 보간을 수행합니다.

❒ VT_I4 cmmIxArc3PStart ([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(補間) 제어에 원호의 중심점 및 종점 좌표를 사용하는 cmmIxArcP(상대 좌표) 또는 cmmIxArcP (절대 좌표) 함수와는 달리 현재 좌표(1 차)를 포함해서 2 차 및 3 차 좌표의 총 3 개의 (X, Y) 순서쌍 좌표를 사용해서 원호 보간을 수행합니다.
함수 내부에서 이송 완료시까지 기다리지 않고 이송 작업만 시작시킨 뒤 바로 반환 됩니다.

❒ VT_I4 cmmIxArc3PTo ([in] VT_I4 MapIndex, [in] VT_R8 P2X, [in] VT_R8 P2Y, [in] VT_R8 P3X, [in] VT_R8 P3Y, [in] VT_R8 EndAngle, [in] VT_I4 IsBlocking)
원호의 중심점 및 종점 좌표를 사용하는 cmmIxArcP(상대 좌표) 또는 cmmIxArcPTo (절대 좌표) 함수와는 달리 현재 좌표(1 차)를 포함해서 2 차 및 3 차 좌표의 총 3 개의 (X, Y) 순서쌍 좌표를 사용해서 원호 보간을 수행합니다. 절대적 위치로 원호 보간 이송이 됩니다.

❒ VT_I4 cmmIxArc3PToStart ([in] VT_I4 MapIndex, [in] VT_R8 P2X, [in] VT_R8 P2Y, [in] VT_R8 P3X, [in] VT_R8 P3Y, [in] VT_R8 EndAngle)
원호의 중심점 및 종점 좌표를 사용하는 cmmIxArcPStart(상대 좌표) 또는 cmmIxArcPToStart (절대 좌표) 함수와는 달리 현재 좌표(1 차)를 포함해서 2 차 및 3 차 좌표의 총 3 개의 (X, Y) 순서쌍 좌표를 사용해서 원호 보간을 수행합니다. 절대적 위치로 원호 보간 이송이 됩니다.
함수 내부에서 이송 완료시까지 기다리지 않고 이송 작업만 시작시킨 뒤 바로 반환 됩니다.

❒ VT_I4 cmmIxIsDone ([in] VT_I4 MapIndex, [out] VT_PI4 IsDone)
보간(補間) 제어 구동 이송의 보간 완료(補間完了) 를 확인합니다.

❒ VT_I4 cmmIxWaitDone ([in] VT_I4 MapIndex, [in] VT_I4 IsBlocking)
보간(補間) 제어 구동 이송의 보간 완료(補間完了) 시점까지 대기 합니다.

❒ VT_I4 cmmIxStop ([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 cmmIxStopEmg ([in] VT_I4 MapIndex)
보간(補間) 제어 구동 이송을 비상 정지(非常停止) 합니다.

8.3.4 함수 설명

NAME

cmmIxMapAxes
– 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 cmmIxMapAxes ([in] VT_I4 MapIndex, [in] VT_I4 MapMask1, [in] VT_I4 MapMask2)

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.

PARAMETER

▶ MapIndex : 맵번호(Map index), 이 번호는 반드시 0 ~ 7 의 숫자이어야 합니다.

▶ bMapMask1 : 축맵에 포함할 축들을 지정할 마스크 값(하위 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.

▶ bMapMask2: 축맵에 포함할 축들을 지정할 마스크 값(상위 32 비트, BIT32 ~ BIT63).
예) 8 축을 사용하는 경우

Bit Number	Meaning
BIT0 (cmX1_MASK)	0 번 축의 포함여부 : 0 => 포함안함, 1 => 포함
BIT1 (cmY1_MASK)	1 번 축의 포함여부 : 0 => 포함안함, 1 => 포함
BIT2 (cmZ1_MASK)	2 번 축의 포함여부 : 0 => 포함안함, 1 => 포함
BIT3 (cmU1_MASK)	3 번 축의 포함여부 : 0 => 포함안함, 1 => 포함
BIT4 (cmX2_MASK)	4 번 축의 포함여부 : 0 => 포함안함, 1 => 포함
BIT5 (cmY2_MASK)	5 번 축의 포함여부 : 0 => 포함안함, 1 => 포함
BIT6 (cmZ2_MASK)	6 번 축의 포함여부 : 0 => 포함안함, 1 => 포함
BIT7 (cmU2_MASK)	7 번 축의 포함여부 : 0 => 포함안함, 1 => 포함

RETURN VALUE

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

EXAMPLE 1

X1 축과 Y1 축의 직선보간

C/C++ :

```
// 맵번호 설정
#define MAP0                    0

cmmIxMapAxes(MAP0, 0x3, 0x0);
// 또는 cmmIxMapAxes(0, cmX1_MASK | cmY1_MASK, 0); //

// 보간 제어 속도 설정, 두번째 인자는 백터 모드 혹은 마스터 속도 모드를 의미함.
cmmIxSetSpeedPattern(MAP0, cmFALSE cmSMODE_S, 1000, 10000, 10000);

// 보간 이송 거리 리스트 설정
double fDistList[2] = {1000, 1000};
cmmIxLine(MAP0, fDistList);
```

Visual Basic

```
‘ 맵 번호 MAP0 은 이미 선언되어 있다고 가정함

Call cmmIxMapAxes(MAP0, &H3, &H0)
‘ 또는 cmmIxMapAxes(MAP0, cmX1_MASK Or cmY1_MASK, 0)

‘보간 제어 속도 설정, 두번째 인자는 백터 모드 혹은 마스터 속도 모드를 의미함.
Call cmmIxSetSpeedPattern(MAP0, cmFALSE, cmSMODE_S, 1000, 10000, 10000)

‘ 보간 이송 거리 리스트 설정
fDistList(0) = 1000
fDistList(1) = 1000
Call cmmIxLine(MAP0, fDistList(0))
```

Delphi

```
fDistList : Array[0 .. 1] of LongInt;

// 맵 번호 MAP0 은 이미 선언되어 있다고 가정함

cmmIxMapAxes(MAP0, $3, $0);
// 또는 cmmIxMapAxes(0, cmX1_MASK or cmY1_MASK, 0);

//보간 제어 속도 설정, 두번째 인자는 백터 모드 혹은 마스터 속도 모드를 의미함.
cmmIxSetSpeedPattern(MAP0, cmFALSE, cmSMODE_S, 1000, 10000, 10000);

// 보간 이송 거리 리스트 설정
fDistList[0] := 1000;
fDistList[1] := 1000;
cmmIxLine(MAP0, @fDistList, 0);
```

NAME

cmmIxSetSpeedPattern
cmmIxGetSpeedPattern
– 보간(補間) 이송 속도 설정

INFORMATION

- Interpolation Motion
- VC++/VB
- BCB/Delphi/.NET
- Level 3
- 다소 주의

SYNOPSIS

❒ VT_I4 cmmIxSetSpeedPattern

([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 cmmIxGetSpeedPattern

([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

cmmIxSetSpeedPattern 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.
cmmIxGetSpeedPattern()은 "기본보간제어"의 이송속도에 대한 설정된 값을 반환합니다.

PARAMETER

▶ MapIndex: 맵번호(Map index), 이 맵번호를 사용하기전에 먼저 cmmIxMapAxes() 함수를 통하여 해당 맵번호에 유효한 축들이 맵핑되어 있어야 합니다.

▶ IsVertorSpeed : cmmIxSetSpeedPattern 함수의 인자이며, TRUE 로 설정했을 경우에는 벡터스피드 모드로, FALSE 로 설정했을 경우에는 마스터스피드 모드로 설정됩니다.

▶ IsVertorSpeed : cmmIxGetSpeedPattern 함수의 인자이며, 스피드 모드를 반환합니다. TRUE 일 경우엔 벡터스피드 모드, FALSE 일 경우엔 마스터스피드 모드입니다.

▶ SpeedMode : cmmIxSetSpeedPattern 함수의 인자이며, 속도모드를 설정합니다. 설정값은 다음과 같습니다.

Value	Meaning
0 또는 cmSMODE_C	CONSTANT 속도모드 => 가감속을 수행하지 않습니다.
1 또는 cmSMODE_T	TRAPEZOIDAL 속도모드 => 사다리꼴 가감속을 수행합니다.
2 또는 cmSMODE_S	S-CURVE 속도모드 => S-CURVE 가감속을 수행합니다.

▶ SpeedMode : cmmIxGetSpeedPattern 함수의 인자이며, 속도모드를 반환합니다. 반환값은 다음과 같습니다.

Value	Meaning
0 또는 cmSMODE_C	CONSTANT 속도모드 => 가감속을 수행하지 않습니다.
1 또는 cmSMODE_T	TRAPEZOIDAL 속도모드 => 사다리꼴 가감속을 수행합니다.
2 또는 cmSMODE_S	S-CURVE 속도모드 => S-CURVE 가감속을 수행합니다.

▶ VelRatio : cmmIxSetSpeedPattern 함수의 인자이며, 마스터스피드모드 일 때는 작업속도 비율을 설정합니다.
벡터스피드모드 일 때는 PPS 단위를 사용하여 설정 합니다.

▶ VelRatio : cmmIxGetSpeedPattern 함수의 인자이며, 마스터스피드모드 일 때는 작업속도 비율을 반환합니다.
벡터스피드모드 일 때의 반환값은 PPS 단위입니다.

▶ AccRatio : cmmIxSetSpeedPattern 함수의 인자이며, 마스터스피드모드 일 때는 가속도 비율을 설정합니다.
벡터스피드모드 일 때는 PPS 단위를 사용하여 가속도를 설정합니다.

▶ AccRatio : cmmIxGetSpeedPattern 함수의 인자이며, 마스터스피드모드 일 때는 가속도 비율을 반환합니다.
벡터스피드모드 일 때의 반환값은 PPS 단위입니다.

▶ DecRatio : cmmIxSetSpeedPattern 함수의 인자이며, 마스터스피드모드 일 때는 감속도 비율을 설정합니다.
벡터스피드모드 일 때는 PPS 단위를 사용하여 감속도를 설정합니다.

▶ DecRatio : cmmIxGetSpeedPattern 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
cmERR_NONE	수행 성공

REFERENCE

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 cmmIxSetSpeedPattern() 함수의 WorkSpeed 매개 변수(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
cmmIxSetSpeedPattern() 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.

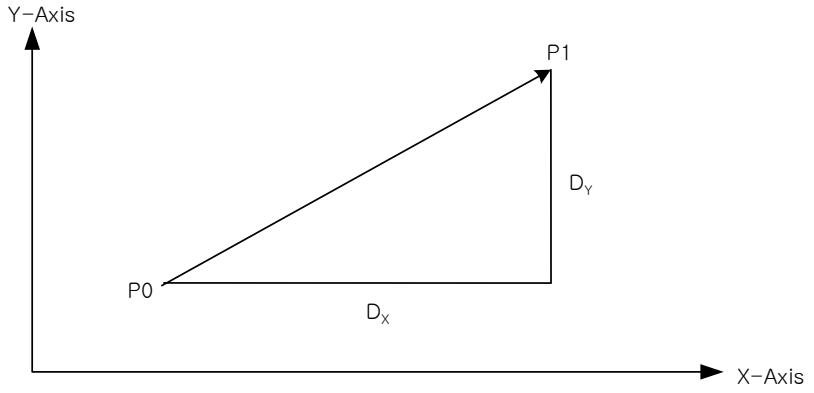


그림 8-1 X, Y 축간의 직선 보간 이송

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 MAP_IDX                    0 // 맵번호 => 0

// 코드의 간결성을 위하여 앞에서 행해져야할 초기화 루틴은 모두 생략 //
……

// X1 축과 Y1 축을 0 번 맵번호로 맵핑 //
cmmIxMapAxes (MAP_IDX, cmX1_MASK | cmY1_MASK, 0);

// 속도패턴 설정 : 벡터속도 1000 PPS, 벡터가속도 10000 PPS/sec (가속시간 0.1 초) //
// 여기서 2 번째 인자가 cmTRUE 이면, Vector 속도 모드를 의미한다.
cmmIxSetSpeedPattern(MAP_IDX, ,cmTRUE, cmSMODE_S, 1000, 10000, 10000);

// 직선보간이동 수행 : (3000, 4000) 만큼 이동 //
double fDistList[2]={3000, 4000};
cmmIxLine (MAP_IDX, fDistList, cmFALSE);
```

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

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

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

EXAMPLE

C/C++

```
#include “Cmmsdk.h”
#include “CmmsdkDef.h”

#define MAP0      0             // 보간 맵 정의

cmmIxMapAxes ( MAP0,  cmX1_MASK | cmY1_MASK, 0 );

/* MAP0 로 설정된 축들을 벡터 스피드 모드로
작업 속도 1000, 가속도 10000, 감속도 10000 으로 설정합니다.*/
cmmIxSetSpeedPattern (      MAP0,               // 보간 맵 번호
                            cmTRUE,             // cmFALSE : 마스터 모드, cmTRUE : 벡터 모드
                            cmSMODE_T,          // 가감속 모드
                            1000,               // 마스터 모드: 작업 속도 비율, 벡터 모드: 작업 속도 (PPS)
                            10000,              // 마스터 모드 : 가속도 비율, 벡터 모드: 가속도 (PPS)
                            10000               // 마스터 모드 : 감속도 비율, 벡터 모드: 감속도 (PPS)
                            );

/* 현재 위치가 (0, 0) 이라고 가정할 때, (3000, 4000) 좌표로 직선 보간 이송을 수행합니다. 벡터 속도를 ‘1000’ 으로
지정하였으므로 위 계산식에 의하여 Vx = 600, Vy = 800 의 속도로 이송합니다. */
Double fDistList[2] = {3000, 4000};
cmmIxLineStart ( MAP0, fDistList );
```

Visual Basic

```
Dim nMapIdx As Long         ‘보간 맵 정의
Dim fDistList(2) As Double   ‘ 이송 좌표값의 배열 정보

nMapIdx = 0

Call cmmIxMapAxes ( nMapIdx, cmX1_MASK Or cmY1_MASK, 0 )

‘ MAP0 로 설정된 축들을 벡터 스피드 모드로
‘ 작업 속도 1000, 가속도 10000, 감속도 10000 으로 설정합니다.
Call cmmIxSetSpeedPattern ( nMapIdx, cmTRUE, cmSMODE_T, 1000, 10000, 10000 )

‘ 현재 위치가 (0, 0) 이라고 가정할 때, (3000, 4000) 좌표로 직선 보간 이송을 수행합니다.
‘ 벡터 속도를 ‘1000’ 으로 지정하였으므로 위 계산식에 의하여 Vx = 600, Vy = 800 의 속도로 이송합니다.
fDistList(0) = 3000
fDistList(1) = 4000
Call cmmIxLineStart ( nMapIdx, fDistList(0) )
```

Delphi

```
var
        nMapIdx  : LongInt;                 // 보간 맵 정의
        fDistList : Array[0..1] of Double;  // 이송 좌표값의 배열 정보
```

```
begin
        nMapIdx := 0;

        cmmIxMapAxes ( nMapIdx, cmX1_MASK or cmY1_MASK, 0 );

        { MAP0 로 설정된 축들을 벡터 스피드 모드로
        작업 속도 1000, 가속도 10000, 감속도 10000 으로 설정합니다. }
        cmmIxSetSpeedPattern ( nMapIdx, cmTRUE, cmSMODE_T, 1000, 10000, 10000 );

        { 현재 위치가 (0, 0) 이라고 가정할 때, (3000, 4000) 좌표로 직선 보간 이송을 수행합니다.
        벡터 속도를 '1000' 으로 지정하였으므로 위 계산식에 의하여 Vx = 600, Vy = 800 의 속도로 이송합니다. }
        fDistList[0] := 3000;
        fDistList[1] := 4000;
        cmmIxLineStart ( nMapIdx, @fDistList );

end;
```

NAME

cmmIxSetSpeedPattern_T
- 보간(補間) 이송 속도 설정

INFORMATION

Interpolation Motion

VC++/VB

BCB/Delphi/.NET

Level 3

다소 주의

SYNOPSIS

❒VT_I4 cmmIxSetSpeedPattern_T

([in] VT_I4 MapIndex, [in] VT_I4 SpeedMode, [in] VT_R8 Vel, [in] VT_R8 AccelTime,

[in] VT_R8 DecelTime)

DESCRIPTION

cmmIxSetSpeedPattern_T 은 "기본보간제어"의 이송 속도에 대한 환경설정을 정의합니다.
cmmIxSetSpeedPattern 함수와는 다르게 가속 및 감속도 설정 시 펄스 단위가 아닌 시간 단위(msec) 로 설정 합니다.

PARAMETER

▶ MapIndex: 맵번호(Map index), 이 맵번호를 사용하기전에 먼저 cmmIxMapAxes() 함수를 통하여 해당 맵번호에 유효한 축들이 맵핑되어 있어야 합니다.

▶ SpeedMode : 속도모드를 설정합니다. 설정값은 다음과 같습니다.

Value	Meaning
0 또는 cmSMODE_C	CONSTANT 속도모드 => 가감속을 수행하지 않습니다.
1 또는 cmSMODE_T	TRAPEZOIDAL 속도모드 => 사다리꼴 가감속을 수행합니다.
2 또는 cmSMODE_S	S-CURVE 속도모드 => S-CURVE 가감속을 수행합니다.

▶ Vel : 작업속도를 PPS 단위로 설정합니다.

▶ AccTime : 가속 시간을 밀리초(msec) 단위로 설정합니다.

▶ DecTime : 감속 시간을 밀리초(msec) 단위로 설정합니다.

SEE ALSO

cmmIxSetSpeedPattern
cmmIxGetSpeedPattern_T

RETURN VALUE

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

REFERENCE

□ cmmIxSetSpeedPattern 함수 설명의 REFERENCE 참고

EXAMPLE

C/C++

```
#include “Cmmsdk.h”
#include “CmmsdkDef.h”

#define MAP0      0             // 보간 맵 정의

cmmIxMapAxes ( MAP0, cmX1_MASK | cmY1_MASK, 0 );

/* MAP0 로 설정된 축들의 보간 제어를 위해
작업 속도 1000 PPS, 가속 시간 1 초(1000 ms), 감속 시간 1 초(1000 ms) 로 설정합니다.*/
cmmIxSetSpeedPattern_T (    MAP0,               // 보간 맵 번호
                            cmSMODE_T,          // 가감속 모드
                            1000,               // 보간 작업 속도 (PPS)
                            1000,               // 가속 시간 (msec)
                            1000                // 감속 시간 (msec)
                            );

// 현재 위치가 (0, 0) 이라고 가정할 때, (3000, 4000) 좌표로 직선 보간 이송을 수행합니다.
Double fDistList[2] = {3000, 4000};
cmmIxLineStart ( MAP0, fDistList );
```

Visual Basic

```
Dim nMapIdx As Long         ‘보간 맵 정의
Dim fDistList(2) As Double   ‘ 이송 좌표값의 배열 정보

nMapIdx = 0

Call cmmIxMapAxes ( nMapIdx, cmX1_MASK Or cmY1_MASK, 0 )

‘MAP0 로 설정된 축들의 보간 제어를 위해
‘작업 속도 1000 PPS, 가속 시간 1 초(1000 ms), 감속 시간 1 초(1000 ms) 로 설정합니다.
Call cmmIxSetSpeedPattern_T ( nMapIdx, cmSMODE_T, 1000, 1000, 1000 )

‘ 현재 위치가 (0, 0) 이라고 가정할 때, (3000, 4000) 좌표로 직선 보간 이송을 수행합니다.
fDistList(0) = 3000
fDistList(1) = 4000
Call cmmIxLineStart ( nMapIdx, fDistList(0) )
```

Delphi

```
var
        nMapIdx  : LongInt;                 // 보간 맵 정의
        fDistList : Array[0..1] of Double;  // 이송 좌표값의 배열 정보

begin
        nMapIdx := 0;

        cmmIxMapAxes ( nMapIdx, cmX1_MASK or cmY1_MASK, 0 );

        { MAP0 로 설정된 축들의 보간 제어를 위해
        작업 속도 1000 PPS, 가속 시간 1 초(1000 ms), 감속 시간 1 초(1000 ms) 로 설정합니다. }
        cmmIxSetSpeedPattern_T ( nMapIdx, cmSMODE_T, 1000, 1000, 1000 );

        // 현재 위치가 (0, 0) 이라고 가정할 때, (3000, 4000) 좌표로 직선 보간 이송을 수행합니다.
        fDistList[0] := 3000;
        fDistList[1] := 4000;
        cmmIxLineStart ( nMapIdx, @fDistList );

end;
```

NAME

cmmIxGetSpeedPattern_T
- 보간(補間) 이송 속도 설정

INFORMATION

Interpolation Motion

VC++/VB

BCB/Delphi/.NET

Level 3

다소 주의

SYNOPSIS

❒VT_I4 cmmIxGetSpeedPattern_T

([in] VT_I4 MapIndex, [out] VT_PI4 SpeedMode, [out] VT_PR8 Vel, [out] VT_PR8 AccelTime,

[out] VT_PR8 DecelTime)

DESCRIPTION

cmmIxGetSpeedPattern_T 은 "기본보간제어"의 이송 속도에 대한 설정된 환경설정 값을 반환 합니다.
cmmIxGetSpeedPattern 함수와는 다르게 가속 및 감속도를 펄스 단위가 아닌 시간 단위(msec) 로 설정 된 값을 반환 합니다.

PARAMETER

▶ MapIndex: 맵번호(Map index), 이 맵번호를 사용하기전에 먼저 cmmIxMapAxes() 함수를 통하여 해당 맵번호에 유효한 축들이 맵핑되어 있어야 합니다.

▶ SpeedMode : 설정된 속도모드를 반환합니다. 반환값은 다음과 같습니다.

Value	Meaning
0 또는 cmSMODE_C	CONSTANT 속도모드 => 가감속을 수행하지 않습니다.
1 또는 cmSMODE_T	TRAPEZOIDAL 속도모드 => 사다리꼴 가감속을 수행합니다.
2 또는 cmSMODE_S	S-CURVE 속도모드 => S-CURVE 가감속을 수행합니다.

▶ Vel : 설정된 작업속도를 PPS 단위로 반환 합니다.

▶ AccTime : 설정된 가속 시간을 밀리초(msec) 단위로 반환 합니다.

▶ DecTime : 설정된 감속 시간을 밀리초(msec) 단위로 반환 합니다.

SEE ALSO

cmmIxGetSpeedPattern
cmmIxSetSpeedPattern_T

RETURN VALUE

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

REFERENCE

□ cmmIxSetSpeedPattern 함수 설명의 REFERENCE 참고

EXAMPLE

C/C++

```
#include “Cmmsdk.h”
#include “CmmsdkDef.h”

#define MAP0      0             // 보간 맵 정의

int nSpdMode;
double fVel, fAccTime, fDecTime;

cmmIxMapAxes ( MAP0, cmX1_MASK | cmY1_MASK, 0 );

/* MAP0 로 설정된 축들의 보간 제어를 위해
작업 속도 1000 PPS, 가속 시간 1 초(1000 ms), 감속 시간 1 초(1000 ms) 로 설정합니다.*/
cmmIxSetSpeedPattern_T (    MAP0,               // 보간 맵 번호
                            cmSMODE_T,          // 가감속 모드
                            1000,               // 보간 작업 속도 (PPS)
                            1000,               // 가속 시간 (msec)
                            1000                // 감속 시간 (msec)
                            );

cmmIxGetSpeedPattern_T( MAP0, &nSpdMode, &fVel, &fAccTime, &fDecTime);

// 현재 위치가 (0, 0) 이라고 가정할 때, (3000, 4000) 좌표로 직선 보간 이송을 수행합니다.
Double fDistList[2] = {3000, 4000};
cmmIxLineStart ( MAP0, fDistList );
```

Visual Basic

```
Dim nMapIdx As Long         ‘보간 맵 정의
Dim fDistList(2) As Double  ‘이송 좌표값의 배열 정보
Dim nSpdMode As Long        ‘속도 모드
Dim fVel As Double          ‘보간 작업 속도
Dim fAccTime As Double      ‘보간 가속 시간
Dim fDecTime As Double      ‘보간 감속 시간

nMapIdx = 0

Call cmmIxMapAxes ( nMapIdx, cmX1_MASK Or cmY1_MASK, 0 )

‘MAP0 로 설정된 축들의 보간 제어를 위해
‘작업 속도 1000 PPS, 가속 시간 1 초(1000 ms), 감속 시간 1 초(1000 ms) 로 설정합니다.
Call cmmIxSetSpeedPattern_T ( nMapIdx, cmSMODE_T, 1000, 1000, 1000 )

Call cmmIxGetSpeedPattern_T( MAP0, nSpdMode, fVel, fAccTime, fDecTime);

‘ 현재 위치가 (0, 0) 이라고 가정할 때, (3000, 4000) 좌표로 직선 보간 이송을 수행합니다.
fDistList(0) = 3000
fDistList(1) = 4000
Call cmmIxLineStart ( nMapIdx, fDistList(0) )
```

Delphi

```
var
        nMapIdx  : LongInt;             // 보간 맵 정의
        nSpdMode : LongInt;             // 보간 이송 속도 모드
        fVel : Double;                  // 보간 작업 속도
        fAccTime : Double;              // 보간 가속 시간
        fDecTime : Double;              // 보간 감속 시간
```

```
        fDistList : Array[0..1] of Double;   // 이송 좌표값의 배열 정보

begin
        nMapIdx := 0;

        cmmIxMapAxes ( nMapIdx, cmX1_MASK or cmY1_MASK, 0 );

        { MAP0 로 설정된 축들의 보간 제어를 위해
        작업 속도 1000 PPS, 가속 시간 1 초(1000 ms), 감속 시간 1 초(1000 ms) 로 설정합니다. }
        cmmIxSetSpeedPattern_T ( nMapIdx, cmSMODE_T, 1000, 1000, 1000 );

        cmmIxGetSpeedPattern_T( MAP0, @nSpdMode, @fVel, @fAccTime, @fDecTime);

        // 현재 위치가 (0, 0) 이라고 가정할 때, (3000, 4000) 좌표로 직선 보간 이송을 수행합니다.
        fDistList[0] := 3000;
        fDistList[1] := 4000;
        cmmIxLineStart ( nMapIdx, @fDistList );

end;
```

NAME

cmmIxLine
cmmIxLineStart
– 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 cmmIxLine ([in] VT_I4 MapIndex, [in] VT_PR8 DistList, [in] VT_I4 IsBlocking)

❒ VT_I4 cmmIxLineStart ([in] VT_I4 MapIndex, [in] VT_PR8 DistList)

DESCRIPTION

이 함수는 현재 위치로부터의 상대 좌표로의 직선 보간 이동을 수행합니다. cmmIxLine() 함수는 모션이 완료되기 전까지 반환되지 않으며, cmmIxLineStart() 함수는 모션을 시작시킨 후에 바로 반환됩니다.

PARAMETER

▶ MapIndex : 맵번호(Map index), 이 맵번호를 사용하기전에 먼저 cmnIxMapAxes() 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 cmnIxMapAxes() 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
cmFALSE	블록(Blocking)을 하지 않습니다. 따라서 해당 모션이 완료되는 동안에도 윈도우 이벤트를 처리합니다.
cmTRUE	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
cmERR_NONE	수행 성공

SEE ALSO

cmmIxLineTo, cmmIxLineToStart

REFERENCE

□ 논리적 거리 단위는 cmmCfgSetUnitDist() 함수에 의해 결정됩니다.

□ cmmIxLineStart() 함수를 사용하는 경우에는 cmmIxIsDone() 함수나 cmmIxWaitDone() 함수를 사용하여 모션의 완료를 확인(確認)할 수 있습니다.

□ cmmIxLine() 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 방식3의 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 "Cmmsdk.h"
#include "CmmsdkDef.h"

/*************************************************************
* OnProgramInitial : 이 함수는 가상의 함수로서 프로그램 초기화 루틴이
* 적용되는 부분을 의미합니다.
*************************************************************/
void OnProgramInitial()
{
	long m_nNumAxes;
	cmmLoadDll();
	if(cmmGnDeviceLoad(cmTRUE, &m_nNumAxes) != cmERR_NONE)
	{
		//Handle 은 사용자가 생성한 폼의 핸들 값입니다.
		cmmErrShowLast(Handle);
		return;
	}
}

/*************************************************************
* OnSetSpeed : 이 함수는 속도설정의 변경이 필요할 때
* 호출되는 가상의 함수 입니다.
*************************************************************/
#define MAP0 0 //맵번호 (0)
void OnSetSpeed()
{
	cmmIxMapAxes(MAP0, cmX1_MASK | cmY1_MASK, 0 );
	//또는 cmmIxMapAxes(MAP0, 0x3, 0x0);
	//보간 이동할 축들의 기본속도를 설정합니다.
	cmmCfgSetSetSpeedPattern(cmX1, cmSMODE_T, 1000, 5000, 5000);
	cmmCfgSetSetSpeedPattern(cmY1, cmSMODE_T, 1000, 5000, 5000);
}

/*************************************************************
* 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%,
```

3 이것을 순수 우리말로 옮기면 이벤트 반응형 운영체제라고 할 수 있습니다.

```
        //가속도의 70%, 감속도의 70%로 설정 합니다.
        cmmIxSetSpeedPattern(MAP0, cmFALSE, cmSMODE_T, 100, 70, 70 );
        cmmIxLine(MAP0, fDistList, cmFALSE);
        fDistList[0] = -1000; fDistList[1] = -2000;
        cmmIxLine(MAP0, fDistList, cmFALSE);

        //cmmIxLineStart() 함수를 사용하는 경우에는 다음과 같이 코드를 작성합니다.
        //double fDistList[2] = {1000, 2000};
        //cmmIxSetSpeedPattern(MAP0, cmFALSE, cmSMODE_T, 100, 70,
        //70 );
        //cmmIxLineStart(MAP0, fDistList);
        //cmmIxWaitDone(MAP0, cmFALSE);
        //fDistList[0] = -1000; fDistList[1] = -2000;
        //cmmIxLineStart(MAP0, fDistList);
        //cmmIxWaitDone(MAP0, cmFALSE);
}
```

Visual Basic

```
‘맵 번호 MAP0 은 이미 선언되어 있다고 가정함.
‘================================================================
‘cmmGnDeviceLoad 함수로 장치를 초기화 합니다.
‘================================================================

Private Sub Form_Load()
   Dim nTotalAxis As Long
   Dim IRetVal As Long
'================================================================
 ' cmmGnDeviceLoad 함수로 장치를 초기화합니다.
  IRetVal = cmmGnDeviceLoad(cmTRUE, nTotalAxis)

  If IRetVal <> cmERR_NONE Then
     MsgBox ("cmmGnDeviceLoad 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

   ' ============================================================
   ' cmmIxMapAxes 함수는 다음과 같은 인자를 필요로 합니다.   '
   ' cmmIxMapAxes(맵번호, 비트(Bit)를 통한 맵구성 #1, 비트(Bit)를  통한 맵구성 #2
   ' ============================================================

   nRetVal = cmmIxMapAxes(MAP0, &H3, &H0)
   ' //////////////////////////////////////////////////////////
   If cmmIxSetSpeedPattern(MAP0, cmFALSE, cmSMODE_S, 100, 100, _
                       100) <> cmERR_NONE Then

      ' 아래와 같은 커미조아 CMMSDK 전용 에러 표시 함수를
      ‘ 사용할 수 있습니다.
      ' cmmErrShowLast (IxLine.Hwnd)
      ' ------------------------------------------------------
      MsgBox ("cmmIxSetSpeedPattern has been failed")
   End If

   ' cmmIxLine 함수를 통해 선형 보간을 수행합니다. 인자는 순서대로, 맵번호,
         ‘ 거리정보를 가지고 있는 배열, 블록 여부 입니다.
   nRetVal = cmmIxLine(MAP0, DistanceList(0), cmFALSE)
End Sub
```

```
Private Sub CfgSpeed(nTotalAxis As Long)

  Dim i As Integer
   '=================================================================
   ' 이 함수에서 cmmCfgSetSpeedPattern 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 cmmCfgSetSpeedPattern(0, cmSMODE_S, 1000, 2000, 2000)
   Next
End Sub
```

Delphi

```
Const MAPINDEX = 0;
// * 이 함수는 폼이 생성될때 이벤트에 의해 불려지며 , 장치를 로드하는 함수입
// * 니다.

procedure OnCreate();
var
        g_nAxis : LongInt;
begin
        // Load CMMSDK(DLL) Library
        if ( cmmGnDeviceLoad(cmTRUE,@g_nAxis) <> cmERR_NONE ) then
        begin
                // 마지막에 발생한 에러를 화면에 표시합니다.
                // 함수 인자로는 Form 의 Handle 이 전달됩니다.
                cmmErrShowLast(Form1.Handle);
                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 := cmSMODE_S;

        // cmX1 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.
        cmmCfgSetSpeedPattern(
        cmX1,           // 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.
        cmmCfgSetSpeedPattern(
        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

      AxisList : Array[0..1] of LongInt;
      fDistanceList : Array[0..1] of Double;

begin
      // cmmIxMapAxes 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.
      cmmIxMapAxes(MAPINDEX,$3,$0);

      // ------------------------------------------------------
      // 보간제어의 속도 모드에 대해서 다음과 같이 설정할 수 있습니다.

      // 아래는 Vecter Speed 모드로 동작하는 예제입니다.
      //cmmIxSetSpeedPattern(MAPINDEX, cmTRUE, cmSMODE_S, 1000,  2000, 2000);

      // 아래는 Master Speed 모드로 동작하는 예제입니다.
      // Master Speed 설정
      // 가속도 : 100%
      // 감속도 : 100%
      // 작업속도 : 100%

      cmmIxSetSpeedPattern(MAPINDEX, cmFALSE, cmSMODE_S,
                  100,100,100);

      // ------------------------------------------------------

      AxisList[0] := cmX1;
      AxisList[1] := cmY1;

      fDistanceList[0] := 1000;
      fDistanceList[1] := 1000;

      // 보간 대상 그룹을 통해 실제 보간 작업을 수행합니다.
      // 이때 함수의 이름을 통해
      // cmmIxLine 은 상대 좌표 보간을 의마힙니다.
      // 직선 보간이 완료된 후 반환됩니다.
      // cmmIxLineTo 는 절대 좌표 보간을 의미합니다.
      // 직선 보간이 완료된 후 반환됩니다.
      // cmmIxLineStart / cmmIxLineToStart 는 각각 상대좌표와 절대 좌표를
      // 목적 위치로 하여,
      // 직선 보간이 시작되자 마자 바로 반환합니다.
      cmmIxLine(MAPINDEX, @fDistanceList, cmFALSE);

end;
```

NAME

cmmIxLineTo
cmmIxLineToStart
– 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 cmmIxLineTo ([in] VT_I4 MapIndex, [in] VT_PR8 PosList, [in] VT_I4 IsBlocking)

❒ VT_I4 cmmIxLineToStart ([in] VT_I4 MapIndex, [in] VT_PR8 PosList)

DESCRIPTION

이 함수는 절대 좌표로의 직선 보간 이동을 수행합니다. cmmIxLineTo() 함수는 모션이 완료되기 전까지 반환되지 않으며, cmmIxLineToStart() 함수는 모션을 시작시킨 후에 바로 반환됩니다.

PARAMETER

▶ MapIndex : 맵번호(Map index), 이 맵번호를 사용하기전에 먼저 cmmIxMapAxes() 함수를 통하여 해당 맵번호에 유효한 축들이 맵핑되어 있어야 합니다.

▶ PosList : 이동할 목표 절대좌표값(각 축의 절대좌표값)의 배열 주소. 이 배열의 크기는 cmmIxMapAxes() 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
cmFALSE	블록(Blocking)을 하지 않습니다. 따라서 해당 모션이 완료되는 동안에도 윈도우 이벤트를 처리합니다.
cmTRUE	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
cmERR_NONE	수행 성공

SEE ALSO

□ 논리적 거리 단위는 cmmCfgSetUnitDist() 함수에 의해 결정됩니다.

□ cmmIxLineToStart() 함수를 사용하는 경우에는 cmmIxIsDone() 함수나 cmmIxWaitDone() 함수를 사용하여 모션의 완료를 확인(確認)할 수 있습니다.

□ cmmIxLineTo() 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 "Cmmsdk.h"
#include "CmmsdkDef.h"

/****************************************************************
 * OnProgramInitial : 이 함수는 가상의 함수로서 프로그램 초기화 루틴이
 * 적용되는 부분을 의미합니다.
****************************************************************/
void OnProgramInitial()
{

	long m_nNumAxes;

	cmmLoadDll();
	if(cmmGnDeviceLoad(cmTRUE, &m_nNumAxes) != cmERR_NONE)
	{
		//Handle 은 사용자가 생성한 폼의 핸들 값입니다.
		cmmErrShowLast(Handle);
		return;
	}
}

/****************************************************************
* OnSetSpeed : 이 함수는 속도설정의 변경이 필요할 때
* 호출되는 가상의 함수 입니다.
****************************************************************/
#define MAP0 0 //맵번호 (0)
void OnSetSpeed()
{
	cmmIxMapAxes(MAP0, cmX1_MASK | cmY1_MASK, 0 );
	//또는 cmmIxMapAxes(MAP0, 0x3, 0x0);
	//보간 이동할 축들의 기본속도를 설정합니다.
	cmmCfgSetSetSpeedPattern(cmX1, cmSMODE_T, 1000, 5000, 5000);
	cmmCfgSetSetSpeedPattern(cmY1, cmSMODE_T, 1000, 5000, 5000);
}

/****************************************************************
* DoMotion : 작업명령시에 호출되는 가상의 함수 입니다.
* 이 함수는 X1, Y1 축에 대하여 절대좌표 (1000, 2000)으로 이동한 후
* 다시 (0, 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.
	cmmIxSetSpeedPattern(MAP0, cmFALSE, cmSMODE_T, 100, 70, 70 );
	cmmIxLineTo(MAP0, fPosList, cmFALSE);
	fPosList[0] = 0; fPosList[1] = 0;
	cmmIxLineTo(MAP0, fPosList, cmFALSE);

	//cmmIxLineToStart() 함수를 사용하는 경우에는 다음과 같이 코드를
```

```
        //작성합니다.
        //double fPosList[2] = {1000, 2000};
        //cmmIxSetSpeedPattern(MAP0, cmFALSE, cmSMODE_T, 100, 70,
        //70 );
        //cmmIxLineToStart(MAP0, fPosList);
        //cmmIxWaitDone(MAP0, cmFALSE);
        //fPosList[0] = -1000; fPosList[1] = -2000;
        //cmmIxLineToStart(MAP0, fPosList);
        //cmmIxWaitDone(MAP0, cmFALSE);
}
```

```
Visual Basic

‘맵 번호 MAP0 은 이미 선언되어 있다고 가정함.
‘=============================================================
‘cmmGnDeviceLoad 함수로 장치를 초기화 합니다.
‘=============================================================

Private Sub Form_Load()

   Dim nTotalAxis As Long
   Dim IRetVal As Long
'=============================================================
 ' cmmGnDeviceLoad 함수로 장치를 초기화합니다.
  IRetVal = cmmGnDeviceLoad(cmTRUE, nTotalAxis)

  If IRetVal <> cmERR_NONE Then
     MsgBox ("cmmGnDeviceLoad 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

   ' ============================================================
   ' cmmIxMapAxes 함수는 다음과 같은 인자를 필요로 합니다.   '
   ' cmmIxMapAxes(맵번호, 비트(Bit)를 통한 맵구성 #1, 비트(Bit)를  통한 맵구성 #2

         nRetVal = cmmIxMapAxes(MAP0, &H3, &H0)
         If cmmIxSetSpeedPattern(MAP0, cmFALSE, cmSMODE_S, 100, 100, _
                  100) <> cmERR_NONE Then
         MsgBox ("cmmIxSetSpeedPattern has been failed")
   End If

        nRetVal = cmmIxLineTo(MAP0, DistanceList(0), cmFALSE)
End Sub

Private Sub btnStop_Click()

   Dim nRetVal As Long

   ' cmmIxStop 을 통해 보간제어를 정지(停止)합니다.
   ' 각 자세한 함수인자는 매뉴얼을 참조해주시기 바랍니다.
   nRetVal = cmmIxStop(MAP0, cmTRUE, cmFALSE)

   If nRetVal <> cmERR_NONE Then
      ‘ Handle 은 사용자가 생성한 폼의 핸들 값입니다.
      cmmErrShowLast (Handle)
   End If

End Sub
```

```
Private Sub CfgSpeed(nTotalAxis As Long)

   Dim i As Integer
   Dim nTotalAxis As Integer
   '============================================================
   ' 이 함수에서 cmmCfgSetSpeedPattern 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 cmmCfgSetSpeedPattern(i, cmSMODE_S, 1000, 2000, 2000)
   Next

End Sub
```

```
Delphi

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_nAxis : LongInt;
begin

        // Load CMMSDK(DLL) Library
        if ( cmmGnDeviceLoad(cmTRUE,@g_nAxis) <> cmERR_NONE ) then
        begin
                // 마지막에 발생한 에러를 화면에 표시합니다.
                // 함수 인자로는 Form 의 Handle 이 전달됩니다.
                cmmErrShowLast(Form1.Handle);
                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 := cmSMODE_S;

        // cmX1 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.
```

```
        cmmCfgSetSpeedPattern(
        cmX1,           // 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.
        cmmCfgSetSpeedPattern(
        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

      AxisList : Array[0..1] of LongInt;
      fDistanceList : Array[0..1] of Double;
begin

      // cmmIxMapAxes 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.
      cmmIxMapAxes(MAPINDEX,$3,$0);

      // ------------------------------------------------------
      // 보간제어의 속도 모드에 대해서 다음과 같이 설정할 수 있습니다.

      // 아래는 Vecter Speed 모드로 동작하는 예제입니다.
      //cmmIxSetSpeedPattern(MAPINDEX, cmTRUE, cmSMODE_S, 1000, 2000, 2000);

      // 아래는 Master Speed 모드로 동작하는 예제입니다.
      cmmIxSetSpeedPattern(MAPINDEX, cmFALSE, cmSMODE_S,
                  100,100,100);
      // Master Speed 설정
      // 가속도 : 100%
      // 감속도 : 100%
      // 작업속도 : 100%

      // ------------------------------------------------------

      AxisList[0] := cmX1;
      AxisList[1] := cmY1;

      fDistanceList[0] := 1000;
      fDistanceList[1] := 1000;

      // 보간 대상 그룹을 통해 실제 보간 작업을 수행합니다.
      // 이때 함수의 이름을 통해
      // cmmIxLine 은 상대 좌표 보간을 의마힙니다.
      // 직선 보간이 완료된 후 반환됩니다.
      // cmmIxLineTo 는 절대 좌표 보간을 의미합니다.
      // 직선 보간이 완료된 후 반환됩니다.
      // cmmIxLineStart / cmmIxLineToStart 는 각각 상대좌표와 절대 좌표를
```

```
    // 목적 위치로 하여,
    // 직선 보간이 시작되자 마자 바로 반환합니다.

                If cmmIxLineTo(MAPINDEX, @fDistanceList[0], cmFALSE) <> cmERR_NONE then begin
                        //Handle 은 사용자가 생성한 폼의 핸들 값입니다.
                        cmmErrShowLast(Handle);
                end;
end;

// * Description :
// *
// *  현재 수행되고 있는 모션 동작에 대해서 감속 후 정지(停止) 합니다.

procedure btnStopClick();
begin
    cmmIxStop(MAPINDEX,cmTRUE, cmFALSE);
end;
```

NAME

cmmIxArcA
cmmIxArcAStart
– 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 cmmIxArcA

([in] VT_I4 MapIndex, [in] VT_R8 XCentOffset, [in] VT_R8 YCentOffset, [in] VT_R8 EndAngle,

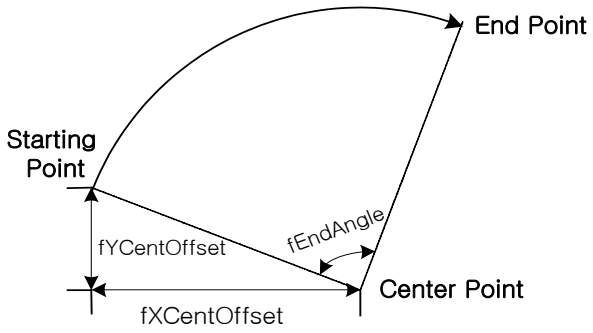
[in] VT_I4 IsBlocking)

❒ VT_I4 cmmIxArcAStart

([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. cmmIxArcA() 함수는 모션이 완료되기 전까지 반환되지 않으며, cmmIxArcAStart() 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.



PARAMETER

▶ MapIndex : 맵번호(Map index), 이 맵번호를 사용하기전에 먼저 cmmIxMapAxes() 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
cmFALSE	블록(Blocking)을 하지 않습니다. 따라서 해당 모션이 완료되는 동안에도 윈도우 이벤트를 처리합니다.
cmTRUE	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
cmERR_NONE	수행 성공

SEE ALSO

□ 논리적 거리 단위는 cmmCfgSetUnitDist() 함수에 의해 결정됩니다.

□ cmmIxArcAStart() 함수를 사용하는 경우에는 cmmIxIsDone() 함수나 cmmIxWaitDone() 함수를 사용하여 모션의 완료를 확인(確認)할 수 있습니다.

□ cmmIxArcA() 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.

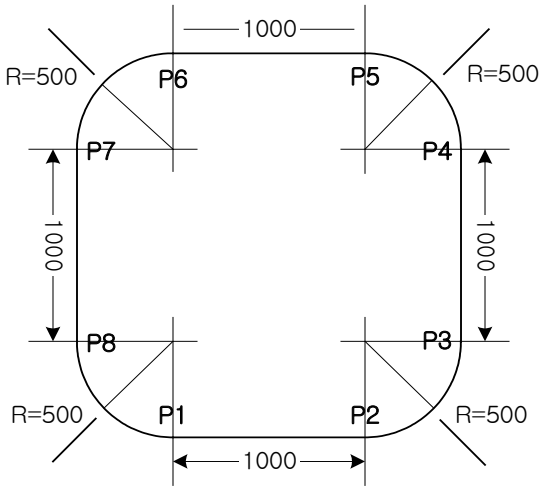
□ cmmIxArcA() 함수를 사용하는 경우에는 INP 입력신호가 Enable 로 설정되었으면 Command 펄스 출력이 완료되어도 INP 입력이 ON 이 되기 전까지는 모션이 완료되지 않은 것으로 간주되어 반환되지 않습니다.
스탭 드라이브는 INP 출력이 없는 경우가 일반적인데, 고객(顧客)님의 부주의나 잘못된 설정으로 INP 입력에 대한 설정이 Enable 로 되어 있을 경우 INP 입력이 스탭 드라이브를 통해 발생하지 않는 이유 때문에 모션 완료가 되지 않는 경우가 발생할 수 있습니다.
고객(顧客) 여러분들께서는 스탭 드라이브 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 "Cmmsdk.h"
#include "CmmsdkDef.h"

/*************************************************************
* OnProgramInitial : 이 함수는 가상의 함수로서 프로그램 초기화 루틴이
* 적용되는 부분을 의미합니다.
*************************************************************/
void OnProgramInitial()
{

	long m_nNumAxes;

	cmmLoadDll();

	if(cmmGnDeviceLoad(cmTRUE, &m_nNumAxes) != cmERR_NONE)
	{
		//Handle 은 사용자가 생성한 폼의 핸들 값입니다.
		cmmErrShowLast(Handle);
		return;
	}
}

/*************************************************************
* OnSetSpeed : 이 함수는 속도설정의 변경이 필요할 때
* 호출되는 가상의 함수 입니다.
*************************************************************/
#define MAP0 0 //맵번호 (0)
void OnSetSpeed()
{
	cmmIxMapAxes(MAP0, cmX1_MASK | cmY1_MASK, 0 );
	//또는 cmmIxMapAxes(MAP0, 0x3, 0x0);
	//보간 이동할 축들의 기본속도를 설정합니다.
	cmmCfgSetSpeedPattern(cmX1, cmSMODE_T, 1000, 5000, 5000);
	cmmCfgSetSpeedPattern(cmY1, cmSMODE_T, 1000, 5000, 5000);
}

/*************************************************************
* 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.
	cmmIxSetSpeedPattern(MAP0, cmFALSE, cmSMODE_T, 100, 70, 70 );

	// Move from P1 to P2 //
```

```
        fDistList[0]=1000; fDistList[1]=0;
        cmmIxLine(MAP0, fDistList, cmFALSE);

        // Move from P2 to P3 //
        cmmIxArcA(MAP0, 0, 500, 90, cmFALSE);

        // Move from P3 to P4 //
        fDistList[0]=0; fDistList[1]=1000;
        cmmIxLine(MAP0, fDistList, cmFALSE);

        // Move from P4 to P5 //
        cmmIxArcA(MAP0, -500, 0, 90, cmFALSE);

        // Move from P5 to P6 //
        fDistList[0]=-1000; fDistList[1]=0;
        cmmIxLine(MAP0, fDistList, cmFALSE);

        // Move from P6 to P7 //
        cmmIxArcA(MAP0, 0, -500, 90, cmFALSE);

        // Move from P7 to P8 //
        fDistList[0]=0; fDistList[1]=-1000;
        cmmIxLine(MAP0, fDistList, cmFALSE);

        // Move from P8 to P1 //
        cmmIxArcA(MAP0, 500, 0, 90, cmFALSE);
}
```

```
Visual Basic
‘맵 번호 MAP0 은 이미 선언되어 있다고 가정함.
‘================================================================
‘cmmGnDeviceLoad 함수로 장치를 초기화 합니다.
‘================================================================

Private Sub Form_Load()
   Dim nTotalAxis As Long
   Dim IRetVal As Long
'================================================================
 ' cmmGnDeviceLoad 함수로 장치를 초기화합니다.
  IRetVal = cmmGnDeviceLoad(cmTRUE, nTotalAxis)

  If IRetVal <> cmERR_NONE Then
     MsgBox ("cmmGnDeviceLoad has been failed")
  End If  '================================================================
End Sub

Private Sub CfgSpeed(nTotalAxis As Long)

   Dim i As Integer

   '================================================================
   ' 이 함수에서 cmmCfgSetSpeedPattern 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 cmmCfgSetSpeedPattern(i, cmSMODE_S, 1000, 2000, 2000)
   Next

End Sub

Private Sub btnMove_Click()

   Dim nRetVal As Long
   Dim dXCentOfs As Double
   Dim dYCentOfs As Double
   Dim dAngle As Double
```

```
nRetVal = cmmIxMapAxes(MAP0, &H3, &H0)

If cmmIxSetSpeedPattern(MAP0, False, cmSMODE_S, 100, 100, 100) <> cmERR_NONE Then

        MsgBox ("cmmIxSetSpeedPattern has been failed")
 End If

   dXCentOfs = 5000
   dYCentOfs = 5000
   dAngle = 90

   nRetVal = cmmIxArcA(MAP0, dXCentOfs, dYCentOfs, dAngle, cmFALSE)


End Sub
```

```
Delphi

const
  g_nTargetAxis = 2;
  MAPINDEX = 0;

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

procedure OnCreate();
var
        g_nAxis : LongInt;
begin

        // Load CMMSDK(DLL) Library
        if ( cmmGnDeviceLoad(cmTRUE,@g_nAxis) <> cmERR_NONE ) then
        begin
                // 마지막에 발생한 에러를 화면에 표시합니다.
                // 함수 인자로는 Form 의 Handle 이 전달됩니다.
                cmmErrShowLast(Form1.Handle);
                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);
      // cmmIxMapAxes 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.
      cmmIxMapAxes(MAPINDEX,$3,$0);

      dXCentOfs := 1000;
      dYCentOfs := 1000;
      dAngle       := 90;

      // 기준 속도란 cmmCfgSetSpeedPattern 함수를 통해 설정된 속도를 의미하며,
      // 아래의 cmmIxSetSpeedPattern 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.
    //cmmIxSetSpeedPattern(MAPINDEX, cmTRUE, cmSMODE_S, 1000,  2000, 2000);

    // 아래는 Master Speed 모드로 동작하는 예제입니다.
    cmmIxSetSpeedPattern(MAPINDEX, cmFALSE, cmSMODE_S, fWorkSpeedRatio,fAccelSpeedRatio,fDecelSpeedRatio);

    // ------------------------------------------------------

    // 원호 보간을 수행합니다.

    // cmmIxArcA 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. cmmIxArcA : 상대 거리를 목표로 하여, 원호 보간이 완료된 상태에서 함수가 반환됩니다.
    // 2. cmmIxArcAStart : 상대 거리를 목표로 하여,  원호 보간이 시작된 후 바로 반환됩니다.
    // 3. cmmIxArcATo : 절대 거리를 목표로 하여, 원호 보간이 완료된 상태에서 함수가 반환됩니다.
    // 4. cmmIxArcAToStart : 절대 거리를 목표로 하여,  원호 보간이  시작된 후 바로 반환됩니다.

    cmmIxArcA( MAPINDEX, dXCentOfs, dYCentOfs, dAngle, cmFALSE);
end;
```

NAME

cmmIxArcATo
cmmIxArcAToStart
– 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 cmmIxArcATo

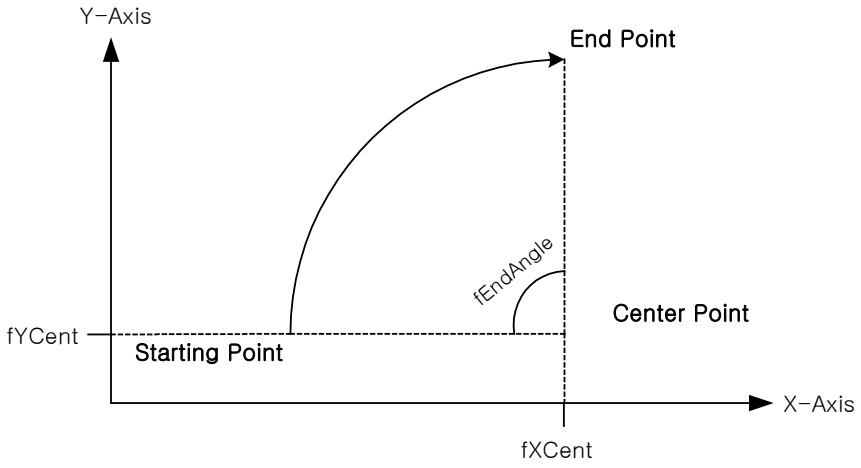
([in] VT_I4 MapIndex, [in] VT_R8 XCent, [in] VT_R8 YCent, [in] VT_R8 EndAngle, [in] VT_I4 IsBlocking)

❒ VT_I4 cmmIxArcAToStart

([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. cmmIxArcATo() 함수는 모션이 완료되기 전까지 반환되지 않으며, cmmIxArcAToStart() 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.



PARAMETER

▶ MapIndex : 맵번호(Map index), 이 맵번호를 사용하기전에 먼저 cmmIxMapAxes() 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
cmFALSE	블록(Blocking)을 하지 않습니다. 따라서 해당 모션이 완료되는 동안에도 윈도우 이벤트를 처리합니다.
cmTRUE	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
cmERR_NONE	수행 성공

SEE ALSO

☐ 논리적 거리 단위는 cmmCfgSetUnitDist() 함수에 의해 결정됩니다.

☐ cmmIxArcAToStart() 함수를 사용하는 경우에는 IxIsDone() 함수나 IxWaitDone() 함수를 사용하여 모션의 완료를 확인(確認)할 수 있습니다.

☐ cmmIxArcATo() 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.

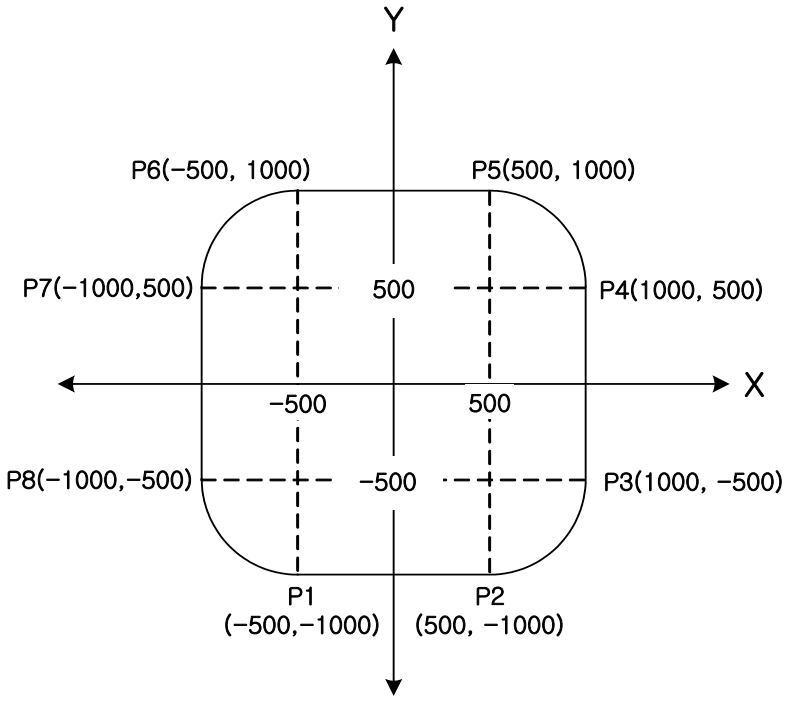
☐ cmmIxArcATo() 함수를 사용하는 경우에는 INP 입력신호가 Enable 로 설정되었으면 Command 펄스 출력이 완료되어도 INP 입력이 ON 이 되기 전까지는 모션이 완료되지 않은 것으로 간주되어 반환되지 않습니다 스탭 드라이브는 INP 출력이 없는 경우가 일반적인데, 고객(顧客)님의 부주의나 잘못된 설정으로 INP 입력에 대한 설정이 Enable 로 되어 있을 경우 INP 입력이 스탭 드라이브를 통해 발생하지 않는 이유 때문에 모션 완료가 되지 않는 경우가 발생할 수 있습니다.
고객(顧客) 여러분들께서는 스탭 드라이브 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 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 "Cmmsdk.h"
#include "CmmsdkDef.h"

/****************************************************************
* OnProgramInitial : 이 함수는 가상의 함수로서 프로그램 초기화 루틴이
* 적용되는 부분을 의미합니다.
****************************************************************/
void OnProgramInitial()
{

	long m_nNumAxes;

	cmmLoadDll();

	if(cmmGnDeviceLoad(cmTRUE, &m_nNumAxes) != cmERR_NONE)
	{
		//Handle 은 사용자가 생성한 폼의 핸들 값입니다.
		cmmErrShowLast(Handle);
		return;
	}
}

/****************************************************************
* OnSetSpeed : 이 함수는 속도설정의 변경이 필요할 때
* 호출되는 가상의 함수 입니다.
****************************************************************/
#define MAP0 0 //맵번호 (0)
void OnSetSpeed()
{
	cmmIxMapAxes(MAP0, cmX1_MASK | cmY1_MASK, 0 );

	//또는 cmmIxMapAxes(MAP0, 0x3, 0x0);
	//보간 이동할 축들의 기본속도를 설정합니다.
	cmmCfgSetSpeedPattern(cmX1, cmSMODE_T, 1000, 5000, 5000);
	cmmCfgSetSpeedPattern(cmY1, cmSMODE_T, 1000, 5000, 5000);
}

/****************************************************************
```

```
* 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.
	cmmIxSetSpeedPattern(MAP0, cmFALSE, cmSMODE_T, 100, 70, 70 );

	// Move from P1 to P2 //
	fPosList[0]=500; fPosList[1]=-1000;
	cmmIxLineTo(MAP0, fPosList, cmFALSE);

	// Move from P2 to P3 //
	cmmIxArcATo(MAP0, 500, -500, 90, cmFALSE);

	// Move from P3 to P4 //
	fPosList[0]=1000; fPosList[1]=500;
	cmmIxLineTo(MAP0, fPosList, cmFALSE);

	// Move from P4 to P5 //
	cmmIxArcATo(MAP0, 500, 500, 90, cmFALSE);

	// Move from P5 to P6 //
	fPosList[0]=-500; fPosList[1]=1000;
	cmmIxLineTo(MAP0, fPosList, cmFALSE);

	// Move from P6 to P7 //
	cmmIxArcATo(MAP0, -500, 500, 90, cmFALSE);

	// Move from P7 to P8 //
	fPosList[0]=-1000; fPosList[1]=-500;
	cmmIxLineTo(MAP0, fPosList, cmFALSE);

	// Move from P8 to P1 //
	cmmIxArcATo(MAP0, -500, -500, 90, cmFALSE);
}
```

```
Visual Basic
‘맵 번호 MAP0 은 이미 선언되어 있다고 가정함.
'/************************************************************
'* OnSetSpeed : 이 함수는 속도설정의 변경이 필요할 때
'* 호출되는 가상의 함수 입니다.
'************************************************************/
Private Sub OnSetSpeed()

	Call cmmIxMapAxes(MAP0, &H3, 0) '//&H3 is cmX1 | cmY1

	'보간 이동할 축들의 기본속도를 설정합니다.
	Call cmmCfgSetSpeedPattern(cmX1, cmSMODE_T, 1000, 5000, 5000)
	Call cmmCfgSetSpeedPattern(cmY1, cmSMODE_T, 1000, 5000, 5000)

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 cmmIxSetSpeedPattern(MAP0, cmFALSE, cmSMODE_T, 100, 70, 70)

	'// Move from P1 to P2 //
	fPosList(0) = 500
	fPosList(1) = -1000
```

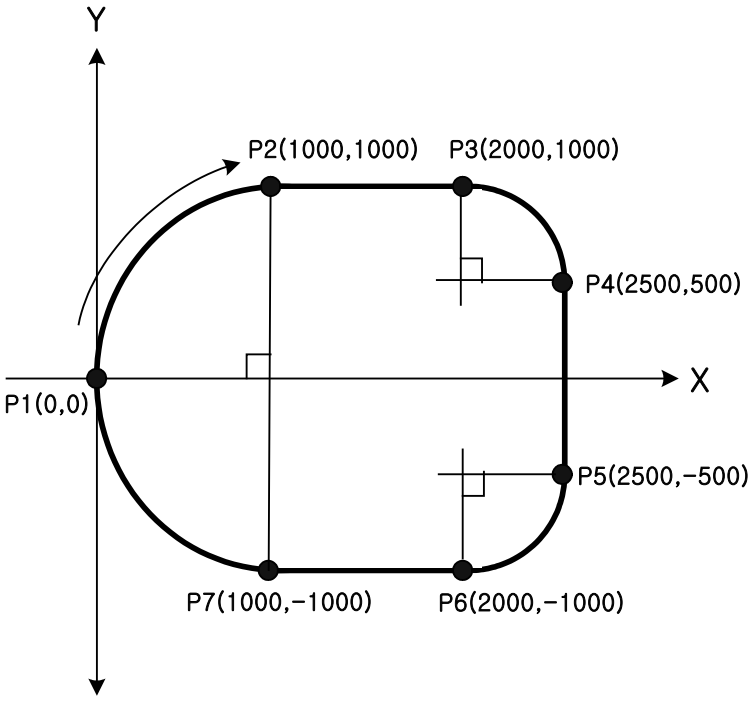
```
        Call cmmIxLineTo(MAP0, fPosList(0), cmFALSE)

        '// Move from P2 to P3 //
        Call cmmIxArcATo(MAP0, 500, -500, 90, cmFALSE)

        '// Move from P3 to P4 //
        fPosList(0) = 1000
        fPosList(1) = 500
        Call cmmIxLineTo(MAP0, fPosList(0), cmFALSE)

        '// Move from P4 to P5 //
        Call cmmIxArcATo(MAP0, 500, 500, 90, cmFALSE)

        '// Move from P5 to P6 //
        fPosList(0) = -500
        fPosList(1) = 1000
        Call cmmIxLineTo(MAP0, fPosList(0), cmFALSE)

        '// Move from P6 to P7 //
        Call cmmIxArcATo(MAP0, -500, 500, 90, cmFALSE)

        '// Move from P7 to P8 //
        fPosList(0) = -1000
        fPosList(1) = -500
        Call cmmIxLineTo(MAP0, fPosList(0), cmFALSE)

        '// Move from P8 to P1 //
        Call cmmIxArcATo(MAP0, -500, -500, 90, cmFALSE)

End Sub
```

```
Delphi

var
        nIxMap : LongInt;       // 보간 맵 설정
begin
        nIxMap := 0;
end;

procedure OnSetIxConfig ();
begin
        // 0 번 축과 1 번 축을 보간 맵으로 설정합니다.
        cmmIxMapAxes ( nIxMap, cmX1_MASK or cmY1_MASK, 0 );

        // 각 축의 속도를 설정합니다.
        cmmCfgSetSpeedPattern (cmX1, cmSMODE_T, 1000, 5000, 5000 );
        cmmCfgSetSpeedPattern (cmY1, cmSMODE_T, 1000, 5000, 5000 );

end;


procedure OnIxArcATo_Move ();
var
        fPosList : Array[0..1] of Double;

begin
        // 맵에 포함된 축의 속도 비율을 설정합니다.
        cmmCfgSetSpeedPattern ( nIxMap, cmSMODE_T, 100, 70, 70 );

        // P1 에서 P2 로 이동합니다.
        fPosList[0] := 500;
        fPosList[1] := -1000;
        cmmIxLineTo ( nIxMap, @fPosList, cmFALSE );

        // P2 에서 P3 로 이동합니다.
        cmmIxArcATo ( nIxMap, 500, -500, 90, cmFALSE );

        // P3 에서 P4 로 이동합니다.
        fPosList[0] := 1000;
        fPosList[1] := 500;
```

```
        cmmIxLineTo ( nIxMap, @fPosList, cmFALSE );

        // P4 에서 P5 로 이동합니다.
        cmmIxArcATo ( nIxMap, 500, 500, 90, cmFALSE );

        // P5 에서 P6 로 이동합니다.
        fPosList[0] := -500;
        fPosList[1] := 1000;
        cmmIxLineTo ( nIxMap, @fPosList, cmFALSE );

        // P6 에서 P7 로 이동합니다.
        cmmIxArcATo ( nIxMap, -500, 500, 90, cmFALSE );

        // P7 에서 P8 로 이동합니다.
        fPosList[0] := -1000;
        fPosList[1] := -500;
        cmmIxLineTo ( nIxMap, @fPosList, cmFALSE );

        // P8 에서 P1 로 이동합니다.
        cmmIxArcATo ( nIxMap, -500, -500, 90, cmFALSE );

end;
```

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 "Cmmsdk.h"
#include "CmmsdkDef.h"

/************************************************************
* OnProgramInitial : 이 함수는 가상의 함수로서 프로그램 초기화 루틴이
* 적용되는 부분을 의미합니다.
************************************************************/
void OnProgramInitial()
```

```
{
	long m_nNumAxes;

	cmmLoadDll();
	if(cmmGnDeviceLoad(cmTRUE, &m_nNumAxes) != cmERR_NONE)
	{
		//Handle 은 사용자가 생성한 폼의 핸들 값입니다.
		cmmErrShowLast(Handle);
		return;
	}
}

/*************************************************************
* OnSetSpeed : 이 함수는 속도설정의 변경이 필요할 때
* 호출되는 가상의 함수 입니다.
*************************************************************/
#define MAP0 0 //맵번호 (0)
void OnSetSpeed()
{
	cmmIxMapAxes(MAP0, cmX1_MASK | cmY1_MASK, 0 );
	//또는 cmmIxMapAxes(MAP0, 0x3, 0x0);
	//보간 이동할 축들의 기본속도를 설정합니다.
	cmmCfgSetSpeedPattern(cmX1, cmSMODE_T, 1000, 5000, 5000);
	cmmCfgSetSpeedPattern(cmY1, cmSMODE_T, 1000, 5000, 5000);
}

/*************************************************************
* 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.
	cmmIxSetSpeedPattern(MAP0, cmFALSE, cmSMODE_T, 100, 70, 70 );

	// Move from P1 to P2 //
	cmmIxArcATo(MAP0, 1000, 0, -90, cmFALSE);

	// Move from P2 to P3 //
	fPosList[0]=2000; fPosList[1]=1000;
	cmmIxLineTo(MAP0, fPosList, cmFALSE);

	// Move from P3 to P4 //
	cmmIxArcATo(MAP0, 2000, 500, -90, cmFALSE);

	// Move from P4 to P5 //
	fPosList[0]=2500; fPosList[1]=-500;
	cmmIxLineTo(MAP0, fPosList, cmFALSE);

	// Move from P5 to P6 //
	cmmIxArcATo(MAP0, 2000, -500, -90, cmFALSE);

	// Move from P6 to P7 //
	fPosList[0]=1000; fPosList[1]=-1000;
	cmmIxLineTo(MAP0, fPosList, cmFALSE);

	// Move from P7 to P1 //
	cmmIxArcATo(MAP0, 1000, 0, -90, cmFALSE);
}
```

```
Visual Basic
‘맵 번호 MAP0 은 이미 선언되어 있다고 가정함.
'/*************************************************************
'* OnSetSpeed : 이 함수는 속도설정의 변경이 필요할 때
'* 호출되는 가상의 함수 입니다.
'*************************************************************/
Private Sub OnSetSpeed()
```

```
  Call cmmIxMapAxes(MAP0, &H3, 0) '//&H3 = cmX1 | cmY1
  '//보간 이동할 축들의 기본속도를 설정합니다.
  Call cmmCfgSetSpeedPattern(cmX1, cmSMODE_T, 1000, 5000, 5000)
  Call cmmCfgSetSpeedPattern(cmY1, cmSMODE_T, 1000, 5000, 5000)

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 cmmIxSetSpeedPattern(MAP0, cmFALSE, cmSMODE_T, 100, 70, 70)

	'// Move from P1 to P2 //
	Call cmmIxArcATo(MAP0, 1000, 0, -90, cmFALSE)

	'// Move from P2 to P3 //
	fPosList(0) = 2000
	fPosList(1) = 1000
	Call cmmIxLineTo(MAP0, fPosList(0), cmFALSE)

	'// Move from P3 to P4 //
	Call cmmIxArcATo(MAP0, 2000, 500, -90, cmFALSE)

	'// Move from P4 to P5 //
	fPosList(0) = 2500
	fPosList(1) = -500
	Call cmmIxLineTo(MAP0, fPosList(0), cmFALSE)

	'// Move from P5 to P6 //
	Call cmmIxArcATo(MAP0, 2000, -500, -90, cmFALSE)

	'// Move from P6 to P7 //
	fPosList(0) = 1000
	fPosList(1) = -1000
	Call cmmIxLineTo(MAP0, fPosList(0), cmFALSE)

	'// Move from P7 to P1 //
	Call cmmIxArcATo(MAP0, 1000, 0, -90, cmFALSE)

End Sub
```

```
Delphi

Const MAP0 = 0;

//************************************************************
// OnProgramInitial : 이 함수는 가상의 함수로서 프로그램 초기화 루틴이
// 적용되는 부분을 의미합니다.
//************************************************************

procedure OnProgramInitial();
var
	m_nNumAxes : LongInt;

begin

	if(cmmGnDeviceLoad(cmTRUE, @m_nNumAxes) <> cmERR_NONE) then
	begin
		//Handle 은 사용자가 생성한 폼의 핸들 값입니다.
		cmmErrShowLast(Handle);
		exit;
	end;
end;
```

```
//*************************************************************
// OnSetSpeed : 이 함수는 속도설정의 변경이 필요할 때
// 호출되는 가상의 함수 입니다.
//*************************************************************

procedure OnSetSpeed();
begin
        cmmIxMapAxes(MAP0, cmX1_MASK or cmY1_MASK, 0 );
        //또는 cmmIxMapAxes(MAP0, $3, $0);
        //보간 이동할 축들의 기본속도를 설정합니다.
        cmmCfgSetSpeedPattern(cmX1, cmSMODE_T, 1000, 5000, 5000);
        cmmCfgSetSpeedPattern(cmY1, cmSMODE_T, 1000, 5000, 5000);
end;

//*************************************************************
// OnDoMotion() : 작업명령시에 호출되는 가상의 함수
//*************************************************************

procedure OnDoMotion();
var
        fPosList : Array[0..1] of Double;

begin

        //MAP0 를 마스터 속도 모드, Trapezoidal 속도 패턴으로 작업속도의 100%,
        //가속도의 70%, 감속도의 70%로 설정 합니다.
        cmmIxSetSpeedPattern(MAP0, cmFALSE, cmSMODE_T, 100, 70, 70 );

        // Move from P1 to P2 //
        cmmIxArcATo(MAP0, 1000, 0, -90, cmFALSE);

        // Move from P2 to P3 //
        fPosList[0]:=2000;
fPosList[1]:=1000;
        cmmIxLineTo(MAP0, @fPosList, cmFALSE);

        // Move from P3 to P4 //
        cmmIxArcATo(MAP0, 2000, 500, -90, cmFALSE);

        // Move from P4 to P5 //
        fPosList[0]:=2500;
fPosList[1]:=-500;
        cmmIxLineTo(MAP0,@fPosList, cmFALSE);

        // Move from P5 to P6 //
        cmmIxArcATo(MAP0, 2000, -500, -90, cmFALSE);

        // Move from P6 to P7 //
        fPosList[0]:=1000;
fPosList[1]:=-1000;
        cmmIxLineTo(MAP0, @fPosList, cmFALSE);

        // Move from P7 to P1 //
        cmmIxArcATo(MAP0, 1000, 0, -90, cmFALSE);
end;
```

NAME

cmmIxArcP
cmmIxArcPStart
– 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 cmmIxArcP

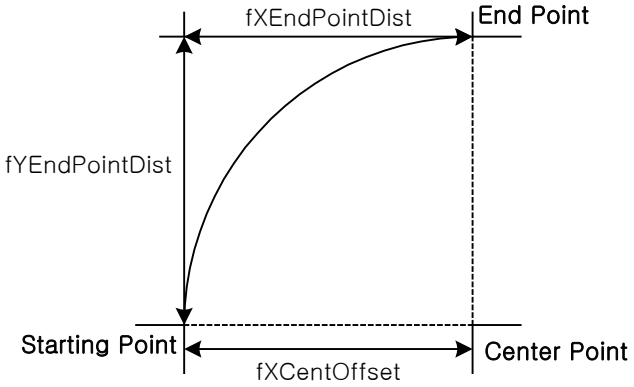
([in] VT_I4 MapIndex, [in] VT_R8 XCentOffset, [in] VT_R8 YCentOffset, [in] VT_R8 XEndPointDist, [in] VT_R8 YEndPointDist, [in] VT_I4 Direction, [in] VT_I4 IsBlocking)

❒ VT_I4 cmmIxArcPStart

([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. cmmIxArcP() 함수는 모션이 완료되기 전까지 반환되지 않으며, cmmIxArcPStart() 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.



PARAMETER

▶ MapIndex : 맵번호(Map index), 이 맵번호를 사용하기전에 먼저 cmmIxMapAxes() 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 cmARC_CW	시계 방향(CW)으로 회전
1 또는 cmARC_CCW	반시계 방향(CCW)으로 회전

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

Value	Meaning
cmFALSE	블록(Blocking)을 하지 않습니다. 따라서 해당 모션이 완료되는 동안에도 윈도우 이벤트를 처리합니다.
cmTRUE	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
cmERR_NONE	수행 성공

SEE ALSO

□ 논리적 거리 단위는 cmmCfgSetUnitDist() 함수에 의해 결정됩니다

□ cmmIxArcPStart() 함수를 사용하는 경우에는 cmmIxIsDone() 함수나 cmmIxWaitDone() 함수를 사용하여 모션의 완료를 확인(確認)할 수 있습니다.

□ cmmIxArcP() 함수를 사용하는 경우에는 내부적으로 루프를 수행하면서 모션이 완료되기를 기다리는데, 이때 "Blocking Mode"설정에 따라 윈도우 이벤트를 처리하는 방식이 달라집니다.

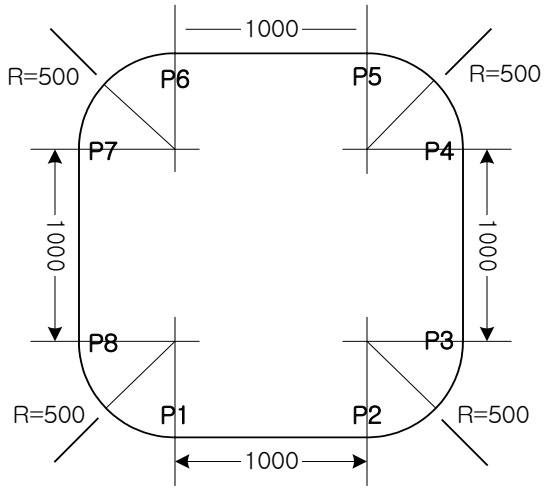
□ cmmIxArcP() 함수를 사용하는 경우에는 INP 입력신호가 Enable 로 설정되었으면 Command 펄스 출력이 완료되어도 INP 입력이 ON 이 되기 전까지는 모션이 완료되지 않은 것으로 간주되어 반환되지 않습니다.
스탭 드라이브는 INP 출력이 없는 경우가 일반적인데, 고객(顧客)님의 부주의나 잘못된 설정으로 INP 입력에 대한 설정이 Enable 로 되어 있을 경우 INP 입력이 스탭 드라이브를 통해 발생하지 않는 이유 때문에 모션 완료가 되지 않는 경우가 발생할 수 있습니다.
고객(顧客) 여러분들께서는 스탭 드라이브 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 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 "Cmmsdk.h"
#include "CmmsdkDef.h"

/****************************************************************
* OnProgramInitial : 이 함수는 가상의 함수로서 프로그램 초기화 루틴이
* 적용되는 부분을 의미합니다.
****************************************************************/
void OnProgramInitial()
{

	long m_nNumAxes;

	cmmLoadDll();
	if(cmmGnDeviceLoad(cmTRUE, &m_nNumAxes) != cmERR_NONE)
	{
		//Handle 은 사용자가 생성한 폼의 핸들 값입니다.
		cmmErrShowLast(Handle);
		return;
	}
}

/****************************************************************
* OnSetSpeed : 이 함수는 속도설정의 변경이 필요할 때
* 호출되는 가상의 함수 입니다.
****************************************************************/
#define MAP0 0 //맵번호 (0)
void OnSetSpeed()
{
	cmmIxMapAxes(MAP0, cmX1_MASK | cmY1_MASK, 0 );
	//또는 cmmIxMapAxes(MAP0, 0x3, 0x0);
	//보간 이동할 축들의 기본속도를 설정합니다.
	cmmCfgSetSpeedPattern(cmX1, cmSMODE_T, 1000, 5000, 5000);
	cmmCfgSetSpeedPattern(cmY1, cmSMODE_T, 1000, 5000, 5000);
}

/****************************************************************
* 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.
	cmmIxSetSpeedPattern(MAP0, cmFALSE, cmSMODE_T, 100, 70, 70 );

	// Move from P1 to P2 //
	fDistList[0]=1000; fDistList[1]=0;
```

```
        cmmIxLine(MAP0, fDistList, cmFALSE);

        // Move from P2 to P3 //
        cmmIxArcP(MAP0, 0, 500, 500, 500, cmARC_CCW, cmFALSE);

        // Move from P3 to P4 //
        fDistList[0]=0; fDistList[1]=1000;
        cmmIxLine(MAP0, fDistList, cmFALSE);

        // Move from P4 to P5 //
        cmmIxArcP(MAP0, -500, 0, -500, 500, cmARC_CCW, cmFALSE);

        // Move from P5 to P6 //
        fDistList[0]=-1000; fDistList[1]=0;
        cmmIxLine(MAP0, fDistList, cmFALSE);

        // Move from P6 to P7 //
        cmmIxArcP(MAP0, 0, -500, -500, -500, cmARC_CCW, cmFALSE);

        // Move from P7 to P8 //
        fDistList[0]=0; fDistList[1]=-1000;
        cmmIxLine(MAP0, fDistList, cmFALSE);

        // Move from P8 to P1 //
        cmmIxArcP(MAP0, 500, 0, 500, -500, cmARC_CCW, cmFALSE);
}
```

Visual Basic

```
Dim nIxMap As Long                    ' 보간 맵 설정

nIxMap = 0

Private Sub OnSetIxConfig ()

        ' 0 번 축과 1 번 축을 보간 맵으로 설정합니다.
        Call cmmIxMapAxes ( nIxMap, cmX1_MASK Or cmY1_MASK, 0 )

        ' 각 축의 속도를 설정합니다.
        Call cmmCfgSetSpeedPattern (cmX1, cmSMODE_T, 1000, 5000, 5000 )
        Call cmmCfgSetSpeedPattern (cmY1, cmSMODE_T, 1000, 5000, 5000 )

End Sub


Private Sub OnIxArcP_Move ()

        Dim fDistList(2) As Double

        ' 맵에 포함된 축의 속도 비율을 설정합니다.
        Call cmmCfgSetSpeedPattern ( nIxMap, cmFALSE, cmSMODE_T, 100, 70, 70 )

        ' P1 에서 P2 로 이송합니다.
        fDistList(0) = 1000
        fDistList(1) = 0
        Call cmmIxLine ( nIxMap, fDistList(0), cmFALSE )

        '  P2 에서 P3 로 이송합니다.
        Call cmmIxArcP ( nIxMap, 0, 500, 500, 500, cmARC_CCW, cmFALSE )

        ' P3 에서 P4 로 이송합니다.
        fDistList(0) = 0
        fDistList(1) = 1000
        Call cmmIxLine ( nIxMap, fDistList(0), cmFALSE )

        ' P4 에서 P5 로 이송합니다.
        Call cmmIxArcP ( nIxMap, -500, 0, -500, 500, cmARC_CCW, cmFALSE )

        ' P5 에서 P6 로 이송합니다.
        fDistList(0) = -1000
        fDistList(1) = 0
```

```
        Call cmmIxLine ( nIxMap, fDistList(0), cmFALSE )

        ' P6 에서 P7 로 이송합니다.
        Call cmmIxArcP ( nIxMap, 0, -500, -500, -500, cmARC_CCW, cmFALSE )

        ' P7 에서 P8 로 이송합니다.
        fDistList(0) = 0
        fDistList(1) = -1000
        Call cmmIxLine ( nIxMap, fDistList(0), cmFALSE )

        ' P8 에서 P1 로 이송합니다.
        Call cmmIxArcP ( nIxMap, 500, 0, 500, -500, cmARC_CCW, cmFALSE )

End Sub
```

```
Delphi

var
        nIxMap : LongInt;                       // 보간 맵 설정
begin
        nIxMap := 0;
end;


procedure OnSetIxConfig ();
begin
        // 0 번 축과 1 번 축을 보간 맵으로 설정합니다.
        cmmIxMapAxes ( nIxMap, cmX1_MASK or cmY1_MASK, 0 );

        // 각 축의 속도를 설정합니다.
        cmmCfgSetSpeedPattern (cmX1, cmSMODE_T, 1000, 5000, 5000 );
        cmmCfgSetSpeedPattern (cmY1, cmSMODE_T, 1000, 5000, 5000 );

end;


procedure OnIxArcP_Move ();
var
        fDistList : Array[0..1] of Double;

begin
        // 맵에 포함된 축의 속도 비율을 설정합니다.
        cmmCfgSetSpeedPattern ( nIxMap, cmSMODE_T, 100, 70, 70 );

        // P1 에서 P2 로 이송합니다.
        fDistList[0] := 1000;
        fDistList[1] := 0;
        cmmIxLine ( nIxMap, @fDistList, cmFALSE );

        //  P2 에서 P3 로 이송합니다.
        cmmIxArcP ( nIxMap, 0, 500, 500, 500, cmARC_CCW, cmFALSE );

        // P3 에서 P4 로 이송합니다.
        fDistList[0] := 0;
        fDistList[1] := 1000;
        cmmIxLine ( nIxMap, @fDistList, cmFALSE );

        // P4 에서 P5 로 이송합니다.
        cmmIxArcP ( nIxMap, -500, 0, -500, 500, cmARC_CCW, cmFALSE );

        // P5 에서 P6 로 이송합니다.
        fDistList[0] := -1000;
        fDistList[1] := 0;
        cmmIxLine ( nIxMap, @fDistList, cmFALSE );

        // P6 에서 P7 로 이송합니다.
        cmmIxArcP ( nIxMap, 0, -500, -500, -500, cmARC_CCW, cmFALSE );

        // P7 에서 P8 로 이송합니다.
```

```
        fDistList[0] := 0;
        fDistList[1] := -1000;
        cmmIxLine ( nIxMap, @fDistList, cmFALSE );

        // P8 에서 P1 로 이송합니다.
        cmmIxArcP ( nIxMap, 500, 0, 500, -500, cmARC_CCW, cmFALSE );

end;
```

NAME

cmmIxArcPTo
cmmIxArcPToStart
- 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 cmmIxArcPTo

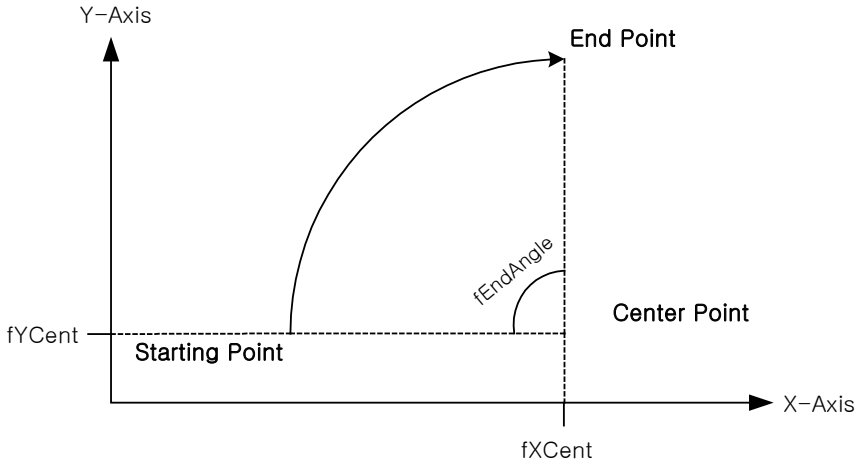
([in] VT_I4 MapIndex, [in] VT_R8 XCent, [in] VT_R8 YCent, [in] VT_R8 XEndPos, [in] VT_R8 YEndPos, [in] VT_I4 Direction, [in] VT_I4 IsBlocking)

❒ VT_I4 cmmIxArcPToStart

([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. cmmIxArcPTo() 함수는 모션이 완료되기 전까지 반환되지 않으며, cmmIxArcPToStart() 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.



PARAMETER

▶ MapIndex : 맵번호(Map index), 이 맵번호를 사용하기전에 먼저 cmmIxMapAxes() 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 cmARC_CW	시계 방향(CW)으로 회전
1 또는 cmARC_CCW	반시계 방향(CCW)으로 회전

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

Value	Meaning
cmFALSE	블록(Blocking)을 하지 않습니다. 따라서 해당 모션이 완료되는 동안에도 윈도우 이벤트를 처리합니다.
cmTRUE	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
cmERR_NONE	수행 성공

SEE ALSO

☐ 논리적 거리 단위는 cmmCfgSetUnitDist() 함수에 의해 결정됩니다.

☐ cmmIxArcPToStart() 함수를 사용하는 경우에는 cmmIxIsDone() 함수나 cmmIxWaitDone() 함수를 사용하여 모션의 완료를 확인(確認)할 수 있습니다.

☐ cmmIxArcPTo() 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.

☐ cmmIxArcPTo() 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.

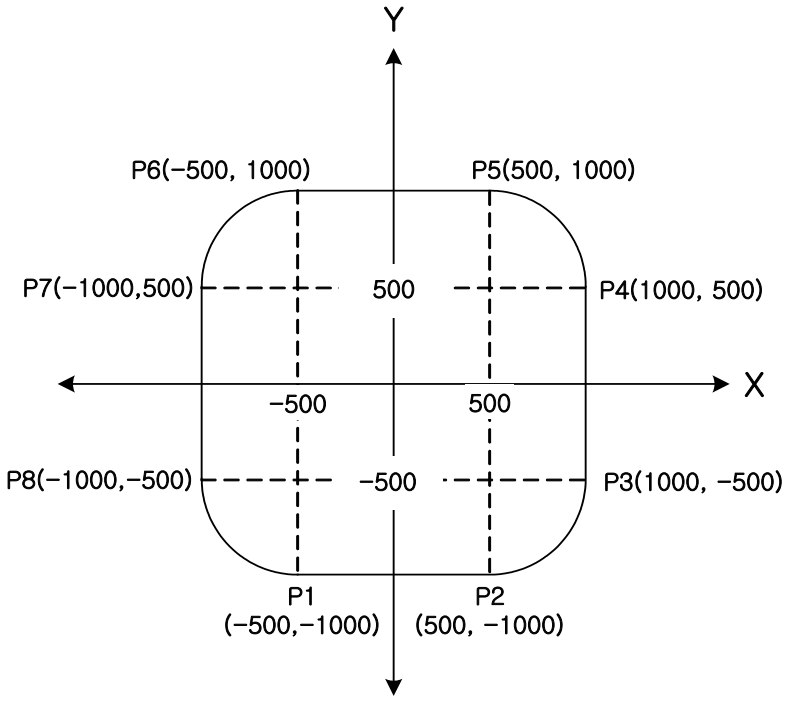
고객(顧客) 여러분들께서는 스탭 드라이브 사용시에 이점을 주의해주시기를 부탁드립니다.

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 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 "Cmmsdk.h"
#include "CmmsdkDef.h"

/************************************************************
* OnProgramInitial : 이 함수는 가상의 함수로서 프로그램 초기화 루틴이
* 적용되는 부분을 의미합니다.
************************************************************/
void OnProgramInitial()
{

	long m_nNumAxes;

	cmmLoadDll();
	if(cmmGnDeviceLoad(cmTRUE, &m_nNumAxes) != cmERR_NONE)
	{
		//Handle 은 사용자가 생성한 폼의 핸들 값입니다.
		cmmErrShowLast(Handle);
		return;
	}
}

/************************************************************
* OnSetSpeed : 이 함수는 속도설정의 변경이 필요할 때
* 호출되는 가상의 함수 입니다.
* ***********************************************************/
#define MAP0 0 //맵번호 (0)

void OnSetSpeed()
{
	cmmIxMapAxes(MAP0, cmX1_MASK | cmY1_MASK, 0 );

	//또는 cmmIxMapAxes(MAP0, 0x3, 0x0);
	//보간 이동할 축들의 기본속도를 설정합니다.
	cmmCfgSetSpeedPattern(cmX1, cmSMODE_T, 1000, 5000, 5000);
	cmmCfgSetSpeedPattern(cmY1, cmSMODE_T, 1000, 5000, 5000);
}

/************************************************************
```

```
* 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.
	cmmIxSetSpeedPattern(MAP0, cmFALSE, cmSMODE_T, 100, 70, 70 );

	// Move from P1 to P2 //
	fPosList[0]=500; fPosList[1]=-1000;
	cmmIxLineTo(MAP0, fPosList, cmFALSE);

	// Move from P2 to P3 //
	cmmIxArcPTo(MAP0, 500, -500, 1000, -500, cmARC_CCW, cmFALSE);

	// Move from P3 to P4 //
	fPosList[0]=1000; fPosList[1]=500;
	cmmIxLineTo(MAP0, fPosList, cmFALSE);

	// Move from P4 to P5 //
	cmmIxArcPTo(MAP0, 500, 500, 500, 1000, cmARC_CCW, cmFALSE);

	// Move from P5 to P6 //
	fPosList[0]=-500; fPosList[1]=1000;
	cmmIxLineTo(MAP0, fPosList, cmFALSE);

	// Move from P6 to P7 //
	cmmIxArcPTo(MAP0, -500, 500, -1000, 500, cmARC_CCW, cmFALSE);

	// Move from P7 to P8 //
	fPosList[0]=-1000; fPosList[1]=-500;
	cmmIxLineTo(MAP0, fPosList, cmFALSE);

	// Move from P8 to P1 //
	cmmIxArcPTo(MAP0,-500, -500, -500, -1000, cmARC_CCW, cmFALSE);

}
```

```
Visual Basic
‘맵번호 MAP0 은 이미 선언되어 있다고 가정함.
'/************************************************************
'* OnSetSpeed : 이 함수는 속도설정의 변경이 필요할 때
'* 호출되는 가상의 함수 입니다.
'************************************************************/
Private Sub OnSetSpeed()

	Call cmmIxMapAxes(MAP0, &H3, 0)    '//&H3 = cmX1_MASK | cmY1_MASK

	'//보간 이동할 축들의 기본속도를 설정합니다.
	Call cmmCfgSetSpeedPattern(cmX1, cmSMODE_T, 1000, 5000, 5000)
	Call cmmCfgSetSpeedPattern(cmY1, cmSMODE_T, 1000, 5000, 5000)

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 cmmIxSetSpeedPattern(MAP0, cmFALSE, cmSMODE_T, 100, 70, 70)

	'// Move from P1 to P2 //
	fPosList(0) = 500
	fPosList(1) = -1000
	Call cmmIxLineTo(MAP0, fPosList(0), cmFALSE)
```

```
        '// Move from P2 to P3 //
        Call cmmIxArcPTo(MAP0, 500, -500, 1000, -500, cmARC_CCW, cmFALSE)

        '// Move from P3 to P4 //
        fPosList(0) = 1000
        fPosList(1) = 500
        Call cmmIxLineTo(MAP0, fPosList(0), cmFALSE)

        '// Move from P4 to P5 //
        Call cmmIxArcPTo(MAP0, 500, 500, 500, 1000, cmARC_CCW, cmFALSE)

        '// Move from P5 to P6 //
        fPosList(0) = -500
        fPosList(1) = 1000
        Call cmmIxLineTo(MAP0, fPosList(0), cmFALSE)

        '// Move from P6 to P7 //
        Call cmmIxArcPTo(MAP0, -500, 500, -1000, 500, cmARC_CCW, cmFALSE)

        '// Move from P7 to P8 //
        fPosList(0) = -1000
        fPosList(1) = -500
        Call cmmIxLineTo(MAP0, fPosList(0), cmFALSE)

        '// Move from P8 to P1 //
        Call cmmIxArcPTo(MAP0, -500, -500, -500, -1000, cmARC_CCW, cmFALSE)

End Sub
```

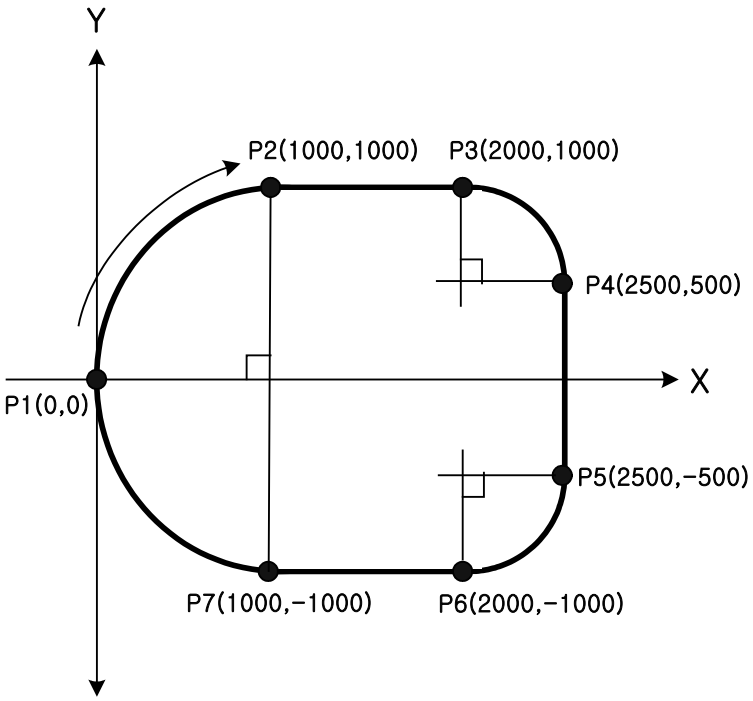
Delphi

```
var
        nIxMap : LongInt;                   // 보간 맵 설정
begin
        nIxMap := 0;
end;


procedure OnSetIxConfig ();
begin

        // 0 번 축과 1 번 축을 보간 맵으로 설정합니다.
        cmmIxMapAxes ( nIxMap, cmX1_MASK or cmY1_MASK, 0 );;

        // 각 축의 속도를 설정합니다.
        cmmCfgSetSpeedPattern (cmX1, cmSMODE_T, 1000, 5000, 5000 );
        cmmCfgSetSpeedPattern (cmY1, cmSMODE_T, 1000, 5000, 5000 );

end;


procedure OnIxArcPTo_Move ();
var
        fPosList : Array[0..1] of Double;

begin
        // 맵에 포함된 축의 속도 비율을 설정합니다.
        cmmCfgSetSpeedPattern ( nIxMap, cmSMODE_T, 100, 70, 70 );

        // P1 에서 P2 로 이동합니다.
        fPosList[0] := 500;
        fPosList[1] := -1000;
        cmmIxLineTo ( nIxMap, @fPosList, cmFALSE );

        // P2 에서 P3 로 이송합니다.
        cmmIxArcPTo ( nIxMap, 500, -500, 1000, -500, cmARC_CCW, cmFALSE );

        // P3 에서 P4 로 이송합니다.
        fPosList[0] := 1000;
        fPosList[1] := 500;
        cmmIxLineTo ( nIxMap, @fPosList, cmFALSE );
```

```
    // P4 에서 P5 로 이송합니다.
    cmmIxArcPTo ( nIxMap, 500, 500, 500, 1000, cmARC_CCW, cmFALSE );

    // P5 에서 P6 로 이송합니다.
    fPosList[0] := -500;
    fPosList[1] := 1000;
    cmmIxLineTo ( nIxMap, @fPosList, cmFALSE );

    // P6 에서 P7 로 이송합니다.
    cmmIxArcPTo ( nIxMap, -500, 500, -1000, 500, cmARC_CCW, cmFALSE );

    // P7 에서 P8 로 이송합니다.
    fPosList[0] := -1000;
    fPosList[1] := -500;
    cmmIxLineTo ( nIxMap, @fPosList, cmFALSE );

    // P8 에서 P1 로 이송합니다.
    cmmIxArcPTo ( nIxMap, -500, -500, -500, -1000, cmARC_CCW, cmFALSE );

end;
```

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 "Cmmsdk.h"
#include "CmmsdkDef.h"

/****************************************************************
* OnProgramInitial : 이 함수는 가상의 함수로서 프로그램 초기화 루틴이
* 적용되는 부분을 의미합니다.
****************************************************************/
void OnProgramInitial()
{
```

```
	long m_nNumAxes;

	cmmLoadDll();
	if(cmmGnDeviceLoad(cmTRUE, &m_nNumAxes) != cmERR_NONE)
	{
		//Handle 은 사용자가 생성한 폼의 핸들 값입니다.
		cmmErrShowLast(Handle);
		return;
	}
}

/************************************************************
* OnSetSpeed : 이 함수는 속도설정의 변경이 필요할 때
* 호출되는 가상의 함수 입니다.
************************************************************/
#define MAP0 0 //맵번호 (0)
void OnSetSpeed()
{
	cmmIxMapAxes(MAP0, cmX1_MASK | cmY1_MASK, 0 );
	//또는 cmmIxMapAxes(MAP0, 0x3, 0x0);
	//보간 이동할 축들의 기본속도를 설정합니다.
	cmmCfgSetSpeedPattern(cmX1, cmSMODE_T, 1000, 5000, 5000);
	cmmCfgSetSpeedPattern(cmY1, cmSMODE_T, 1000, 5000, 5000);
}

/************************************************************
* 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.
	cmmIxSetSpeedPattern(MAP0, cmFALSE, cmSMODE_T, 100, 70, 70 );

	// Move from P1 to P2 //
	cmmIxArcPTo(MAP0, 1000, 0, 1000, 1000, cmARC_CW, cmFALSE);

	// Move from P2 to P3 //
	fPosList[0]=2000; fPosList[1]=1000;
	cmmIxLineTo(MAP0, fPosList, cmFALSE);

	// Move from P3 to P4 //
	cmmIxArcPTo(MAP0, 2000, 500, 2500, 500, cmARC_CW, cmFALSE);

	// Move from P4 to P5 //
	fPosList[0]=2500; fPosList[1]=-500;
	cmmIxLineTo(MAP0, fPosList, cmFALSE);

	// Move from P5 to P6 //
	cmmIxArcPTo(MAP0, 2000, -500, 2000, -1000, cmARC_CW, cmFALSE);

	// Move from P6 to P7 //
	fPosList[0]=1000; fPosList[1]=-1000;
	cmmIxLineTo(MAP0, fPosList, cmFALSE);

	// Move from P7 to P1 //
	cmmIxArcPTo(MAP0, 1000, 0, 0, 0, cmARC_CW, cmFALSE);
}
```

```
Visual Basic
‘맵번호 MAP0 은 이미 선언되어 있다고 가정함.
'/************************************************************
'* OnSetSpeed : 이 함수는 속도설정의 변경이 필요할 때
'* 호출되는 가상의 함수 입니다.
'************************************************************/
Private Sub OnSetSpeed()

	Call cmmIxMapAxes(MAP0, &H3, 0)
```

```
	'//또는 cmmIxMapAxes(MAP0, 0x3, 0x0);

	'//보간 이동할 축들의 기본속도를 설정합니다.
	Call cmmCfgSetSpeedPattern(cmX1, cmSMODE_T, 1000, 5000, 5000)
	Call cmmCfgSetSpeedPattern(cmY1, cmSMODE_T, 1000, 5000, 5000)

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 cmmIxSetSpeedPattern(MAP0, cmFALSE, cmSMODE_T, 100, 70, 70)

	'// Move from P1 to P2 //
	Call cmmIxArcPTo(MAP0, 1000, 0, 1000, 1000, cmARC_CW, cmFALSE)
	'// Move from P2 to P3 //
	fPosList(0) = 2000
	fPosList(1) = 1000
	Call cmmIxLineTo(MAP0, fPosList(0), cmFALSE)

	'// Move from P3 to P4 //
	Call cmmIxArcPTo(MAP0, 2000, 500, 2500, 500, cmARC_CW, cmFALSE)

	'// Move from P4 to P5 //
	fPosList(0) = 2500
	fPosList(1) = -500
	Call cmmIxLineTo(MAP0, fPosList(0), cmFALSE)

	'// Move from P5 to P6 //
	Call cmmIxArcPTo(MAP0, 2000, -500, 2000, -1000, cmARC_CW, cmFALSE)

	'// Move from P6 to P7 //
	fPosList(0) = 1000
	fPosList(1) = -1000
	Call cmmIxLineTo(MAP0, fPosList(0), cmFALSE)

	'// Move from P7 to P1 //
	Call cmmIxArcPTo(MAP0, 1000, 0, 0, 0, cmARC_CW, cmFALSE)

End Sub
```

Delphi

```
Const MAP0 = 0;

//*************************************************************
// OnProgramInitial : 이 함수는 가상의 함수로서 프로그램 초기화 루틴이
// 적용되는 부분을 의미합니다.
//*************************************************************

procedure OnProgramInitial();
var
	m_nNumAxes : LongInt;

begin

	if(cmmGnDeviceLoad(cmTRUE, @m_nNumAxes) <> cmERR_NONE) then
	begin
		//Handle 은 사용자가 생성한 폼의 핸들 값입니다.
		cmmErrShowLast(Handle);
		exit;
	end;
end;
```

```
//*************************************************************
// OnSetSpeed : 이 함수는 속도설정의 변경이 필요할 때
// 호출되는 가상의 함수 입니다.
//*************************************************************

procedure OnSetSpeed();
begin
        cmmIxMapAxes(MAP0, cmX1_MASK or cmY1_MASK, 0 );
        //또는 cmmIxMapAxes(MAP0, $3, $0);
        //보간 이동할 축들의 기본속도를 설정합니다.
        cmmCfgSetSpeedPattern(cmX1, cmSMODE_T, 1000, 5000, 5000);
        cmmCfgSetSpeedPattern(cmY1, cmSMODE_T, 1000, 5000, 5000);
end;

//*************************************************************
// OnDoMotion() : 작업명령시에 호출되는 가상의 함수
//*************************************************************

Procedure OnDoMotion();
var
        fPosList : Array[0..1] of Double;

begin

        //MAP0 를 마스터 속도 모드, Trapezoidal 속도 패턴으로 작업속도의 100%,
        //가속도의 70%, 감속도의 70%로 설정 합니다.
        cmmIxSetSpeedPattern(MAP0, cmFALSE, cmSMODE_T, 100, 70, 70 );

        // Move from P1 to P2 //
        cmmIxArcPTo(MAP0, 1000, 0, 1000, 1000, cmARC_CW, cmFALSE);

        // Move from P2 to P3 //
        fPosList[0]:=2000;
        fPosList[1]:=1000;
        cmmIxLineTo(MAP0, @fPosList, cmFALSE);

        // Move from P3 to P4 //
        cmmIxArcPTo(MAP0, 2000, 500, 2500, 500, cmARC_CW, cmFALSE);

        // Move from P4 to P5 //
        fPosList[0]:=2500;
        fPosList[1]:=-500;
        cmmIxLineTo(MAP0, @fPosList, cmFALSE);

        // Move from P5 to P6 //
        cmmIxArcPTo(MAP0, 2000, -500, 2000, -1000, cmARC_CW, cmFALSE);

        // Move from P6 to P7 //
        fPosList[0]:=1000;
        fPosList[1]:=-1000;
        cmmIxLineTo(MAP0, @fPosList, cmFALSE);

        // Move from P7 to P1 //
        cmmIxArcPTo(MAP0, 1000, 0, 0, 0, cmARC_CW, cmFALSE);

end;
```

NAME

cmmIxArc3P
– 3 점 사용 원호 보간 상대좌표 이송
(현재 좌표 및 상대적 2 차 좌표와 3 차 좌표 사용)

INFORMATION

Interpolation Motion

VC++/VB

BCB/Delphi/.NET

Level 4

☺ 이송 함수

실제 이송이 진행되는 함수이므로, 사전에 반드시 안전을 확인합니다

SYNOPSIS

❒VT_I4 cmmIxArc3P

([in] VT_I4 MapIndex, [in] VT_R8 P2X, [in] VT_R8 P2Y, [in] VT_R8 P3X, [in] VT_R8 P3Y,

[in] VT_R8 EndAngle, [in] VT_I4 IsBlocking)

DESCRIPTION

본 함수는 원호의 중심점 및 종점 좌표를 사용하는 cmmIxArcP(상대 좌표) 또는 cmmIxArcP (절대 좌표) 함수와는 달리 현재 좌표(1 차)를 포함해서 2 차 및 3 차 좌표의 총 3 개의 (X, Y) 순서쌍 좌표를 사용해서 원호 보간을 수행합니다. 결과적으로 세 점을 지나게 되므로 중심점은 자동으로 결정됩니다.

PARAMETER

▶ MapIndex : 맵번호(Map index), 이 맵번호를 사용하기전에 먼저 cmmIxMapAxes() 함수를 통하여 해당 맵번호에 유효한 축들이 맵핑되어 있어야 합니다.

▶ VT_R8P2X : 원호가 지나는 2 번째 점의 X 좌표 입니다.

▶ VT_R8P2Y : 원호가 지나는 2 번째 점의 Y 좌표 입니다.

▶ VT_R8P3X : 원호가 지나는 3 번째 점의 X 좌표 입니다.

▶ VT_R8P3Y : 원호가 지나는 3 번째 점의 Y 좌표 입니다.

▶ EndAngle : 중심점과 출발점을 잇는 선분이 중심점과 종점을 잇는 선분과 이루는 각도를 의미 합니다.

Value	Meaning
0 도	3 번째 점에서 보간 이송 완료 하도록 설정. 이 경우 종점도 3 번째 점이 됩니다.
0 도 초과 360 도 이하	현재 좌표(1 번째), 2 번째 및 3 번째를 지나 중심점으로부터 해당 각도 만큼 원호 보간 이송이 수행됩니다. 단, 각도가 2 번째 또는 3 번째 점을 지나기에 모자라는 경우는 해당 각도까지만 그려지고 2 번째 및 3 번째 점은 지나지 않게 됩니다.

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

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

SEE ALSO

cmmIxArcP
cmmIxArcPTo
cmmIxArcPStart
cmmIxArcPToStart
cmmIxArc3PStart

RETURN VALUE

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

REFERENCE

이 함수 사용 시 현재 좌표를 포함한 3 개의 점이 거의 직선에 가깝게 설정되는 경우는 사실상 지름이 무한대 거나 상당히 큰 원을 그려야 하는 문제가 발생하며, 이 경우는 함수 수행은 되고 반환값이 cmERR_NONE 이 되더라도 실제로는 제대로 이송이 안되고 원호 보간 이송에 실패할 수 있음에 유의해야 합니다.

EXAMPLE

C/C++

```
/*************************************************************
* OnSetSpeed : 이 함수는 속도설정의 변경이 필요할 때
* 호출되는 가상의 함수 입니다.
* ************************************************************/
#define MAP0 0 //맵번호 (0)

void OnSetSpeed()
{
	cmmIxMapAxes(MAP0, cmX1_MASK | cmY1_MASK, 0 );

	//또는 cmmIxMapAxes(MAP0, 0x3, 0x0);
	//보간 이동할 축들의 기본속도를 설정합니다.
	cmmCfgSetSpeedPattern(cmX1, cmSMODE_T, 1000, 5000, 5000);
	cmmCfgSetSpeedPattern(cmY1, cmSMODE_T, 1000, 5000, 5000);
}

/*************************************************************
* OnDoMotion() : 작업명령시에 호출되는 가상의 함수
*************************************************************/
void OnDoMotion()
{
	double fPos2x, fPos2y, fPos3x, fPos3y;

	//MAP0 를 마스터 속도 모드, Trapezoidal 속도 패턴으로 작업속도의 100%,
	//가속도의 70%, 감속도의 70%로 설정 합니다.
	cmmIxSetSpeedPattern(MAP0, cmFALSE, cmSMODE_T, 100, 70, 70 );

	// 현재 위치를 (0, 0) 이라고 가정, 2 번째 및 3 번째 지나는 점 설정 및 이송 시작
	// 끝 각은 270 도로 설정
	fPos2x=1000; fPos2y=1000; fPos3x=2000; fPos3y=2000;
	cmmIxArc3P(MAP0, fPos2x, fPos2y, fPos3x, fPos3y, 270, cmFALSE);
}
```

Visual Basic

```
' ************************************************************
' OnSetSpeed : 이 함수는 속도설정의 변경이 필요할 때
' 호출되는 가상의 함수 입니다.
' ************************************************************

Const MAP0 = 0 '맵번호 (0)

Private Sub OnSetSpeed()

        Call cmmIxMapAxes(MAP0, cmX1_MASK Or cmY1_MASK, 0 )

        ' 또는 Call cmmIxMapAxes(MAP0, &H3, &H0);
        ' 보간 이동할 축들의 기본속도를 설정합니다.
        Call cmmCfgSetSpeedPattern(cmX1, cmSMODE_T, 1000, 5000, 5000)
        Call cmmCfgSetSpeedPattern(cmY1, cmSMODE_T, 1000, 5000, 5000)

End Sub

' ************************************************************
' OnDoMotion() : 작업명령시에 호출되는 가상의 함수
' ************************************************************

Private Sub OnDoMotion()

        Dim fPos2x As Double
        Dim fPos2y As Double
        Dim fPos3x As Double
        Dim fPos3y As Double

        ' MAP0 를 마스터 속도 모드, Trapezoidal 속도 패턴으로 작업속도의 100%,
        ' 가속도의 70%, 감속도의 70%로 설정 합니다.
        Call cmmIxSetSpeedPattern(MAP0, cmFALSE, cmSMODE_T, 100, 70, 70 )

        ' 현재 위치를 (0, 0) 이라고 가정, 2 번째 및 3 번째 지나는 점 설정 및 이송 시작
        ' 끝 각은 270 도로 설정
        fPos2x=1000
        fPos2y=1000
        fPos3x=2000
        fPos3y=2000
        Call cmmIxArc3P(MAP0, fPos2x, fPos2y, fPos3x, fPos3y, 270, cmFALSE)

End Sub
```

```
Delphi

//************************************************************
// OnSetSpeed : 이 함수는 속도설정의 변경이 필요할 때
// 호출되는 가상의 함수 입니다.
//************************************************************

Const MAP0 = 0;  //맵번호 (0)

procedure OnSetSpeed();
begin
        cmmIxMapAxes(MAP0, cmX1_MASK or cmY1_MASK, 0 );

        //또는 cmmIxMapAxes(MAP0, $3, $0);
        //보간 이동할 축들의 기본속도를 설정합니다.
        cmmCfgSetSpeedPattern(cmX1, cmSMODE_T, 1000, 5000, 5000);
        cmmCfgSetSpeedPattern(cmY1, cmSMODE_T, 1000, 5000, 5000);
end;

//************************************************************
// OnDoMotion() : 작업명령시에 호출되는 가상의 함수
//************************************************************

procedure OnDoMotion();
var
        fPos2x : Double;
```

```
        fPos2y : Double;
        fPos3x : Double;
        fPos3y : Double;

begin
        //MAP0 를 마스터 속도 모드, Trapezoidal 속도 패턴으로 작업속도의 100%,
        //가속도의 70%, 감속도의 70%로 설정 합니다.
        cmmIxSetSpeedPattern(MAP0, cmFALSE, cmSMODE_T, 100, 70, 70 );

        // 현재 위치를 (0, 0) 이라고 가정, 2 번째 및 3 번째 지나는 점 설정 및 이송 시작
        // 끝 각은 270 도로 설정
        fPos2x :=1000;
        fPos2y :=1000;
        fPos3x :=2000;
        fPos3y :=2000;
        cmmIxArc3P(MAP0, fPos2x, fPos2y, fPos3x, fPos3y, 270, cmFALSE);
end;
```

NAME

cmmIxArc3PStart
– 3 점 사용 원호 보간 상대좌표 이송
(현재 좌표 및 상대적 2 차 좌표와 3 차 좌표 사용)

INFORMATION

- Interpolation Motion
- VC++/VB
- BCB/Delphi/.NET
- Level 4
- ☺ 이송 함수

실제 이송이 진행되는 함수이므로, 사전에 반드시 안전을 확인합니다

SYNOPSIS

❒VT_I4 cmmIxArc3PStart

([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

본 함수는 원호의 중심점 및 종점 좌표를 사용하는 cmmIxArcPStart(상대 좌표) 또는 cmmIxArcPToStart (절대 좌표) 함수와는 달리 현재 좌표(1 차)를 포함해서 2 차 및 3 차 좌표의 총 3 개의 (X, Y) 순서쌍 좌표를 사용해서 원호 보간을 수행합니다.
결과적으로 세 점을 지나게 되므로 중심점은 자동으로 결정됩니다.
또한 함수 이름이 'Start' 로 끝나므로 함수 내부에서 이송 완료시까지 기다리지 않고 이송 작업만 시작시킨 뒤 바로 반환 됩니다. 따라서 cmmIxWaitDone 함수와 함께 사용하셔야 이송 완료 체크를 하실 수 있습니다.

PARAMETER

▶ MapIndex : 맵번호(Map index), 이 맵번호를 사용하기전에 먼저 cmmIxMapAxes() 함수를 통하여 해당 맵번호에 유효한 축들이 맵핑되어 있어야 합니다.

▶ VT_R8P2X : 원호가 지나는 2 번째 점의 X 좌표 입니다.

▶ VT_R8P2Y : 원호가 지나는 2 번째 점의 Y 좌표 입니다.

▶ VT_R8P3X : 원호가 지나는 3 번째 점의 X 좌표 입니다.

▶ VT_R8P3Y : 원호가 지나는 3 번째 점의 Y 좌표 입니다.

▶ EndAngle : 중심점과 출발점을 잇는 선분이 중심점과 종점을 잇는 선분과 이루는 각도를 의미 합니다.

Value	Meaning
0 도	3 번째 점에서 보간 이송 완료 하도록 설정. 이 경우 종점도 3 번째 점이 됩니다.
0 도 초과 360 도 이하	현재 좌표(1 번째), 2 번째 및 3 번째를 지나 중심점으로부터 해당 각도 만큼 원호 보간 이송이 수행됩니다. 단, 각도가 2 번째 또는 3 번째 점을 지나기에 모자라는 경우는 해당 각도 까지만 그려지고 2 번째 및 3 번째 점은 지나지 않게 됩니다.

SEE ALSO

cmmIxArcP
cmmIxArcPTo
cmmIxArcPStart
cmmIxArcPToStart

cmmIxArc3P

RETURN VALUE

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

REFERENCE

이 함수 사용 시 현재 좌표를 포함한 3 개의 점이 거의 직선에 가깝게 설정되는 경우는 사실상 지름이 무한대 거나 상당히 큰 원을 그려야 하는 문제가 발생하며, 이 경우는 함수 수행은 되고 반환값이 cmERR_NONE 이 되더라도 실제로는 제대로 이송이 안되고 원호 보간 이송에 실패할 수 있음에 유의해야 합니다.

EXAMPLE

C/C++

```
/*************************************************************
* OnSetSpeed : 이 함수는 속도설정의 변경이 필요할 때
* 호출되는 가상의 함수 입니다.
* ************************************************************/
#define MAP0 0 //맵번호 (0)

void OnSetSpeed()
{
	cmmIxMapAxes(MAP0, cmX1_MASK | cmY1_MASK, 0 );

	//또는 cmmIxMapAxes(MAP0, 0x3, 0x0);
	//보간 이동할 축들의 기본속도를 설정합니다.
	cmmCfgSetSpeedPattern(cmX1, cmSMODE_T, 1000, 5000, 5000);
	cmmCfgSetSpeedPattern(cmY1, cmSMODE_T, 1000, 5000, 5000);
}

/*************************************************************
* OnDoMotion() : 작업명령시에 호출되는 가상의 함수
*************************************************************/
void OnDoMotion()
{
	double fPos2x, fPos2y, fPos3x, fPos3y;

	//MAP0 를 마스터 속도 모드, Trapezoidal 속도 패턴으로 작업속도의 100%,
	//가속도의 70%, 감속도의 70%로 설정 합니다.
	cmmIxSetSpeedPattern(MAP0, cmFALSE, cmSMODE_T, 100, 70, 70 );

	// 현재 위치를 (0, 0) 이라고 가정, 2 번째 및 3 번째 지나는 점 설정 및 이송 시작
	// 끝 각은 270 도로 설정
	fPos2x=1000; fPos2y=1000; fPos3x=2000; fPos3y=2000;
	cmmIxArc3PStart(MAP0, fPos2x, fPos2y, fPos3x, fPos3y, 270);
}
```

Visual Basic

```
' ************************************************************
' OnSetSpeed : 이 함수는 속도설정의 변경이 필요할 때
' 호출되는 가상의 함수 입니다.
' ************************************************************

Const MAP0 = 0 '맵번호 (0)

Private Sub OnSetSpeed()
```

```
        Call cmmIxMapAxes(MAP0, cmX1_MASK Or cmY1_MASK, 0 )

        ' 또는 Call cmmIxMapAxes(MAP0, &H3, &H0);
        ' 보간 이동할 축들의 기본속도를 설정합니다.
        Call cmmCfgSetSpeedPattern(cmX1, cmSMODE_T, 1000, 5000, 5000)
        Call cmmCfgSetSpeedPattern(cmY1, cmSMODE_T, 1000, 5000, 5000)

End Sub

' ************************************************************
' OnDoMotion() : 작업명령시에 호출되는 가상의 함수
' ************************************************************

Private Sub OnDoMotion()

        Dim fPos2x As Double
        Dim fPos2y As Double
        Dim fPos3x As Double
        Dim fPos3y As Double

        ' MAP0 를 마스터 속도 모드, Trapezoidal 속도 패턴으로 작업속도의 100%,
        ' 가속도의 70%, 감속도의 70%로 설정 합니다.
        Call cmmIxSetSpeedPattern(MAP0, cmFALSE, cmSMODE_T, 100, 70, 70 )

        ' 현재 위치를 (0, 0) 이라고 가정, 2 번째 및 3 번째 지나는 점 설정 및 이송 시작
        ' 끝 각은 270 도로 설정
        fPos2x=1000
        fPos2y=1000
        fPos3x=2000
        fPos3y=2000
        Call cmmIxArc3PStart(MAP0, fPos2x, fPos2y, fPos3x, fPos3y, 270)

End Sub
```

Delphi

```
//************************************************************
// OnSetSpeed : 이 함수는 속도설정의 변경이 필요할 때
// 호출되는 가상의 함수 입니다.
//************************************************************

Const MAP0 = 0;  //맵번호 (0)

procedure OnSetSpeed();
begin
        cmmIxMapAxes(MAP0, cmX1_MASK or cmY1_MASK, 0 );

        //또는 cmmIxMapAxes(MAP0, $3, $0);
        //보간 이동할 축들의 기본속도를 설정합니다.
        cmmCfgSetSpeedPattern(cmX1, cmSMODE_T, 1000, 5000, 5000);
        cmmCfgSetSpeedPattern(cmY1, cmSMODE_T, 1000, 5000, 5000);
end;

//************************************************************
// OnDoMotion() : 작업명령시에 호출되는 가상의 함수
//************************************************************

procedure OnDoMotion();
var
        fPos2x : Double;
        fPos2y : Double;
        fPos3x : Double;
        fPos3y : Double;

begin
        //MAP0 를 마스터 속도 모드, Trapezoidal 속도 패턴으로 작업속도의 100%,
        //가속도의 70%, 감속도의 70%로 설정 합니다.
        cmmIxSetSpeedPattern(MAP0, cmFALSE, cmSMODE_T, 100, 70, 70 );
```

```
        // 현재 위치를 (0, 0) 이라고 가정, 2 번째 및 3 번째 지나는 점 설정 및 이송 시작
        // 끝 각은 270 도로 설정
        fPos2x :=1000;
        fPos2y :=1000;
        fPos3x :=2000;
        fPos3y :=2000;
        cmmIxArc3PStart(MAP0, fPos2x, fPos2y, fPos3x, fPos3y, 270);
end;
```

NAME

cmmIxArc3PTo
– 3 점 사용 원호 보간 절대좌표 이송 (현재 좌표 및 절대적 2 차 좌표와 3 차 좌표 사용)

INFORMATION

- Interpolation Motion
- VC++/VB
- BCB/Delphi/.NET
- Level 4
- ☺ 이송 함수

실제 이송이 진행되는 함수이므로, 사전에 반드시 안전을 확인합니다

SYNOPSIS

❒VT_I4 cmmIxArc3PTo

([in] VT_I4 MapIndex, [in] VT_R8 P2X, [in] VT_R8 P2Y, [in] VT_R8 P3X, [in] VT_R8 P3Y,

[in] VT_R8 EndAngle, [in] VT_I4 IsBlocking)

DESCRIPTION

본 함수는 원호의 중심점 및 종점 좌표를 사용하는 cmmIxArcP(상대 좌표) 또는 cmmIxArcPTo (절대 좌표) 함수와는 달리 현재 좌표(1 차)를 포함해서 2 차 및 3 차 좌표의 총 3 개의 (X, Y) 순서쌍 좌표를 사용해서 원호 보간을 수행합니다. 함수 이름에 'To' 가 포함되므로 절대적 위치로 원호 보간 이송이 됩니다.
결과적으로 세 점을 지나게 되므로 중심점은 자동으로 결정됩니다.

PARAMETER

▶ MapIndex : 맵번호(Map index), 이 맵번호를 사용하기전에 먼저 cmmIxMapAxes() 함수를 통하여 해당 맵번호에 유효한 축들이 맵핑되어 있어야 합니다.

▶ VT_R8P2X : 원호가 지나는 2 번째 점의 X 좌표 입니다.

▶ VT_R8P2Y : 원호가 지나는 2 번째 점의 Y 좌표 입니다.

▶ VT_R8P3X : 원호가 지나는 3 번째 점의 X 좌표 입니다.

▶ VT_R8P3Y : 원호가 지나는 3 번째 점의 Y 좌표 입니다.

▶ EndAngle : 중심점과 출발점을 잇는 선분이 중심점과 종점을 잇는 선분과 이루는 각도를 의미 합니다.

Value	Meaning
0 도	3 번째 점에서 보간 이송 완료 하도록 설정. 이 경우 종점도 3 번째 점이 됩니다.
0 도 초과 360 도 이하	현재 좌표(1 번째), 2 번째 및 3 번째를 지나 중심점으로부터 해당 각도 만큼 원호 보간 이송이 수행됩니다. 단, 각도가 2 번째 또는 3 번째 점을 지나기에 모자라는 경우는 해당 각도 까지만 그려지고 2 번째 및 3 번째 점은 지나지 않게 됩니다.

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

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

SEE ALSO

cmmIxArcP
cmmIxArcPTo
cmmIxArcPStart
cmmIxArcPToStart
cmmIxArc3PToStart

RETURN VALUE

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

REFERENCE

이 함수 사용 시 현재 좌표를 포함한 3 개의 점이 거의 직선에 가깝게 설정되는 경우는 사실상 지름이 무한대 거나 상당히 큰 원을 그려야 하는 문제가 발생하며, 이 경우는 함수 수행은 되고 반환값이 cmERR_NONE 이 되더라도 실제로는 제대로 이송이 안되고 원호 보간 이송에 실패할 수 있음에 유의해야 합니다.

EXAMPLE

C/C++

```
/*************************************************************
* OnSetSpeed : 이 함수는 속도설정의 변경이 필요할 때
* 호출되는 가상의 함수 입니다.
* ************************************************************/
#define MAP0 0 //맵번호 (0)

void OnSetSpeed()
{
	cmmIxMapAxes(MAP0, cmX1_MASK | cmY1_MASK, 0 );

	//또는 cmmIxMapAxes(MAP0, 0x3, 0x0);
	//보간 이동할 축들의 기본속도를 설정합니다.
	cmmCfgSetSpeedPattern(cmX1, cmSMODE_T, 1000, 5000, 5000);
	cmmCfgSetSpeedPattern(cmY1, cmSMODE_T, 1000, 5000, 5000);
}

/*************************************************************
* OnDoMotion() : 작업명령시에 호출되는 가상의 함수
*************************************************************/
void OnDoMotion()
{
	double fPos2x, fPos2y, fPos3x, fPos3y;

	//MAP0 를 마스터 속도 모드, Trapezoidal 속도 패턴으로 작업속도의 100%,
	//가속도의 70%, 감속도의 70%로 설정 합니다.
	cmmIxSetSpeedPattern(MAP0, cmFALSE, cmSMODE_T, 100, 70, 70 );

	// 현재 위치를 (0, 0) 이라고 가정, 2 번째 및 3 번째 지나는 점 설정 및 이송 시작
	// 끝 각은 270 도로 설정
	fPos2x=1000; fPos2y=1000; fPos3x=2000; fPos3y=2000;
	cmmIxArc3PTo(MAP0, fPos2x, fPos2y, fPos3x, fPos3y, 270, cmFALSE);
}
```

Visual Basic

```
' ************************************************************
```

```
' OnSetSpeed : 이 함수는 속도설정의 변경이 필요할 때
' 호출되는 가상의 함수 입니다.
' ************************************************************

Const MAP0 = 0 '맵번호 (0)

Private Sub OnSetSpeed()

        Call cmmIxMapAxes(MAP0, cmX1_MASK Or cmY1_MASK, 0 )

        ' 또는 Call cmmIxMapAxes(MAP0, &H3, &H0);
        ' 보간 이동할 축들의 기본속도를 설정합니다.
        Call cmmCfgSetSpeedPattern(cmX1, cmSMODE_T, 1000, 5000, 5000)
        Call cmmCfgSetSpeedPattern(cmY1, cmSMODE_T, 1000, 5000, 5000)

End Sub

' ************************************************************
' OnDoMotion() : 작업명령시에 호출되는 가상의 함수
' ************************************************************

Private Sub OnDoMotion()

        Dim fPos2x As Double
        Dim fPos2y As Double
        Dim fPos3x As Double
        Dim fPos3y As Double

        ' MAP0 를 마스터 속도 모드, Trapezoidal 속도 패턴으로 작업속도의 100%,
        ' 가속도의 70%, 감속도의 70%로 설정 합니다.
        Call cmmIxSetSpeedPattern(MAP0, cmFALSE, cmSMODE_T, 100, 70, 70 )

        ' 현재 위치를 (0, 0) 이라고 가정, 2 번째 및 3 번째 지나는 점 설정 및 이송 시작
        ' 끝 각은 270 도로 설정
        fPos2x=1000
        fPos2y=1000
        fPos3x=2000
        fPos3y=2000
        Call cmmIxArc3PTo(MAP0, fPos2x, fPos2y, fPos3x, fPos3y, 270, cmFALSE)

End Sub
```

Delphi

```
//************************************************************
// OnSetSpeed : 이 함수는 속도설정의 변경이 필요할 때
// 호출되는 가상의 함수 입니다.
//************************************************************

Const MAP0 = 0;  //맵번호 (0)

procedure OnSetSpeed();
begin
        cmmIxMapAxes(MAP0, cmX1_MASK or cmY1_MASK, 0 );

        //또는 cmmIxMapAxes(MAP0, $3, $0);
        //보간 이동할 축들의 기본속도를 설정합니다.
        cmmCfgSetSpeedPattern(cmX1, cmSMODE_T, 1000, 5000, 5000);
        cmmCfgSetSpeedPattern(cmY1, cmSMODE_T, 1000, 5000, 5000);
end;

//************************************************************
// OnDoMotion() : 작업명령시에 호출되는 가상의 함수
//************************************************************

procedure OnDoMotion();
var
        fPos2x : Double;
        fPos2y : Double;
```

```
        fPos3x : Double;
        fPos3y : Double;

begin
        //MAP0 를 마스터 속도 모드, Trapezoidal 속도 패턴으로 작업속도의 100%,
        //가속도의 70%, 감속도의 70%로 설정 합니다.
        cmmIxSetSpeedPattern(MAP0, cmFALSE, cmSMODE_T, 100, 70, 70 );

        // 현재 위치를 (0, 0) 이라고 가정, 2 번째 및 3 번째 지나는 점 설정 및 이송 시작
        // 끝 각은 270 도로 설정
        fPos2x :=1000;
        fPos2y :=1000;
        fPos3x :=2000;
        fPos3y :=2000;
        cmmIxArc3PTo(MAP0, fPos2x, fPos2y, fPos3x, fPos3y, 270, cmFALSE);
end;
```

NAME

cmmIxArc3PToStart
– 3 점 사용 원호 보간 절대좌표 이송 (현재 좌표 및 절대적 2 차 좌표와 3 차 좌표 사용)

INFORMATION

Interpolation Motion

VC++/VB

BCB/Delphi/.NET

Level 4

☺ 이송 함수

실제 이송이 진행되는 함수이므로, 사전에 반드시 안전을 확인합니다

SYNOPSIS

❒VT_I4 cmmIxArc3PToStart

([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

본 함수는 원호의 중심점 및 종점 좌표를 사용하는 cmmIxArcPStart(상대 좌표) 또는 cmmIxArcPToStart (절대 좌표) 함수와는 달리 현재 좌표(1 차)를 포함해서 2 차 및 3 차 좌표의 총 3 개의 (X, Y) 순서쌍 좌표를 사용해서 원호 보간을 수행합니다. 함수 이름에 'To' 가 포함되므로 절대적 위치로 원호 보간 이송이 됩니다.
결과적으로 세 점을 지나게 되므로 중심점은 자동으로 결정됩니다.
또한 함수 이름이 'Start' 로 끝나므로 함수 내부에서 이송 완료시까지 기다리지 않고 이송 작업만 시작시킨 뒤 바로 반환 됩니다. 따라서 cmmIxWaitDone 함수와 함께 사용하셔야 이송 완료 체크를 하실 수 있습니다.

PARAMETER

▶ MapIndex : 맵번호(Map index), 이 맵번호를 사용하기전에 먼저 cmmIxMapAxes() 함수를 통하여 해당 맵번호에 유효한 축들이 맵핑되어 있어야 합니다.

▶ VT_R8P2X : 원호가 지나는 2 번째 점의 X 좌표 입니다.

▶ VT_R8P2Y : 원호가 지나는 2 번째 점의 Y 좌표 입니다.

▶ VT_R8P3X : 원호가 지나는 3 번째 점의 X 좌표 입니다.

▶ VT_R8P3Y : 원호가 지나는 3 번째 점의 Y 좌표 입니다.

▶ EndAngle : 중심점과 출발점을 잇는 선분이 중심점과 종점을 잇는 선분과 이루는 각도를 의미 합니다.

Value	Meaning
0 도	3 번째 점에서 보간 이송 완료 하도록 설정. 이 경우 종점도 3 번째 점이 됩니다.
0 도 초과 360 도 이하	현재 좌표(1 번째), 2 번째 및 3 번째를 지나 중심점으로부터 해당 각도 만큼 원호 보간 이송이 수행됩니다. 단, 각도가 2 번째 또는 3 번째 점을 지나기에 모자라는 경우는 해당 각도 까지만 그려지고 2 번째 및 3 번째 점은 지나지 않게 됩니다.

SEE ALSO

cmmIxArcP
cmmIxArcPTo
cmmIxArcPStart
cmmIxArcPToStart
cmmIxArc3PTo

RETURN VALUE

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

REFERENCE

이 함수 사용 시 현재 좌표를 포함한 3 개의 점이 거의 직선에 가깝게 설정되는 경우는 사실상 지름이 무한대 거나 상당히 큰 원을 그려야 하는 문제가 발생하며, 이 경우는 함수 수행은 되고 반환값이 cmERR_NONE 이 되더라도 실제로는 제대로 이송이 안되고 원호 보간 이송에 실패할 수 있음에 유의해야 합니다.

EXAMPLE

C/C++

```
/*************************************************************
* OnSetSpeed : 이 함수는 속도설정의 변경이 필요할 때
* 호출되는 가상의 함수 입니다.
* ************************************************************/
#define MAP0 0 //맵번호 (0)

void OnSetSpeed()
{
	cmmIxMapAxes(MAP0, cmX1_MASK | cmY1_MASK, 0 );

	//또는 cmmIxMapAxes(MAP0, 0x3, 0x0);
	//보간 이동할 축들의 기본속도를 설정합니다.
	cmmCfgSetSpeedPattern(cmX1, cmSMODE_T, 1000, 5000, 5000);
	cmmCfgSetSpeedPattern(cmY1, cmSMODE_T, 1000, 5000, 5000);
}

/*************************************************************
* OnDoMotion() : 작업명령시에 호출되는 가상의 함수
*************************************************************/
void OnDoMotion()
{
	double fPos2x, fPos2y, fPos3x, fPos3y;

	//MAP0 를 마스터 속도 모드, Trapezoidal 속도 패턴으로 작업속도의 100%,
	//가속도의 70%, 감속도의 70%로 설정 합니다.
	cmmIxSetSpeedPattern(MAP0, cmFALSE, cmSMODE_T, 100, 70, 70 );

	// 현재 위치를 (0, 0) 이라고 가정, 2 번째 및 3 번째 지나는 점 설정 및 이송 시작
	// 끝 각은 270 도로 설정
	fPos2x=1000; fPos2y=1000; fPos3x=2000; fPos3y=2000;
	cmmIxArc3PToStart(MAP0, fPos2x, fPos2y, fPos3x, fPos3y, 270);
}
```

Visual Basic

```
' *************************************************************
' OnSetSpeed : 이 함수는 속도설정의 변경이 필요할 때
' 호출되는 가상의 함수 입니다.
' *************************************************************

Const MAP0 = 0 '맵번호 (0)

Private Sub OnSetSpeed()

	Call cmmIxMapAxes(MAP0, cmX1_MASK Or cmY1_MASK, 0 )
```

```
        ' 또는 Call cmmIxMapAxes(MAP0, &H3, &H0);
        ' 보간 이동할 축들의 기본속도를 설정합니다.
        Call cmmCfgSetSpeedPattern(cmX1, cmSMODE_T, 1000, 5000, 5000)
        Call cmmCfgSetSpeedPattern(cmY1, cmSMODE_T, 1000, 5000, 5000)

End Sub

' ************************************************************
' OnDoMotion() : 작업명령시에 호출되는 가상의 함수
' ************************************************************

Private Sub OnDoMotion()

        Dim fPos2x As Double
        Dim fPos2y As Double
        Dim fPos3x As Double
        Dim fPos3y As Double

        ' MAP0 를 마스터 속도 모드, Trapezoidal 속도 패턴으로 작업속도의 100%,
        ' 가속도의 70%, 감속도의 70%로 설정 합니다.
        Call cmmIxSetSpeedPattern(MAP0, cmFALSE, cmSMODE_T, 100, 70, 70 )

        ' 현재 위치를 (0, 0) 이라고 가정, 2 번째 및 3 번째 지나는 점 설정 및 이송 시작
        ' 끝 각은 270 도로 설정
        fPos2x=1000
        fPos2y=1000
        fPos3x=2000
        fPos3y=2000
        Call cmmIxArc3PToStart(MAP0, fPos2x, fPos2y, fPos3x, fPos3y, 270)

End Sub
```

Delphi

```
//************************************************************
// OnSetSpeed : 이 함수는 속도설정의 변경이 필요할 때
// 호출되는 가상의 함수 입니다.
//************************************************************

Const MAP0 = 0;  //맵번호 (0)

procedure OnSetSpeed();
begin
        cmmIxMapAxes(MAP0, cmX1_MASK or cmY1_MASK, 0 );

        //또는 cmmIxMapAxes(MAP0, $3, $0);
        //보간 이동할 축들의 기본속도를 설정합니다.
        cmmCfgSetSpeedPattern(cmX1, cmSMODE_T, 1000, 5000, 5000);
        cmmCfgSetSpeedPattern(cmY1, cmSMODE_T, 1000, 5000, 5000);
end;

//************************************************************
// OnDoMotion() : 작업명령시에 호출되는 가상의 함수
//************************************************************

procedure OnDoMotion();
var
        fPos2x : Double;
        fPos2y : Double;
        fPos3x : Double;
        fPos3y : Double;

begin
        //MAP0 를 마스터 속도 모드, Trapezoidal 속도 패턴으로 작업속도의 100%,
        //가속도의 70%, 감속도의 70%로 설정 합니다.
        cmmIxSetSpeedPattern(MAP0, cmFALSE, cmSMODE_T, 100, 70, 70 );

        // 현재 위치를 (0, 0) 이라고 가정, 2 번째 및 3 번째 지나는 점 설정 및 이송 시작
        // 끝 각은 270 도로 설정
```

```
        fPos2x :=1000;
        fPos2y :=1000;
        fPos3x :=2000;
        fPos3y :=2000;
        cmmIxArc3PToStart(MAP0, fPos2x, fPos2y, fPos3x, fPos3y, 270);
end;
```

NAME

cmmIxIsDone
– 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 cmmIxIsDone ([in] VT_I4 MapIndex, [out] VT_PI4 IsDone)

DESCRIPTION

지정한 보간맵에 해당하는 보간작업이 완료됐는지를 확인(確認)합니다.

PARAMETER

▶ MapIndex : 맵번호(Map index), 이 맵번호를 사용하기전에 먼저 cmmIxMapAxes() 함수를 통하여 해당 맵번호에 유효한 축들이 맵핑되어 있어야 합니다.

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

Value	Meaning
0 또는 cmFALSE	모션작업이 완료되지 않음
1 또는 cmTRUE	모션작업이 완료됨

RETURN VALUE

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

EXAMPLE

C/C++

```
#define MAP0 0 //맵번호 (0)

cmmIxMapAxes(MAP0, cmX1_MASK | cmY1_MASK, 0 );
//또는 cmmIxMapAxes(MAP0, 0x3, 0x0);
//보간 이동할 축들의 기본속도를 설정합니다.

...//속도 패턴 설정

long nIsDone = 0;
double fDistList[2] = {1000, 1000};

if(cmmIxLine(MAP0, fDistList, cmFALSE) != cmERR_NONE){
        cmmErrShowLast(Handle);//에러처리
        return;
}

while (1){
        cmmIxIsDone(MAP0, &nIsDone );
        if(nIsDone == cmTRUE) break;
        else{
```

```
                ...
        }
}
```

Visual Basic

```
Dim fDistList As Double
Dim nIsDone As Long

If (cmmIxLineStart(MAP0, fDistList(0)) = cmFALSE) Then
        Call cmmErrShowLast(Handle)  '마지막에 발생한 에러 메시지를 화면에 표시합니다.
        Exit Sub
End If

While (cmmIxIsDone(0, nIsDone) = cmFALSE)
…
end
If (Not (cmmErrGetLastCode(nErrCode) = cmERR_NONE)) Then
        Call cmmErrShowLast(Handle) '마지막에 발생한 에러 메시지를 화면에 표시합니다.
        Exit Sub
End If
```

Delphi

```
procedure OnIxMove ();
var
        nIxMap : LongInt;                       // 보간 맵 설정
        nIsDone : LongInt;                      // 직선 보간 이송 완료 정보.
        fDistList : Array[0..1] of Double;

begin
        nIxMap := 0;

        fDistList[0] := 10000;
        fDistList[1] := 20000;

        cmmIxMapAxes ( nIxMap , cmX1_MASK or cmY1_MASK, 0 );

        // 직선 보간 이송을 시작합니다.
        cmmIxLineStart ( nIxMap , @fDistList );

        nIsDone := cmFALSE;

        while nIsDone = cmFALSE do
        begin
                cmmIxIsDone ( nIxMap, @nIsDone );       // 직선 보간 이송 완료 여부를 확인합니다.
        end;

end;
```

NAME

cmmIxWaitDone
- 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 cmmIxWaitDone ([in] VT_I4 MapIndex, [in] VT_I4 IsBlocking)

DESCRIPTION

지정한 보간맵에 해당하는 보간작업이 완료(完了)될 때까지 기다립니다.

PARAMETER

▶MapIndex : 맵번호(Map index), 이 맵번호를 사용하기전에 먼저 cmmIxMapAxes() 함수를 통하여 해당 맵번호에 유효한 축들이 맵핑되어 있어야 합니다.

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

Value	Meaning
cmFALSE	블록(Blocking)을 하지 않습니다. 따라서 해당 모션이 완료되는 동안에도 윈도우 이벤트를 처리합니다.
cmTRUE	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
cmERR_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)

cmmIxMapAxes(MAP0, cmX1_MASK | cmY1_MASK, 0 );
//또는 cmmIxMapAxes(MAP0, 0x3, 0x0);
//보간 이동할 축들의 기본속도를 설정합니다.

...//속도 패턴 설정

long nIsDone = 0;
double fDistList[2] = {1000, 1000};

if(cmmIxLine(MAP0, fDistList, cmFALSE) != cmERR_NONE){
        cmmErrShowLast(Handle);//에러처리
        return;
}

//모션이 완료 될 때 까지 기다립니다.
if(cmmIxWaitDone(MAP0, cmFALSE) != cmERR_NONE){
        cmmErrShowLast(Handle);//에러처리
        return ;
}
```

Visual Basic

```
Private Sub OnIxMove ()

        Dim nIxMap As Long ' 보간 맵 설정
        Dim nIsDone As Long             ' 직선 보간 이송 완료 정보.
        Dim fDistList(2) As Double

        nIxMap = 0

        fDistList(0) = 10000
        fDistList(1) = 20000

        Call cmmIxMapAxes ( nIxMap, cmX1_MASK Or cmY1_MASK, 0 )

        ' 직선 보간 이송을 시작합니다.
        If cmmIxLineStart ( MAP0 , fDistList(0) ) = ceERR_NONE Then
                Call cmmIxWaitDone ( nIxMap, cmFALSE ) ' 모션이 완료 시까지 기다립니다.
        End If

End Sub
```

Delphi

```
procedure OnIxMove ();
var
        nIxMap : LongInt;                       // 보간 맵 설정
        nIsDone : LongInt;                      // 직선 보간 이송 완료 정보.
        fDistList : Array[0..1] of Double;

begin
        nIxMap := 0;

        fDistList[0] := 10000;
        fDistList[1] := 20000;

        cmmIxMapAxes ( nIxMap, cmX1_MASK or cmY1_MASK, 0 );

        // 직선 보간 이송을 시작합니다.
        if cmmIxLineStart ( MAP0 , @fDistList ) = cmERR_NONE then
        begin
                cmmIxWaitDone ( nIxMap, cmFALSE );      // 모션이 완료 시까지 기다립니다.
        end;

end;
```

NAME

cmmIxStop
cmmIxStopEmg
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)(비상 정지(停止)를 주의하십시요. 기구물의 손상이나 안전사고에 원인이 될 수 있습니다.

SYNOPSIS

❒ VT_I4 cmmIxStop ([in] VT_I4 MapIndex, [in] VT_I4 IsWaitComplete, [in] VT_I4 IsBlocking)

❒ VT_I4 cmmIxStopEmg ([in] VT_I4 MapIndex)

DESCRIPTION

지정한 보간맵에 대한 보간작업을 정지(停止)합니다. 정지(停止)시에 cmmIxStop() 함수를 사용하면 감속 후 정지(停止)하며, cmmIxStopEmg()를 사용하면 감속없이 즉시정지(停止)를 수행합니다.

PARAMETER

▶ MapIndex : 맵번호(Map index), 이 맵번호를 사용하기전에 먼저 cmmIxMapAxes() 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
cmFALSE	블록(Blocking)을 하지 않습니다. 따라서 해당 모션이 완료되는 동안에도 윈도우 이벤트를 처리합니다.
cmTRUE	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
cmERR_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 스텝 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

Void OnStop() {
        if(cmmIxStop(MAP0, cmTRUE, cmFALSE) != cmERR_NONE){
                cmmErrShowLast(Handle);
}
```

Visual Basic

```
‘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
        cmmIxStop(MAP0, cmTRUE, cmFALSE)
end
```

Delphi

```
Const
MAPINDEX = 0;

// * Description :
// *
// *  현재 수행되고 있는 모션 동작에 대해서 감속 후 정지(停止) 합니다.
procedure btnStopClick();
begin
      cmmIxStop(MAPINDEX,cmTRUE, cmFALSE);
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 Counter는 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 신호

엔코더의 제로 펄스 신호를 의미합니다. 이 신호는 원점 복귀 모드에 따라 ORG신호 또는 EL신호와 함께 사용되어 보다 정밀한 원점복귀 작업을 수행할 수 있도록 해줍니다. 이 신호는 'EZ+'단자와 'EZ-'단자를 통하여 입력되어야 합니다.

□ EL 신호

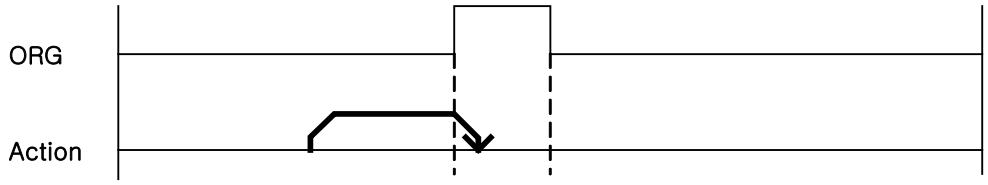
기계적인 리미트(Limit) 신호를 의미합니다. 이 신호는 일반적으로 구조물이 움직일 수 있는 한계점을 감지하기 위해 사용되나 원점 복귀 모드에 따라 ORG신호의 대용으로도 사용될 수 있습니다. EL신호는 (+)방향 리미트 신호와 (-)방향 리미트 신호의 두 가지 신호가 있습니다. (+)방향 리미트 신호는 '+EL'단자, 그리고 (-)방향 리미트 신호는'-EL'단자를 통하여 입력되어야 합니다.

8.4.1 원점복귀모드 안내

㈜커미조아 모션제어보드는 다음과 같이 13 가지의 다양한 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 0 : ORG ON => Stop

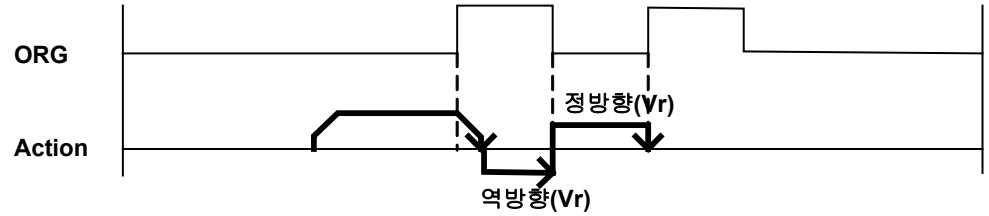
MODE 0 에서는 ORG신호가 OFF에서 ON으로 바뀌는 순간에 모션을 감속 후 정지(停止)하고 복귀 작업을 종료합니다.



□ MODE 1 : ORG ON => Stop => Back (Vr) => ORG OFF => Foward(Vr) => ORG ON => Stop

MODE 1 에서는 ORG신호가 OFF에서 ON으로 바뀌는 순간에 모션을 감속 후 정지(停止)한 후 ORG신호가 OFF가 될때까지 Vr(Reverse Speed)의 속도로 역방향 회전을 수행합니다. ORG신호가 OFF되면 다시 Vr 의 속도로 정방향 회전을 수행하다가 ORG신호가 다시 ON되는 순간에 복귀작업을 종료합니다.

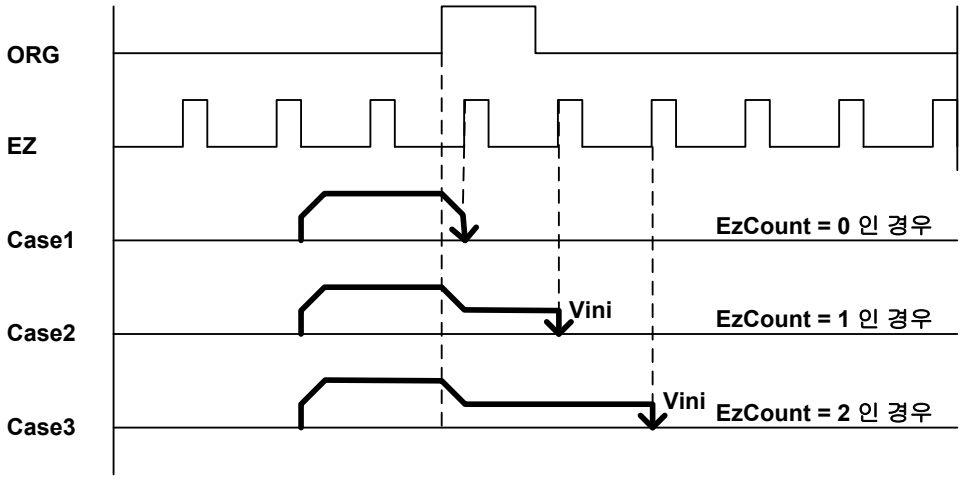
이 모드를 사용할 때 주의할 것은 처음 ORG 신호가 ON으로 변경되어 감속후 정지(停止)하는 도중에 ORG 센서의 ON으로 유지되는 시간이 짧아서 이미 OFF상태로 변경되면 역방향 이동을 생략하고 최종적으로 원점센서를 찾으려는 단계로 넘어갑니다. 따라서 이러한 경우에는 (-)Limit 위치까지 이동하게 됩니다. 이러한 경우에도 자동으로 다시 탈출하여 정상적인 원점복귀 작업을 다시 수행하지만 의도하지 않게 원점복귀 시간이 길어질 수 있습니다. 이를 방지하는 방법은 구조적으로 ORG 신호가 ON으로 유지되는 시간을 길게해주거나 또는 원점복귀시의 작업속도를 낮추거나 감속도를 크게해주어 감속시 이동되는 거리를 짧게해주면 됩니다.



Vr : Reverse Speed를 의미합니다.

☐ MODE 2 : ORG ON => Slow Down (Vini) => Stop on EZ Count

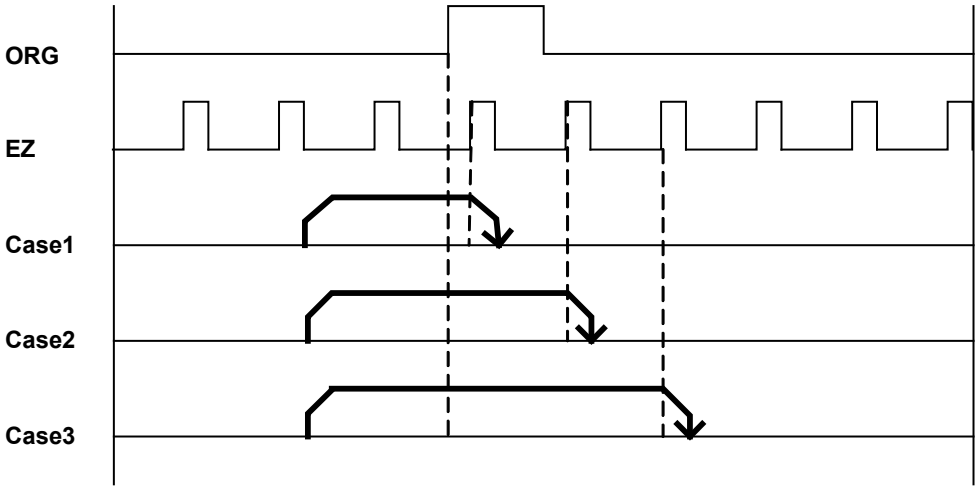
MODE 2 에서는 ORG신호가 OFF에서 ON으로 바뀌는 순간에 모션을 감속한 후 초기속도값으로 모션을 진행하다가 EZ 신호에 따라 복귀작업을 종료합니다. cmmHomeSetConfig 함수를 통하여 미리 설정된 EzCount값에 따라 종료하는 시점은 아래와같이 달라집니다.



Vini : 초기속도를 의미하며, cmmSxOptSetIniSpeed 함수 참조

☐ MODE 3 : ORG => Stop on EZ Count

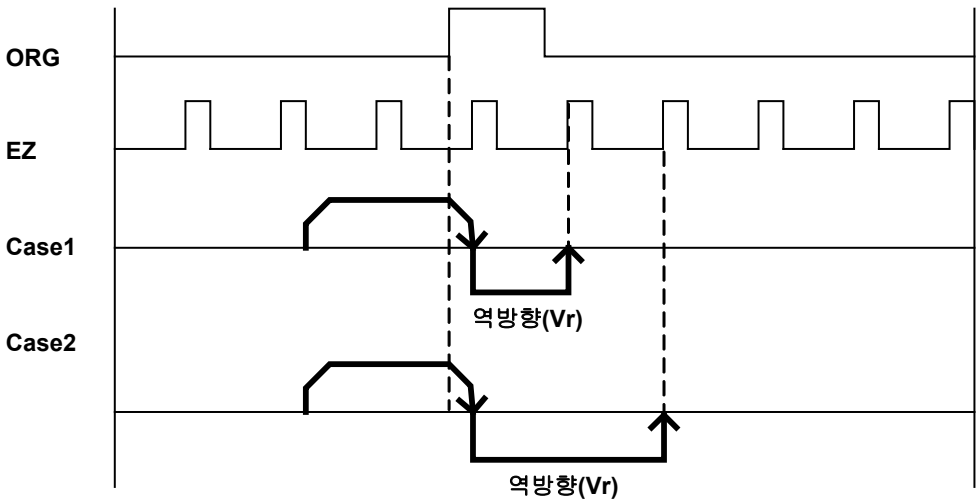
MODE 3 에서는 ORG신호가 OFF에서 ON으로 바뀌고 난 후 발생하는 EZ 신호에 따라 감속 후 정지(停止)합니다. cmmHomeSetConfig() 함수를 통하여 미리 설정된 EzCount값에 따라 감속을 시작하는 시점은 아래와같이 달라집니다.



Case1 : EzCount = 0 인 경우
Case2 : EzCount = 1 인 경우
Case3 : EzCount = 2 인 경우

□ MODE 4 : ORG ON => Stop => Back (Vr) => Stop on EZ Count

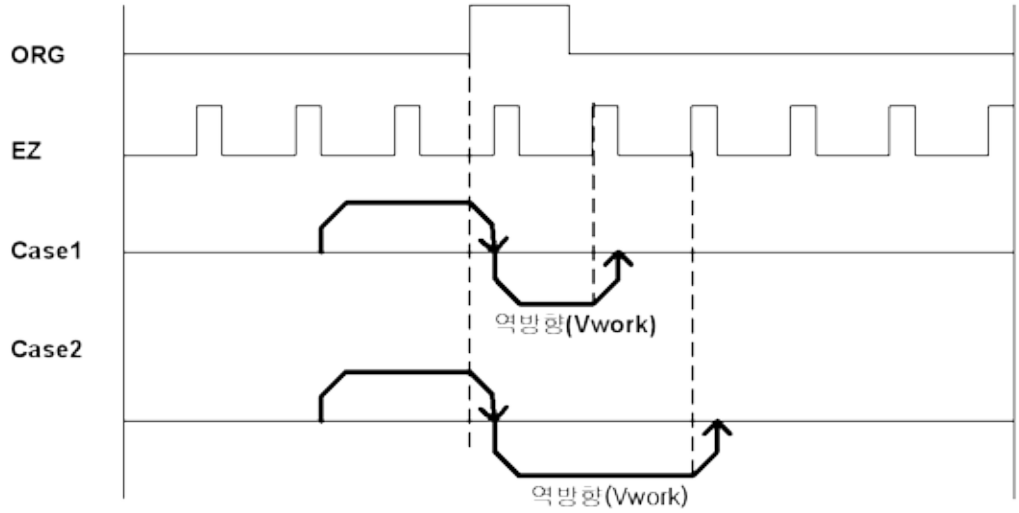
MODE 4 에서는 ORG신호가 OFF에서 ON으로 바뀌는 순간 감속 후 정지(停止)합니다. 그리고 Vr의 속도로 역방향 회전한 후 EZ 신호에 따라 복귀 작업을 종료합니다.



Case1 : EzCount = 1 인 경우
Case2 : EzCount = 2 인 경우
Vr : Reverse Speed

☐ MODE 5 : ORG ON => Stop => Back (Vwork) => Stop on EZ Count

MODE 5 에서는 ORG신호가 OFF에서 ON으로 바뀌는 순간 감속 후 정지(停止)합니다. 그리고 작업속도까지 가속하여 역방향 회전한 후 EZ 신호에 따라 감속 후 복귀 작업을 종료합니다.

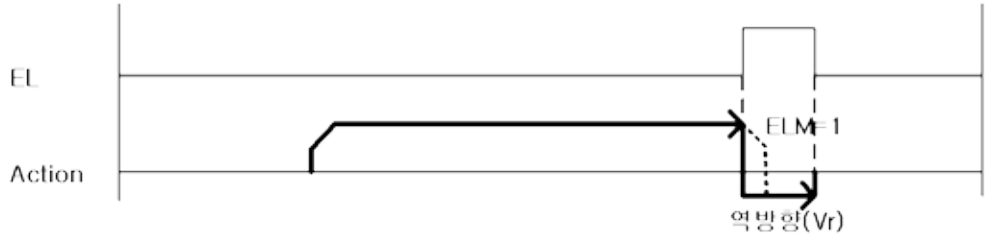


Case1 : EzCount = 1 인 경우, cmmHomeSetConfig 함수 참조
Case2 : EzCount = 2 인 경우, cmmHomeSetConfig 함수 참조

☐ MODE 6 : EL ON => Stop => Back (Vr) => EL OFF => Stop

MODE 6 에서는 EL신호가 ON으로 바뀌는 순간 즉시 정지(停止)(또는 ELM=1 인 경우에 감속후 정지(停止)합니다. 그리고 반대 방향으로 Vr 속도로 회전하다가 EL 신호가 OFF되는 순간에 복귀작업을 종료합니다. 여기서 ELM=1 은 EL의 "Stop mode"가 "감속후 정지(停止)" 모드로 설정됨을 의미합니다.

cmmCfgSetMioProperty 함수를 통해, EL 신호가 ON 으로 변경되는 순간 모션을 즉시 정지를 할 것인지 감속 후 정지를 할 것인지를 설정할 수 있습니다. 자세한 설명은 cmmCfgSetMioProperty 함수를 통해 확인해 주시기 바랍니다.

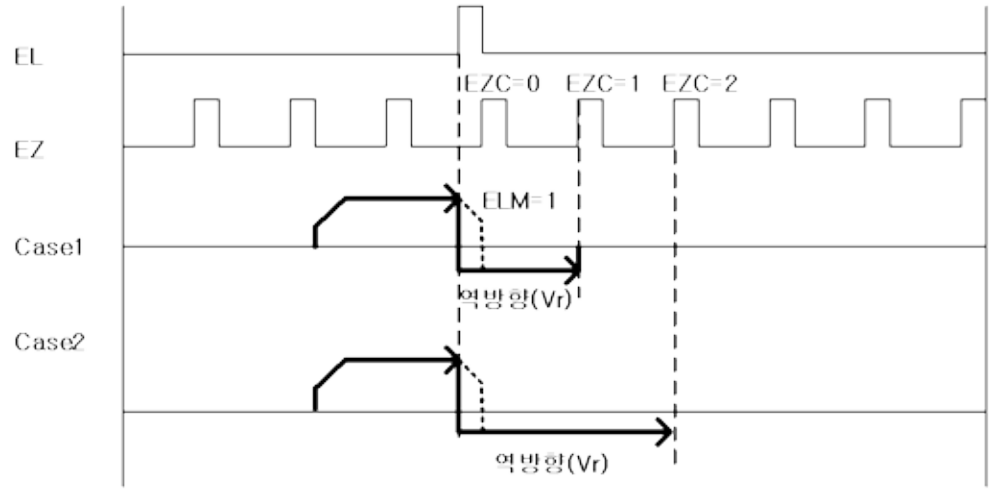


Vr : Reverse Speed

☐ MODE 7 : EL ON => Stop => Back (Vr) => Stop on EZ count

MODE 7 에서는 EL신호가 ON으로 바뀌는 순간 즉시 정지(停止)(또는 ELM=1 인 경우에 감속후 정지(停止)합니다. 그리고 반대 방향으로 Vr 속도로 회전하다가 EZ_Count에 따라 복귀작업을 종료합니다. 여기서 ELM=1 은 EL의 "Stop mode"가 "감속후 정지(停止)" 모드로 설정됨을 의미합니다.

안내	cmmCfgSetMioProperty 함수를 통해, EL 신호가 ON 으로 변경되는 순간 모션을 즉시 정지를 할 것인지 감속 후 정지를 할 것인지를 설정할 수 있습니다. 자세한 설명은 cmmCfgSetMioProperty 함수를 통해 확인해 주시기 바랍니다.

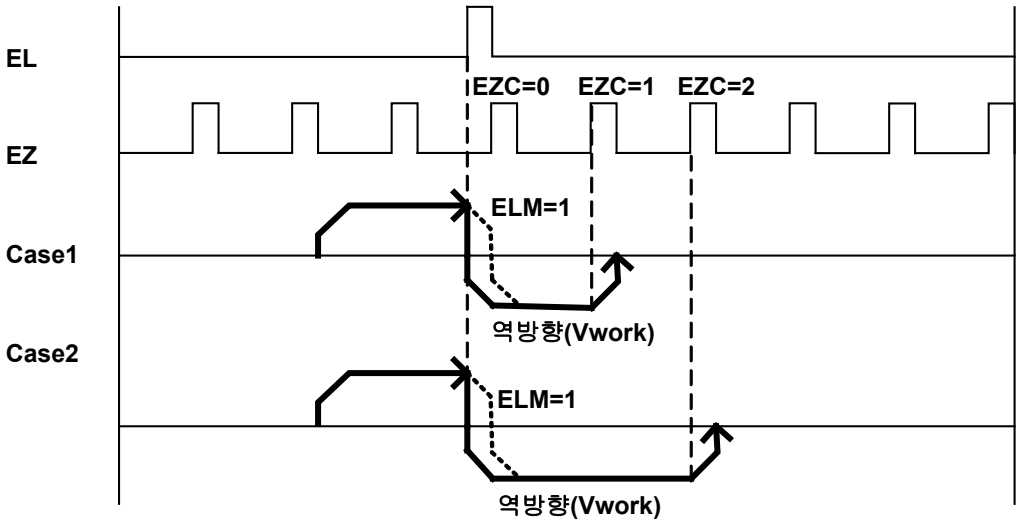


Case1 : EzCount = 1 인 경우
Case2 : EzCount = 2 인 경우
Vr : Reverse Speed

☐ MODE 8 : EL ON => Stop => Back (Vwork) => Stop on EZ count

MODE 8 에서는 EL신호가 ON으로 바뀌는 순간 즉시 정지(停止)(또는 ELM=1 인 경우에 감속후 정지(停止)합니다. 그리고 작업속도까지 가속하여 반대 방향으로 회전한 후 EZ 신호에 따라 감속 후 복귀 작업을 종료합니다. 여기서 ELM=1 은 EL의 "Stop mode"가 "감속후 정지(停止)" 모드로 설정됨을 의미합니다.

안내	cmmCfgSetMioProperty 함수를 통해, EL 신호가 ON 으로 변경되는 순간 모션을 즉시 정지를 할 것인지 감속 후 정지를 할 것인지를 설정할 수 있습니다. 자세한 설명은 cmmCfgSetMioProperty 함수를 통해 확인해 주시기 바랍니다.



Case1 : EzCount = 1 인 경우
Case2 : EzCount = 2 인 경우
Vwork : 작업속도를 의미하며 cmmHomeSetSpeedPattern 함수 참조

☐ MODE 9 : MODE0 => Operate till dev. counter 0

MODE 9 에서는 MODE 0 과 동일한 복귀 작업을 수행합니다. 그리고 난후, 편차카운터(Deviation Counter)가 0 이 되도록 모션을 다시 취한 후에 복귀 작업을 종료합니다.

☐ MODE 10 : MODE3 => Operate till dev. counter 0

MODE 10 에서는 MODE 3 과 동일한 복귀 작업을 수행합니다. 그리고 난후 편차카운터(Deviation Counter)가 0 이 되도록 모션을 다시 취한 후에 복귀 작업을 종료합니다.

☐ MODE 11 : MODE5 => Operate till dev. counter 0

MODE 11 에서는 MODE 5 과 동일한 복귀 작업을 수행합니다. 그리고 난후 편차카운터(Deviation Counter)가 0 이 되도록 모션을 다시 취한 후에 복귀 작업을 종료합니다.

☐ MODE 12 : MODE8 > Operate till dev. counter 0

MODE 12 에서는 MODE 8 과 동일한 복귀 작업을 수행합니다. 그리고 난후 편차카운터(Deviation Counter)가 0 이 되도록 모션을 다시 취한 후에 복귀 작업을 종료합니다.

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 ON이 되어 있는 위치에 있거나 원점센서와 (-)EL 센서 사이에 위치한 경우에는 원점센서를 기준으로 (+) 방향의 위치까지 탈출한 후에 정상적인 원점복귀 작업을 수행하여야 합니다. 이러한 기능을 "자동원점검색" 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 ON인 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 ON되면 정지(停止) 후 양의 방향으로 이동합니다. 원점센서가 ON되면 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.

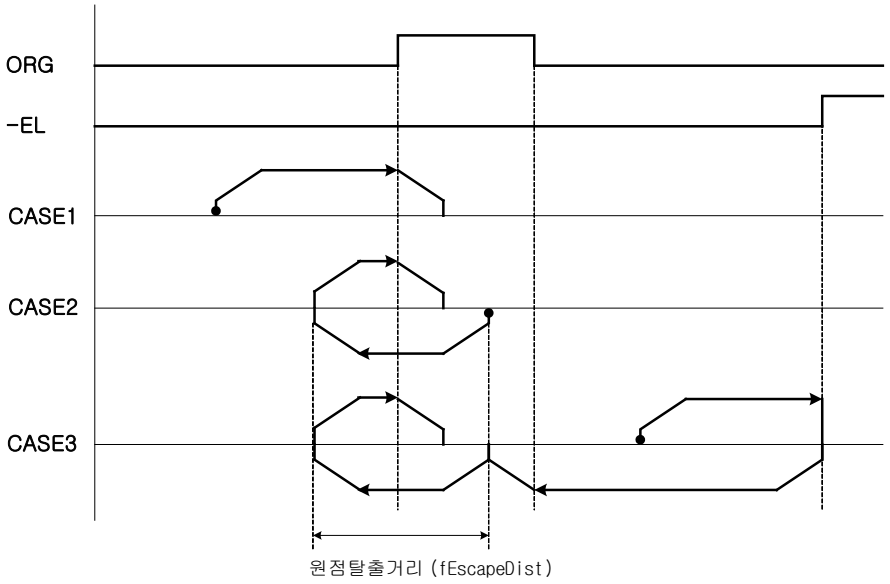


그림 8-2 기구물과 센서의 위치에 따른 원점복귀 작업

8.4.3 함수 요약

원점복귀와 관련된 함수들은 아래의 표와 같습니다. 그리고 속도패턴 설정은 cmmHomeSetSpeedPattern 함수를 사용합니다.

Summary of Functions
❒ VT_I4 cmmHomeSetConfig ([in] VT_I4 Axis, [in] VT_I4 HomeMode, [in] VT_I4 EzCount, [in] VT_I8 EscDist, [in] VT_R8 Offset) 대상(對象) 모션 채널에 대해서, 원점복귀(原點復歸)에 대한 환경설정(環境設定)을 구성합니다. 홈 복귀 모드와 Z 상 검출 횟수, 자동 원점 탈출 거리, 원점복귀(原點復歸) 완료 후 추가 이송 거리등을 설정할 수 있습니다.
❒ VT_I4 cmmHomeGetConfig ([in] VT_I4 Axis, [out] VT_PI4 HomeMode, [out] VT_PI4 EzCount, [out] VT_ PI8 EscDist, [out] VT_PR8 Offset) 대상(對象) 모션 채널에 대해서, 원점복귀(原點復歸)에 대해 설정되어 있는 환경 설정을 반환합니다. 원점 복귀 모드와 Z 상 검출 횟수, 자동 원점 탈출 거리, 원점 복귀 완료 후 추가 이송 거리등이 반환됩니다.
❒ VT_I4 cmmHomeSetPosClrMode ([in] VT_I4 Axis, [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 cmmHomeGetPosClrMode ([in] VT_I4 Axis, [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 cmmHomeSetSpeedPattern ([in] VT_I4 Axis, [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 cmmHomeGetSpeedPattern ([in] VT_I4 Axis, [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 cmmHomeSetSpeedPattern_T ([in] VT_I4 Axis, [in] VT_I4 SpeedMode, [in] VT_R8 Vel, [in] VT_R8 AccelTime, [in] VT_R8 DecelTime, [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 cmmHomeGetSpeedPattern_T ([in] VT_I4 Axis, [out] VT_PI4 SpeedMode, [out] VT_PR8 Vel, [out] VT_PR8 AccelTime, [out] VT_PR8 DecelTime, [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 cmmHomeMove ([in] VT_I4 Axis, [in] VT_I4 Direction, [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 cmmHomeMoveStart ([in] VT_I4 Axis, [in] VT_I4 Direction) 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 cmmHomeMoveAll ([in] VT_I4 NumAxes, [in] VT_PI4 AxisList, [in] VT_PI4 DirList, [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 cmmHomeMoveAllStart ([in] VT_I4 NumAxes, [in] VT_PI4 AxisList, [in] VT_PI4 DirList) 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 cmmHomeIsBusy ([in] VT_I4 Axis, [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 cmmHomeWaitDone ([in] VT_I4 Axis, [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 cmmHomeGetSuccess ([in] VT_I4 Axis, [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 cmmHomeSetSuccess ([in] VT_I4 Axis, [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

cmmHomeSetConfig
cmmHomeGetConfig
– 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 cmmHomeSetConfig

([in] VT_I4 Axis, [in] VT_I4 HomeMode, [in] VT_I4 EzCount, [in] VT_I8 EscDist, [in] VT_R8 Offset)

❒ VT_I4 cmmHomeGetConfig

([in] VT_I4 Axis, [out] VT_PI4 HomeMode, [out] VT_PI4 EzCount, [out] VT_PI8 EscDist,

[out] VT_PR8 Offset)

DESCRIPTION

cmmHomeSetConfig() 함수는 원점복귀에 관련된 여러가지 환경을 설정합니다. 그러나 이러한 모든 설정은 속성대화상자를 통하여 수행할 수 있으므로 특별한 경우가 아니면 사용할 필요가 없습니다.
cmmHomeGetConfig() 함수는 원점복귀 작업에 관련된 환경 설정값을 확인(確認)합니다.

PARAMETER

▶ Axis: 축번호. 축번호는 상수값으로 [cmX1] 부터 0 번째 축을 기준 축으로 임의의 축을 설정할 수 있습니다.

▶ HomeMode : cmmHomeSetConfig 함수의 인자이며, 원점복귀 모드 번호를 설정합니다. 앞서 설명한 바와 같이 ㈜커미조아 모션 제어보드는 13 가지(0 ~ 12)의 다양한 원점복귀 모드를 제공합니다.

▶ HomeMode : cmmHomeGetConfig 함수의 인자이며, 원점복귀 모드 번호를 반환합니다.

▶ EzCount : cmmHomeSetConfig 함수의 인자이며, 이 값은 ORG 신호 또는 EL 신호가 ON 이 된 후 실제로 복귀 작업을 완료하는데 필요한 EZ Count 값을 0 ~ 15 사이의 값으로 설정합니다. 이 값의 참조 여부는 원점복귀 모드에 따라서 다릅니다.

▶ EzCount : cmmHomeGetConfig 함수의 인자이며, EZ Count 값을 0 ~ 15 사이의 값으로 반환합니다.

▶ EscDist : cmmHomeSetConfig 함수의 인자이며, 원점탈출거리를 지정합니다. 거리의 단위는 논리적 거리 단위를 사용합니다.

▶ EscDist : cmmHomeGetConfig 함수의 인자이며, 원점탈출거리를 반환합니다. 거리의 단위는 논리적 거리 단위를 사용합니다.

▶ Offset : cmmHomeSetConfig 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 : cmmHomeGetConfig 함수의 인자이며, Offset 으로 설정된 상대 거리 이동값을 반환합니다.

RETURN VALUE

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

EXAMPLE

C/C++

```
#include "Cmmsdk.h"
#include "CmmsdkDef.h"


void OnSetHomeConfig ()
{
	long nHomeMode = 0;		// 원점 복귀 모드 설정. 0 ~ 12 번 모드가 있습니다.
	long nHomeDir = cmDIR_N;	// 원점 복귀 방향. cmDIR_N: (-) 방향, cmDIR_P: (+) 방향
	long nEzCount = 0;		// Encoder Z 상 카운트. '0' 은 EZ 상 1 회 카운트를 의미합니다.
	double fEscDist = 10.0f;	// 원점 탈출 거리. 자동 원점 검색 기능에 사용되며,
					// 최소 '1' 이상의 값이어야 합니다.
	double fOffset = 0.0f;		// 원점 복귀 완료 후 Offset 값 (상대 거리)


	if (cmmHomeSetConfig (cmX1, nHomeMode, nEzCount, fEscDist, fOffset ) != cmERR_NONE )
	{
		OutputDebugString ( "cmmHomeGetConfig has been failed" );
	}

	// cmmHomeGetConfig () 함수로 설정되어있는 원점 복귀 환경 설정 정보를 반환합니다.
	// cmmHomeGetConfig (cmX1, &nHomeMode, &nHomeDir, &nEzCount, &fEscDist, &fOffset );
}
```

Visual Basic

```
Private Sub OnSetHomeConfig ()

	Dim nHMode As Long			' 원점 복귀 모드 설정.
	Dim nEzCount As Long			' Encoder Z 상 카운트.
	Dim fEscDist As Double			' 원점 탈출 거리.
	Dim fOffset As Double			' 원점 복귀 완료 후 Offset 값 (상대 거리)

	nHomeMode = 0				' 원점 복귀 0 번 모드 설정
	nEzCount = 0				' Encoder Z 상 1 회 카운트.
	fEscDist = 10				' 최소 '1' 이상의 값으로 설정.
	fOffset = 0				' 원점 복귀 완료 후 Offset 값 설정.

	If cmmHomeSetConfig (cmX1, nHomeMode, nEzCount, fEscDist, fOffset ) <> cmERR_NONE Then
		MsgBox ( "cmmHomeGetConfig has been failed" )
	End If

End Sub
```

Delphi

```
procedure OnSetHomeConfig ();
var
nHomeMode, nHomeDir, nEzCount : LongInt;
fEscDist, fOffset : Double;

begin
nHomeMode := 0;		// 원점 복귀 모드 설정. 0 ~ 12 번 모드가 있습니다.
nEzCount := 0;		// Encoder Z 상 카운트. '0' 은 EZ 상 1 회 카운트를 의미합니다.
fEscDist := 10;		// 원점 탈출 거리. 자동 원점 검색 기능에 사용되며,
```

```
                                    // 최소 '1' 이상의 값이어야 합니다.
fOffset := 0;                       // 원점 복귀 완료 후 Offset 값 (상대 거리).

if cmmHomeSetConfig (cmX1, nHomeMode, nEzCount, fEscDist, fOffset ) <> cmERR_NONE then
begin
        ShowMessage ( 'cmmHomeGetConfig has been failed' );
end;

end;
```

NAME

cmmHomeSetPosClrMode
cmmHomeGetPosClrMode
– 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 cmmHomeSetPosClrMode

([in] VT_I4 Axis, [in] VT_I4 PosClrMode)

❒ VT_I4 cmmHomeGetPosClrMode

([in] VT_I4 Axis, [out] VT_PI4 PosClrMode)

DESCRIPTION

cmmHomeSetPosClrMode() 함수는 원점복귀가 완료된 후에 Command 및 Feedback 위치에 대한 처리 모드를 설정하는 함수입니다. cmmHomeGetConfig() 함수는 원점복귀가 완료된 후에 Command 및 Feedback 위치를 소거하는 모드의 설정을 읽어들이는 함수입니다.

PARAMETER

▶ Axis: 축번호. 축번호는 상수값으로 [cmX1] 부터 0 번째 축을 기준 축으로 임의의 축을 설정할 수 있습니다.

▶ PosClrMode : cmmHomeSetPosClrMode 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
cmHPCM_M0	최종 완료 조건에 해당하는 외부 하드웨어 신호가 입력되는 순간에 Command 및 Feedback 의 위치가 0 으로 소거됩니다. 일정량의 Feedback 위치 편차를 보입니다.
cmHPCM_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 위치가 모두 자동으로 0 으로 소거됩니다. 일정량의 Feedback 위치 편차를 보입니다.
cmHPCM_M2	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 : cmmHomeGetPosClrMode 함수의 인자이며, 원점복귀가 완료된 후에 Command 및 Feedback 위치가 클리어되는 모드를 반환합니다. PosClrMode 는 다음과 같이 3 가지 설정값을 갖습니다.

Value	Meaning
cmHPCM_M0	최종 완료 조건에 해당하는 외부 하드웨어 신호가 입력되는 순간에 Command 및 Feedback 의 위치가 0 으로 소거되는 모드입니다.
cmHPCM_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 위치가 모두 자동으로 0 으로 소거되는 모드입니다.

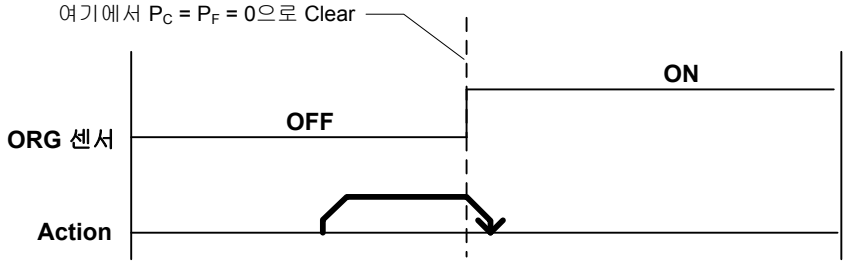
cmHPCM_M2	1 차적으로는 cmHPCM_M0 일 때와 동일하게 동작합니다. 단, 원점복귀가 완료된 후에 Feedback 위치와 동일한 값으로 Command 위치를 셋팅하므로써 Command 와 Feedback 을 일치하도록 동작시키는 모드입니다.

RETURN VALUE

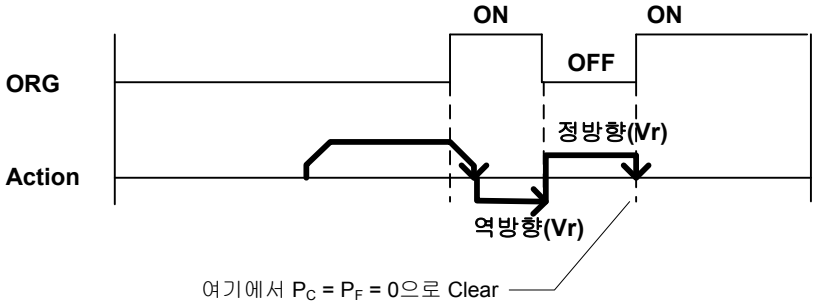
Value	Meaning
음수	수행 실패. 자세한 내용은 '에러처리' 편을 참고합니다
cmERR_NONE	수행 성공

SEE ALSO

□ PosClrMode 가 0 또는 2 로 하면 최종 단계에서 감속 없이 정지(停止)하는 원점복귀 모드(1, 2, 4, 6, 7)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.

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.

EXAMPLE

C/C++

```
#include “Cmmsdk.h”
#include “CmmsdkDef.h”

void OnSetHomePosClrMode ()
{
	long nPosClrMode = 0;		// 원점 복귀 위치 소거 모드 정보.

	/* 원점 복귀 완료 후 위치 소거 모드 */
	// 0 (cmHPCM_M0) : 원점 복귀 완료 HW 신호 ON 시 위치 소거.
	// 1 (cmHPCM_M1) : HW 신호를 포함한 전체 원점 복귀 작업 완료 후 위치 소거.
	// 2 (cmHPCM_M2) : 0 번 모드로 동작 후 원점 복귀 완료 후 Command 위치를 Feedback 위치로 설정.

	// 설정된 위치 소거 모드를 확인 후, 위치 소거 모드 2 번 모드로 설정합니다.
	if (cmmHomeGetPosClrMode (cmX1, &nPosClrMode ) == cmERR_NONE )
	{
		if ( nPosClrMode != cmHPCM_M2 )
		{
			cmmHomeSetPosClrMode (cmX1, cmHPCM_M2 );
		}
	}
}
```

Visual Basic

```
Private Sub OnSetHomePosClrMode ()

	Dim nPosClrMode As Long		‘ 원점 복귀 위치 소거 모드 정보

	‘ 설정된 위치 소거 모드를 확인 후, 위치 소거 모드 2 번 모드로 설정합니다.
	If cmmHomeGetPosClrMode (cmX1, nPosClrMode ) = cmERR_NONE Then

		If  nPosClrMode <> cmHPCM_M2 Then
			Call cmmHomeSetPosClrMode (cmX1, cmHPCM_M2 )
		End If
	End If

End Sub
```

Delphi

```
procedure OnSetHomePosClrMode ();
var
	nPosClrMode : LongInt;			// 원점 복귀 위치 소거 모드 정보

begin
	// 설정된 위치 소거 모드를 확인 후, 위치 소거 모드 2 번 모드로 설정합니다.
	if cmmHomeGetPosClrMode (cmX1, @nPosClrMode ) = cmERR_NONE then
```

```
        begin
                if  nPosClrMode <> cmHPCM_M2 then
                begin
                        cmmHomeSetPosClrMode (cmX1, cmHPCM_M2 );
                end;
        end;

end;
```

NAME

cmmHomeSetSpeedPattern
cmmHomeGetSpeedPattern
– 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 cmmHomeSetSpeedPattern

([in] VT_I4 Axis, [in] VT_I4 SpeedMode, [in] VT_R8 Vel, [in] VT_R8 Accel, [in] VT_R8 Decel,

[in] VT_R8 RevVel)

❒ VT_I4 cmmHomeGetSpeedPattern

([in] VT_I4 Axis, [out] VT_PI4 SpeedMode, [out] VT_PR8 Vel, [out] VT_PR8 Accel,

[out] VT_PR8 Decel, [out] VT_PR8 RevVel)

DESCRIPTION

cmmHomeSetSpeedPattern() 함수는 설정된 축의 원점복귀 속도 모드와 가감속도 및 작업속도, 역방향속도 등을 설정합니다.
cmmHomeGetSpeedPattern() 함수는 설정된 축의 원점복귀 속도 모드와 가감속도 및 작업속도, 역방향속도 등을 읽어옵니다.

PARAMETER

▶ Axis: 축번호. 축번호는 상수값으로 [cmX1] 부터 0 번째 축을 기준 축으로 임의의 축을 설정할 수 있습니다.

▶ SpeedMode : cmmHomeSetSpeedPattern 함수의 인자이며, 원점복귀시의 S-Curve 형 또는 선형 가감속, 가감속이 없는 모드등을 선택할 수 있습니다.

▶ SpeedMode : cmmHomeGetSpeedPattern 함수의 인자이며, 원점복귀모드를 반환합니다.

▶ Vel : cmmHomeSetSpeedPattern 함수의 인자이며, 작업 속도를 의미합니다. 기호로는 Vwork 로 표기합니다.

▶ Vel : cmmHomeGetSpeedPattern 함수의 인자이며, 작업 속도를 반환합니다.

▶ Accel : cmmHomeSetSpeedPattern 함수의 인자이며, 홈복귀시의 가속도를 의미합니다.

▶ Accel : cmmHomeGetSpeedPattern 함수의 인자이며, 홈복귀시의 가속도를 반환합니다.

▶ Decel : cmmHomeSetSpeedPattern 함수의 인자이며, 홈복귀시의 감속도를 의미합니다.

▶ Decel : cmmHomeGetSpeedPattern 함수의 인자이며, 홈복귀시의 감속도를 반환합니다.

▶ Revel : cmmHomeSetSpeedPattern 함수의 인자이며, Reverse Speed 를 설정합니다. 복귀모드에 따라 Reverse Speed 를 필요로 하는 모드가 있습니다. 앞의 복귀 모드 설명에서 Reverse Speed 는 Vr 로 표기되었습니다.

▶ Revel : cmmHomeGetSpeedPattern 함수의 인자이며, Reverse Speed 를 반환합니다.

RETURN VALUE

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

EXAMPLE

C/C++

```
#include "Cmmsdk.h"
#include "CmmsdkDef.h"

void OnSetHomeSpeedPattern ()
{
	long nHSpdMode;				// 원점 복귀 속도 환경 설정 정보.
	double fHVel, fHAcc, fHDec, fHRevVal;

	/* 원점 복귀 속도 패턴을 S-Curve 로 설정하고,
	작업 속도를 1000, 가속도를 10000, 감속도를 10000, Vr 을 10 으로 설정합니다.*/
	 cmmHomeSetSpeedPattern (	cmX1,		// 대상 축 선택.
					cmSMODE_S,	// 원점 복귀 속도 모드 선택.
					1000,		// 원점 복귀 작업 속도
					10000,		// 원점 복귀 가속도
					10000,		// 원점 복귀 감속도
					10		// Reverse Speed
					);

	// 설정되어있는 속도 패턴 및 작업속도, 가속도, 감속도를 반환합니다.
	cmmHomeGetSpeedPattern (cmX1, &nHSpdMode, &fHVel, &fHAcc, &fHDec, &fHRevVal );
}
```

Visual Basic

```
Private Sub OnSetHomeSpeedPattern ()

	Dim nHSpdMode As Long				' 원점 복귀 속도 환경 설정 정보.
	Dim fHVel As Double, fHAcc As Double, fHDec As Double, fHRevVal As Double

	' 0 번축의 속도 패턴을 S-Curve 로 설정하고,
	' 작업 속도를 2000, 가속도를 10000, 감속도를 10000, Vr 을 10 으로 설정합니다.
	 Call cmmHomeSetSpeedPattern (cmX1, cmSMODE_S, 2000, 10000, 10000 )

	' 설정되어있는 속도 패턴 및 작업속도, 가속도, 감속도를 반환합니다.
	Call cmmHomeGetSpeedPattern (cmX1, nHSpdMode, fHVel, fHAcc, fHDec, fHRevVal )

End Sub
```

Delphi

```
procedure OnSetHomeSpeedPattern ();
var
	nHSpdMode : LongInt;				// 원점 복귀 속도 환경 설정 정보.
	fHVel, fHAcc, fHDec,fHRevVal : Double;

begin
	{ 0 번축의 속도 패턴을 S-Curve 로 설정하고,
	 작업속도를 2000, 가속도를 10000, 감속도를 10000, Vr 을 10 으로 설정합니다. }
	 cmmHomeSetSpeedPattern (cmX1, cmSMODE_S, 2000, 10000, 10000, 10 );
```

```
    // 설정되어있는 속도 패턴 및 작업속도, 가속도, 감속도를 반환합니다.
    cmmHomeGetSpeedPattern (cmX1, @nHSpdMode, @fHVel, @fHAcc, @fHDec, @fRevVal );

end;
```

NAME

cmmHomeSetSpeedPattern_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 단위(PPS; Pulse per second)가, 가감속도 설정은 시간(밀리초, msec) 단위가 적용됩니다.

SYNOPSIS

❒VT_I4 cmmHomeSetSpeedPattern_T

([in] VT_I4 Axis, [in] VT_I4 SpeedMode, [in] VT_R8 Vel, [in] VT_R8 AccelTime,

[in] VT_R8 DecelTime, [in] VT_R8 RevVel)

DESCRIPTION

cmmHomeSetSpeedPattern_T() 함수는 설정된 축의 원점복귀 속도 모드와 가감속 시간 및 작업속도, 역방향속도 등을 설정합니다.

PARAMETER

▶ Axis: 축번호. 축번호는 상수값으로 [cmX1] 부터 0 번째 축을 기준 축으로 임의의 축을 설정할 수 있습니다.

▶ SpeedMode : 원점복귀시의 S-Curve 형 또는 선형 가감속, 가감속이 없는 모드등을 선택할 수 있습니다.

▶ Vel : 작업 속도를 의미합니다. 기호로는 Vwork 로 표기합니다.

▶ AccelTime : 원점복귀시의 가속 시간을 의미합니다.

▶ DecelTime : 원점귀시의 감속 시간을 의미합니다.

▶ Revel : 역방향 속도(Reverse Speed) 를 설정합니다. 복귀모드에 따라 Reverse Speed 를 필요로 하는 모드가 있습니다. 앞의 복귀 모드 설명에서 Reverse Speed 는 Vr 로 표기되었습니다.

SEE ALSO

cmmHomeSetSpeedPattern
cmmHomeGetSpeedPattern_T

RETURN VALUE

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

REFERENCE

□ 본 함수는 원점 복귀 시 가속 및 감속을 논리적 속도 단위(PPS) 대신 시간(msec) 으로 설정해야 하는 경우 유용하게 사용하실 수 있습니다.

EXAMPLE

C/C++

```
#include “Cmmsdk.h”
#include “CmmsdkDef.h”

void OnSetHomeSpeedPattern_T ()
{
	long nHSpdMode;				// 원점 복귀 속도 환경 설정 정보.
	double fHVel, fHAccTime, fHDecTime, fHRevVal;

	/* 원점 복귀 속도 패턴을 S-Curve 로 설정하고,
	작업 속도를 1000, 가속시간을 1 초(1000 ms), 감속시간을 1 초(1000 ms), Vr 을 10 으로 설정합니다.*/
	 cmmHomeSetSpeedPattern_T (	cmX1,		// 대상 축 선택.
					cmSMODE_S,	// 원점 복귀 속도 모드 선택.
					1000,		// 원점 복귀 작업 속도
					1000,		// 원점 복귀 가속 시간(msec)
					1000,		// 원점 복귀 감속 시간(msec)
					10		// Reverse Speed
					);

	// 설정되어있는 속도 패턴 및 작업속도, 가속시간, 감속시간을 반환합니다.
	cmmHomeGetSpeedPattern_T (cmX1, &nHSpdMode, &fHVel, &fHAccTime, &fHDecTime, &fHRevVal );
}
```

Visual Basic

```
Private Sub OnSetHomeSpeedPattern_T()

	Dim nHSpdMode As Long				‘ 원점 복귀 속도 환경 설정 정보.
	Dim fHVel As Double, fHAccTime As Double, fHDecTime As Double, fRevVal As Double

	‘ 0 번축의 속도 패턴을 S-Curve 로 설정하고,
	‘ 작업 속도를 2000, 가속시간을 1 초(1000 ms), 감속시간을 1 초(1000 ms), Vr 을 10 으로 설정합니다.
	 Call cmmHomeSetSpeedPattern_T (cmX1, cmSMODE_S, 2000, 1000, 1000, 10 )

	‘ 설정되어있는 속도 패턴 및 작업속도, 가속도, 감속도를 반환합니다.
	Call cmmHomeGetSpeedPattern_T (cmX1, nHSpdMode, fHVel, fHAccTime, fHDecTime, fHRevVal )

End Sub
```

Delphi

```
procedure OnSetHomeSpeedPattern_T ();
var
	nHSpdMode : LongInt;				// 원점 복귀 속도 환경 설정 정보.
	fHVel, fHAccTime, fHDecTime, fRevVal : Double;

begin
	{ 0 번축의 속도 패턴을 S-Curve 로 설정하고,
	 작업속도를 2000, 가속 시간을 1 초(1000 ms), 감속 시간을 1 초(1000 ms), Vr 을 10 으로 설정합니다. }
	 cmmHomeSetSpeedPattern_T (cmX1, cmSMODE_S, 2000, 1000, 1000, 10 );

	// 설정되어있는 속도 패턴 및 작업속도, 가속 시간, 감속 시간을 반환합니다.
	cmmHomeGetSpeedPattern_T (cmX1, @nHSpdMode, @fHVel, @fHAccTime, @fHDecTime, @fRevVal );

end;
```

NAME

cmmHomeGetSpeedPattern_T
– 원점 복귀속도설정 (原點復歸速度設定) 값 반환

INFORMATION

Home Return

VC++/VB

BCB/Delphi/.NET

Level 4

☺ 다소 주의

속도설정은 논리적인 속도 단위(PPS; Pulse per second)가, 가감속도 설정은 시간(밀리초, msec) 단위가 적용됩니다.

SYNOPSIS

❒VT_I4 cmmHomeGetSpeedPattern_T

([in] VT_I4 Axis, [out] VT_PI4 SpeedMode, [out] VT_PR8 Vel, [out] VT_PR8 AccelTime,

[out] VT_PR8 DecelTime, [out] VT_PR8 RevVel)

DESCRIPTION

cmmHomeGetSpeedPattern_T() 함수는 설정된 축의 원점복귀 속도 모드와 가감속 시간 및 작업속도, 역방향속도 등을 읽어옵니다.

PARAMETER

▶ Axis: 축번호. 축번호는 상수값으로 [cmX1] 부터 0 번째 축을 기준 축으로 임의의 축을 설정할 수 있습니다.

▶ SpeedMode : 원점복귀모드를 반환합니다.

▶ Vel : 작업 속도를 반환합니다.

▶ AccelTime : 원점복귀시의 설정된 가속 시간을 반환합니다.

▶ DecelTime : 원점귀시의 설정된 감속 시간을 반환합니다.

▶ Revel : 역방향 속도(Reverse Speed) 를 반환합니다.

SEE ALSO

cmmHomeGetSpeedPattern
cmmHomeSetSpeedPattern_T

RETURN VALUE

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

REFERENCE

□ 본 함수는 원점 복귀 시 가속 및 감속을 cmmHomeSetSpeedPattern_T 함수를 사용해서 논리적 속도 단위(PPS) 대신 시간(msec) 으로 설정 했을 경우 설정된 가속 및 감속 시간을 확인 해야 하는 경우 유용하게 사용하실 수 있습니다.

EXAMPLE

C/C++

```
#include “Cmmsdk.h”
#include “CmmsdkDef.h”

void OnSetHomeSpeedPattern_T ()
{
	long nHSpdMode;				// 원점 복귀 속도 환경 설정 정보.
	double fHVel, fHAccTime, fHDecTime, fHRevVal;

	/* 원점 복귀 속도 패턴을 S-Curve 로 설정하고,
	작업 속도를 1000, 가속시간을 1 초(1000 ms), 감속시간을 1 초(1000 ms), Vr 을 10 으로 설정합니다.*/
	 cmmHomeSetSpeedPattern_T (	cmX1,		// 대상 축 선택.
					cmSMODE_S,	// 원점 복귀 속도 모드 선택.
					1000,		// 원점 복귀 작업 속도
					1000,		// 원점 복귀 가속 시간(msec)
					1000,		// 원점 복귀 감속 시간(msec)
					10		// Reverse Speed
					);

	// 설정되어있는 속도 패턴 및 작업속도, 가속시간, 감속시간을 반환합니다.
	cmmHomeGetSpeedPattern_T (cmX1, &nHSpdMode, &fHVel, &fHAccTime, &fHDecTime, &fHRevVal );
}
```

Visual Basic

```
Private Sub OnSetHomeSpeedPattern_T()

	Dim nHSpdMode As Long				‘ 원점 복귀 속도 환경 설정 정보.
	Dim fHVel As Double, fHAccTime As Double, fHDecTime As Double, fRevVal As Double

	‘ 0 번축의 속도 패턴을 S-Curve 로 설정하고,
	‘ 작업 속도를 2000, 가속시간을 1 초(1000 ms), 감속시간을 1 초(1000 ms), Vr 을 10 으로 설정합니다.
	 Call cmmHomeSetSpeedPattern_T (cmX1, cmSMODE_S, 2000, 1000, 1000, 10 )

	‘ 설정되어있는 속도 패턴 및 작업속도, 가속도, 감속도를 반환합니다.
	Call cmmHomeGetSpeedPattern_T (cmX1, nHSpdMode, fHVel, fHAccTime, fHDecTime, fRevVal )

End Sub
```

Delphi

```
procedure OnSetHomeSpeedPattern_T ();
var
	nHSpdMode : LongInt;				// 원점 복귀 속도 환경 설정 정보.
	fHVel, fHAccTime, fHDecTime,fRevVal : Double;

begin
	{ 0 번축의 속도 패턴을 S-Curve 로 설정하고,
	 작업속도를 2000, 가속 시간을 1 초(1000 ms), 감속 시간을 1 초(1000 ms), Vr 을 10 으로 설정합니다. }
	 cmmHomeSetSpeedPattern_T (cmX1, cmSMODE_S, 2000, 1000, 1000, 10 );

	// 설정되어있는 속도 패턴 및 작업속도, 가속 시간, 감속 시간을 반환합니다.
	cmmHomeGetSpeedPattern_T (cmX1, @nHSpdMode, @fHVel, @fHAccTime, @fHDecTime, @fRevVal );

end;
```

NAME

cmmHomeMove
cmmHomeMoveStart
- 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 cmmHomeMove

([in] VT_I4 Axis, [in] VT_I4 Direction, [in] VT_I4 IsBlocking)

❒ VT_I4 cmmHomeMoveStart

([in] VT_I4 Axis, [in] VT_I4 Direction)

DESCRIPTION

원점복귀 작업을 수행합니다. cmmHomeMove() 함수는 모션이 완료되기 전까지 반환되지 않으며, cmmHomeMoveStart() 함수는 모션을 시작시킨 후에 바로 반환됩니다.

PARAMETER

▶ Axis: 축번호. 축번호는 상수값으로 [cmX1] 부터 0 번째 축을 기준 축으로 임의의 축을 설정할 수 있습니다.

▶ Direction : 원점 복귀 모션을 수행할 방향을 지정합니다.

Value	Meaning
0 또는 cmDIR_N	(-) 방향
1 또는 cmDIR_P	(+) 방향

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

Value	Meaning
cmFALSE	블록(Blocking)을 하지 않습니다. 따라서 해당 모션이 완료되는 동안에도 윈도우 이벤트를 처리합니다.
cmTRUE	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.
cmERR_NONE	수행 성공

SEE ALSO

□ cmmHomeMoveStart() 함수를 사용하는 경우에는 cmmSxIsDone(), cmmSxWaitDone() 또는 cmmMxIsDone(), cmmMxWaitDone() 함수를 사용하여 모션의 완료를 확인(確認)할 수 있습니다. 그러나 가장 바람직한 방법은 cmmHomeWaitDone() 함수를 사용하는 것이 좋습니다.

□ cmmHomeMove() 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 cmmHomeMoveStart() 함수를 이용하여 X1, Y1 축의 원점복귀를 수행하는 함수입니다. 원점복귀에 대한 환경설정은 이미 이루어진 것으로 가정합니다.

C/C++

```
#include "Cmmsdk.h"
#include "CmmsdkDef.h"

/************************************************************
* OnProgramInitial : 이 함수는 가상의 함수로서 프로그램 초기화 루틴이
* 적용되는 부분을 의미합니다.
************************************************************/
void OnProgramInitial()
{

	long m_nNumAxes;

	cmmLoadDll();
	if(cmmGnDeviceLoad(cmTRUE, &m_nNumAxes) != cmERR_NONE)
	{
		//Handle 은 사용자가 생성한 폼의 핸들 값입니다.
		cmmErrShowLast(Handle);
		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.
	cmmHomeSetConfig(cmX1, 0, 1, 100, 10);
	//Y1 축의 홈복귀 모드를 설정합니다.
	cmmHomeSetConfig(cmY1, 0, 1, 100, 10);
	// X1 축 홈복귀 속도패턴 설정 //
	cmmHomeSetSpeedPattern(cmX1, cmSMODE_S, m_fVwork, m_fAcc, m_fDec, m_fRevVel );
	// Y1 축 홈복귀 속도패턴 설정 //
	cmmHomeSetSpeedPattern(cmY1, cmSMODE_S, m_fVwork, m_fAcc, m_fDec, m_fRevVel );
}

/*************************************************************
* OnHomeReturn : 이 함수는 가상의 함수로서  원점복귀를 실행합니다.
*************************************************************/
void OnHomeRetrun()
{
	// X1 축 원점복귀 시작 //
	if(cmmHomeMoveStart(cmX1, cmDIR_N) != cmERR_NONE){
		cmmErrShowLast(Handle);
		return;
	}
	// Y1 축 원점복귀 시작 //
	if(cmmHomeMoveStart(cmY1, cmDIR_N) != cmERR_NONE){
		cmmErrShowLast(Handle);
		return;
	}
	// X1&Y1 두축의 원점복귀 작업이 완료될 때까지 기다린다. //
	if(cmmHomeWaitDone(cmX1, cmFALSE) != cmERR_NONE)  cmmErrShowLast(Handle);
	if(cmmHomeWaitDone(cmY1, cmFALSE) != cmERR_NONE)  cmmErrShowLast(Handle);
	////////////////////////////////////////////////////////
	//위의 cmmSxWaitDone() 함수 대신에 아래와 같이 cmmMxWaitDone()
	//함수를 사용하여 두축의 작업완료를 기다리는 것을 한번에 수행할 수 있다.
	// int nAxes[2] = {cmX1, cmY1};
	// cmmMxWaitDone(2, nAxes, cmFALSE);

	// 원점복귀의 성공 여부를 확인(確認)하여 처리한다. //
	long dwIsSuccess;
	cmmHomeGetSuccess(nAxis, &dwIsSuccess);

	if(dwIsSuccess){
		MessageBox("원점복귀를 성공적으로 수행하였습니다.", "Message", MB_OK);
}else{

	long dwErrCode;
	char szErrMsg[CMM_MAX_STR_LEN_ERR];
	char szErrReason[CMM_MAX_STR_LEN_ERR];

	cmmErrGetLastCode(&dwErrCode);
	cmmErrGetString(dwErrCode, szErrReason, CMM_MAX_STR_LEN_ERR);

	sprintf(szErrMsg, "다음과 같은 이유로 원점복귀에 실패하였습니다.\n%s", szErrReason);
	MessageBox(szErrMsg, "Motion Error", MB_OK | MB_ICONERROR);

	}

}
```

```
Visual Basic

'============================================================
'cmmGnDeviceLoad 함수로 장치를 초기화 합니다.
'============================================================
Private Sub Form_Load()
```

```
    Dim nTotalAxis As Long
    Dim IRetVal As Long

'===============================================================
 ' cmmGnDeviceLoad 함수로 장치를 초기화합니다.
'===============================================================
  IRetVal = cmmGnDeviceLoad(True, nTotalAxis)

  If IRetVal <> cmERR_NONE Then
     MsgBox ("cmmGnDeviceLoad has been failed")
  End If
End Sub

' 버튼 이벤트에 의해서 홈복귀를 시작합니다.
Private Sub btnHome_Click()
    ' cmmHomeSetConfig( 대상 축, 홈 복귀 모드, EZCount, EscDist, Offset )
    Call cmmHomeSetConfig(cmX1, 0, 1, 1000, 0)

    ' cmmHomeMove(대상 축, 홈 복귀 방향, 블럭 여부)
    Call cmmHomeMove(cmX1, GetDirection, cmDIR_N, cmFALSE )
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 = cmmSxStop(cmX1, IsWaitComplete, cmFALSE)
End Sub

Private Sub CfgSpeed(nTotalAxis As Long)

        '===============================================================
        ' 이 함수에서 cmmCfgSetSpeedPattern 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 cmmCfgSetSpeedPattern(cmX1, cmSMODE_S, 1000, 2000, 2000)

End Sub
```

```
Delphi

// *********************************************************************
// 원점 복귀 환경을 설정합니다.
// *********************************************************************
procedure OnSetHomeConfig ();
begin
        { 원점 복귀 환경을 원점 복귀 모드 = 0, 방향 = (-), Ez Count = 0,
         원점 탈출 거리 = 10, Offset = 0 으로 설정합니다. }
        cmmHomeSetConfig (cmX1, 0, 0, 10, 0 );
        cmmHomeSetConfig (cmY1, 0, 0, 10, 0 );

        // 원점 복귀 속도 환경을 설정합니다.
        cmmHomeSetSpeedPattern (cmX1, cmSMODE_S, 1000, 10000, 10000, 10 );
        cmmHomeSetSpeedPattern (cmY1, cmSMODE_S, 1000, 10000, 10000, 10 );

end;


// *********************************************************************
// 원점 복귀 이송을 수행합니다.
```

```
// ********************************************************************
procedure OnHomeReturn ();
var
	nIsHomming : LongInt				// 원점 복귀 진행 상태 정보

begin
	// cmmHomeIsBusy() 명령을 통한 원점 복귀 완료 체크
	if cmmHomeMoveStart (cmX1, cmDIR_N ) = cmERR_NONE then
	begin
		nIsHomming := cmTRUE;
		while nIsHomming = cmTRUE do
	begin
		{ 0 (cmFALSE) : 원점 복귀 이송 작업이 진행중이지 않습니다.
		 1 (cmTRUE) : 원점 복귀 이송 작업이 진행중입니다. }
		cmmHomeIsBusy (cmX1, @nIsHomming );
	end;
end;

	// cmmHomeWaitDone() 명령을 통한 원점 복귀 완료 체크.
	if cmmHomeMoveStart (cmY1, cmDIR_N ) = cmERR_NONE then
	begin
		cmmHomeWaitDone (cmY1, cmFALSE );    // 원점 복귀 완료 시까지 대기합니다.
	end

	// cmmHomeMoveStart(), cmmHomeWaitDone() 을 아래 코드로 대체 할 수 있습니다.
	// cmmHomeMove(cmY1, cmFALSE );

end;


// ********************************************************************
// 원점 복귀 성공 여부를 확인합니다.
// ********************************************************************
procedure OnGetHomeSuccess ();
var
	nIsSuccess : LongInt			// 원점 복귀 성공 여부 정보.

begin
	// 원점 복귀 성공 여부를 확인합니다.
	cmmHomeGetSuccess (cmX1, @nIsSuccess );

	// 원점 복귀 성공 메시지를 표시합니다.
	if nIsSuccess = cmTRUE then
	begin
		ShowMessage ( 'Home return success' );
	end;

end;
```

NAME

cmmHomeMoveAll
cmmHomeMoveAllStart
- 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 cmmMxWaitDone 함수를 사용할 수 있습니다.

SYNOPSIS

❒ VT_I4 cmmHomeMoveAll

([in] VT_I4 NumAxes, [in] VT_PI4 ChannelList, [in] VT_PI4 DirList, [in] VT_I4 IsBlocking)

❒ VT_I4 cmmHomeMoveAllStart

([in] VT_I4 NumAxes, [in] VT_PI4 ChannelList, [in] VT_PI4 DirList)

DESCRIPTION

여러 축에 대한 원점복귀 작업을 동시에 수행합니다. cmmHomeMoveAll() 함수는 모션이 완료되기 전까지 반환되지 않으며, cmmHomeMoveAllStart() 함수는 모션을 시작시킨 후에 바로 반환됩니다.

PARAMETER

▶ NumAxes : 동시에 작업을 수행할 대상 축의 수

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

▶ DirList : 방향을 지시하는 값의 배열 주소값. 이 배열의 크기는 NumAes 값과 일치해야 합니다. 모션의 방향을 지시하는 값은 다음과 같습니다.

Value	Meaning
0 또는 cmDIR_N	(-) 방향
1 또는 cmDIR_P	(+) 방향

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

Value	Meaning
cmFALSE	블록(Blocking)을 하지 않습니다. 따라서 해당 모션이 완료되는 동안에도 윈도우 이벤트를 처리합니다.
cmTRUE	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
cmERR_NONE	수행 성공

SEE ALSO

☐ cmmHomeMoveAllStart() 함수를 사용하는 경우에는 cmmMxIsDone() 함수나 MxWaitDone() 함수를 사용하여 모션의 완료를 확인(確認)할 수 있습니다.
☐ cmmHomeMoveAll() 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 cmmHomeMoveAll() 함수를 이용하여 X1, Y1 축의 원점복귀를 동시에 수행하는 함수입니다.

C/C++

```
#include "Cmmsdk.h"
#include "CmmsdkDef.h"

/*************************************************************
* 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.
	cmmHomeSetConfig(cmX1, 0, 1, 100, 10);
	//Y1 축의 홈복귀 모드를 설정합니다.
	cmmHomeSetConfig(cmY1, 0, 1, 100, 10);
	// X1 축 홈복귀 속도패턴 설정 //
	cmmHomeSetSpeedPattern(cmX1, cmSMODE_S, m_fVwork, m_fAcc, m_fDec, m_fRevVel );
```

```
	// Y1 축 홈복귀 속도패턴 설정 //
	cmmHomeSetSpeedPattern(cmY1, cmSMODE_S, m_fVwork, m_fAcc, m_fDec, m_fRevVel );
}

/*************************************************************
* OnHomeReturn : 이 함수는 가상의 함수로서  원점복귀를 실행합니다.
*************************************************************/
void OnHomeReturn()
{
	long nAxes[2] = {cmX1, cmY1};
	long nDirList[2] = {cmDIR_N, cmDIR_N};

	if(cmmHomeMoveAll(2, nAxes, nDirList, cmFALSE) != cmERR_NONE){
		cmmErrShowLast(Handle);
		return ;
	}
	////////////////////////////////////////////////////////
	// cmmHomeMoveAll() 함수 대신 아래와 같이 cmmHomeMoveAllStart() 함
	// 수를 사용할 수 있습니다.
	// cmmHomeMoveAllStart(2, nAxes, nDirList, cmFALSE);
	// cmmMxWaitDone(2, nAxes, cmFALSE);
}
```

Visual Basic

```
' 홈복귀를 위한 가상 함수를 시작합니다.
Private Sub OnStart()

   Dim AxisList(2) As Long
   Dim DirList(2) As Long

   AxisList(0) = cmX1
   AxisList(1) = cmY1

   DirList(0) = cmDIR_N
   DirList(1) = cmDIR_N


   ' cmmHomeSetConfig( 대상 축, 홈 복귀 모드, EZCount, EscDist, Offset )
   Call cmmHomeSetConfig(AxisList(0), 0, 0, 1000, 0)
   Call cmmHomeSetConfig(AxisList(1), 0, 0, 1000, 0)

   ' cmmHomeSetSpeedPattern 함수를 통해 원점 복귀 속도를 결정합니다.
   Call cmmHomeSetSpeedPattern(AxisList(0), cmSMODE_S, 10000, 20000, 20000, 1000)
   Call cmmHomeSetSpeedPattern(AxisList(1), cmSMODE_S, 10000, 20000, 20000, 1000)

   ' cmmHomeMoveAll(대상 축, 홈 복귀 방향, 블록킹 여부)
   Call cmmHomeMoveAll(2,AxisList(0),DirList(0), cmFALSE)

End Sub
```

Delphi

```
procedure btnHomeMoveClick();
var
   arAxes : Array[0..1] of LongInt;
   arDirecton : Array[0..1] of LongInt;
begin

   // cmmHomeSetConfig( 대상 축, 홈 복귀 모드, EZCount, EscDist, Offset );

   // X1 축 에 대한 홈 설정을 합니다.
   cmmHomeSetConfig(cmX1, 0, 0, 1000, 0 );

   // Y1 축에 대한 홈 설정을 합니다.
   cmmHomeSetConfig(cmY1, 0, 0, 1000, 0 );

   // X1 축의 홈 복귀 속도를 설정합니다.
```

```
    cmmHomeSetSpeedPattern(
        cmX1,
        cmSMODE_S,
        10000,
        20000,
        20000,
        1000);

    // Y1 축의 홈 복귀 속도를 설정합니다.
    cmmHomeSetSpeedPattern(
        cmY1,
        cmSMODE_S,
        10000,
        20000,
        20000,
        1000);

    //  다축의 홈 이송을 시작합니다. 각 축의 홈 복귀 방향은 Negative 로 설정합니다
    arAxes[0] := cmX1;              // X1 축
    arAxes[1] := cmY1;              // Y1 축
    arDirecton[0] := cmDIR_N;
    arDirecton[1] := cmDIR_N;

    // 인자는 다음과 같습니다.
    // cmmHomeMoveAllStart( 홈복귀 대상축, 축의 배열, 방향의 배열)
    // 이 명령은 홈 복귀 명령 실행시 바로 리턴됩니다.
    cmmHomeMoveAllStart(2, @arAxes, @arDirecton);

    // 두 축에 대해서 홈 복귀 완료시까지 대기합니다.
    cmmHomeWaitDone(cmX1, cmFALSE);
    cmmHomeWaitDone(cmY1, cmFALSE);
end;
```

NAME

cmmHomeIsBusy
– 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 cmmHomeIsBusy ([in] VT_I4 Axis, [out] VT_PI4 IsBusy)

DESCRIPTION

지정한 축이 현재 원점복귀를 진행중인지를 IsBusy 버퍼를 통하여 반환합니다.

PARAMETER

▶ Axis: 축번호. 축번호는 상수값으로 [cmX1] 부터 0 번째 축을 기준 축으로 임의의 축을 설정할 수 있습니다.

▶ IsBusy : 현재 원점복귀가 진행중인지를 알려주는 값을 반환받을 버퍼. 이 값에 반환되는 값의 의미는 다음과 같습니다.

Value	Meaning
0 (FALSE)	지정한 축은 현재 원점복귀가 진행중이지 않습니다.
1 (TRUE)	지정한 축은 현재 원점복귀를 진행하고 있습니다.

RETURN VALUE

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

SEE ALSO

□ CMMSDK 라이브러리에서는 cmmSxIsDone()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 cmmHomeGetSuccess() 함수와 혼동될 소지가 있어서 cmmHomeIsDone() 함수 대신에 cmmHomeIsBusy() 함수를 제공하게 되었습니다.

□ cmmHomeIsBusy() 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 cmmHomeIsBusy()함수를 이용하여 원점복귀의 원점복귀가 완료되었음을 확인(確認)한 후에는 cmmHomeGetSuccess() 함수를 사용하여 원점복귀의 성공여부를 확인(確認)하여 각각의 상황에 대한 처리를 해주는 것이 바람직합니다.

EXAMPLE

C/C++

```
BOOL OnHomeMove(int nAxis)
{
        long dwIsHomming = TRUE;
        cmmHomeMoveStart(nAxis, cmDIR_N);
```

```
	while(dwIsHomming){

		cmmHomeIsBusy(nAxis, &dwIsHomming);

		// 원점복귀 진행여부 읽기
		// 윈도우 메시지를 처리해준다 (단, 쓰레드를 사용하는 경우에는
		// 아래 함수는 생략되어야 한다)
		cmmUtlProcessWndMsgM(GetSafeHwnd(), 500, NULL);
	}

	long dwIsSuccess;

	if(cmmHomeGetSuccess(nAxis, &dwIsSuccess) != cmERR_NONE){
		cmmErrShowLast (GetSafeHwnd());
		return FALSE;
	}

	if(dwIsSuccess){
		MessageBox("원점복귀를 성공적으로 수행하였습니다.", "Message", MB_OK);
	}else{
		char szErrMsg[CMM_MAX_STR_LEN_ERR];
		char szErrReason[CMM_MAX_STR_LEN_ERR];

		long dwErrCode;
		cmmErrGetLastCode(&dwErrCode);
		cmmErrGetString(dwErrCode, szErrReason, CMM_MAX_STR_LEN_ERR);

		sprintf(szErrMsg, "다음과 같은 이유로 원점복귀에 실패하였습니다.\n%s", szErrReason);
		MessageBox(szErrMsg, "Motion Error", MB_OK | MB_ICONERROR);
		 return FALSE;
}

return TRUE;
}
```

```
Visual Basic

Dim dwIsHomming As Long
Dim dwIsSuccess As Long

	dwIsHomming = True

	Call cmmHomeMoveStart(cmX1, cmDIR_N)

	Do While (dwIsHomming)
		Call cmmHomeIsBusy(cmX1, dwIsHomming) '원점 진행여부 확인(確認)
	loop

	If (cmmHomeGetSuccess(cmX1, dwIsSuccess) <> cmERR_NONE) Then
		Call cmmErrShowLast(HomeRtn.Hwnd)
	End If

	If (dwIsSuccess) Then
		MsgBox ("원점 복귀를 성공적으로 수행하였습니다.")
	End If
```

```
Delphi

//* cmmHomeMove / cmmHomeMoveStart 예제를 참고하여 주시기 바랍니다.
```

NAME

cmmHomeWaitDone
- 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 cmmHomeWaitDone ([in] VT_I4 Axis, [in] VT_I4 IsBlocking)

DESCRIPTION

cmmHomeWaitDone() 함수는 해당 축에 대해 원점 복귀가 완료(完了)될때까지 기다립니다. 이 함수는 반복문(loop)에서 cmmHomeIsBusy() 함수를 계속 호출하다가 원점복귀가 완료(完了)되면 반복문(loop) 루프를 탈출하는 용도로 사용됩니다.

cmmHomeIsBusy() 함수를 통해 원점 복귀가 완료된 것을 확인할 수 있으며, 내부적으로 반복문을 통해 원점 복귀 완료를 확인하는 함수가 cmmHomeWaitDone() 입니다. 용도에 따라서 사용하시기 바랍니다.

PARAMETER

▶ Axis: 축번호. 축번호는 상수값으로 [cmX1] 부터 0 번째 축을 기준 축으로 임의의 축을 설정할 수 있습니다.

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

Value	Meaning
cmFALSE	블록(Blocking)을 하지 않습니다. 따라서 해당 모션이 완료되는 동안에도 윈도우 이벤트를 처리합니다.
cmTRUE	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
cmERR_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.

EXAMPLE

```
//* cmmHomeMove / cmmHomeMoveStart 예제를 참고하여 주시기 바랍니다.
```

NAME

cmmHomeGetSuccess
cmmHomeSetSuccess
– 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 cmmHomeGetSuccess ([in] VT_I4 Channel, [out] VT_PI4 IsSuccess)

VT_I4 cmmHomeSetSuccess ([in] VT_I4 Channel, [in] VT_I4 IsSuccess)

DESCRIPTION

cmmHomeGetSuccess() 함수는 이 함수가 호출되기 이전에 원점복귀가 성공적으로 완료되었는지를 알려주는 함수입니다.
cmmHomeSetSuccess() 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.

PARAMETER

▶ Channel : 축(채널) 번호. 축번호는 상수값으로 [cmX1] 부터 0 번째 채널을 기준 채널로 임의의 채널을 설정할 수 있습니다.

▶ IsSuccess : cmmHomeGetSuccess 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 : cmmHomeSetSuccess 함수의 인자이며, 원점복귀의 성공여부에 대한 플래그 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
cmERR_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 재 시작(Rebooting) 되면 그 값은 FALSE 로 리셋됩니다. 따라서

cmmHomeGetSuccess() 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 cmmHomeMoveStart() 함수를 사용한 경우에는 먼저 cmmHomeIsBusy() 함수나 cmmHomeWaitDone() 함수를 선행하여 완료를 확인(確認)한 후에 cmmHomeGetSuccess()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 cmmHomeGetSuccess() 함수를 이용하여 이전에 원점복귀가 이미 성공적으로 수행되었는지를 확인(確認)하고, 만일 그러한 경우라면 원점복귀를 생략하도록 합니다.

C/C++ :

```
BOOL OnProgramInitialHome(int nAxis)
{

	long dwAlreadyDone;
	cmmHomeGetSuccess(nAxis, &dwAlreadyDone); // 이전에 원점복귀 상태 확인(確認)
	if(dwAlreadyDone){// 이전에 이미 원점복귀가 이루어졌으면 원점복귀 생략
	return TRUE;

}

long dwIsHomming = TRUE;
cmmHomeMoveStart(nAxis, cmDIR_N);
while(dwIsHomming){

	cmmHomeIsBusy(nAxis, &dwIsHomming); // 원점복귀 진행여부 읽기

	// 윈도우 메시지를 처리해준다 (단, 쓰레드를 사용하는 경우에는
	// 아래 함수는 생략되어야 한다)
	cmmUtlProcessWndMsgM(GetSafeHwnd(), 500, NULL);
}

// 원점복귀의 성공여부를 확인(確認)하여 처리한다. //
long dwIsSuccess;
cmmHomeGetSuccess(nAxis, &dwIsSuccess);
if(dwIsSuccess){

	MessageBox("원점복귀를 성공적으로 수행하였습니다.", "Message", MB_OK);

}else{

	char szErrMsg[CMM_MAX_STR_LEN_ERR];
	char szErrReason[CMM_MAX_STR_LEN_ERR];
	long dwErrCode;
	cmmErrGetLastCode(&dwErrCode);
	cmmErrGetString(dwErrCode, szErrReason, CMM_MAX_STR_LEN_ERR);
	sprintf(szErrMsg, "다음과 같은 이유로 원점복귀에 실패하였습니다.\n%s", szErrReason);
		MessageBox(szErrMsg, "Motion Error", MB_OK | MB_ICONERROR);

    }
}
```

Visual Basic

```
Dim dwIsHomming As Long
Dim dwIsSuccess As Long

dwIsHomming = True

Call cmmHomeMoveStart(cmX1, cmDIR_N)
```

```
Do While (dwIsHomming)

	'원점 진행여부 확인(確認)
	Call cmmHomeIsBusy(cmX1, dwIsHomming)
Loop

If (cmmHomeGetSuccess(cmX1, dwIsSuccess) <> cmERR_NONE)
Then
	Call cmmErrShowLast(handle)

If (dwIsSuccess) Then
	MsgBox ("원점 복귀를 성공적으로 수행하였습니다.")
End If
```

```
Delphi

//* cmmHomeMove / cmmHomeMoveStart 예제를 참고하여 주시기 바랍니다.
```

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 모션제어에서는 1 나노미터의 오차도 허용하지 않는 저희㈜커미조아의 모션제품이 고객(顧客) 여러분들을 만족시켜 드립니다.

이 단원에서는 속도(速度) 및 위치(位置) 오버라이딩 함수들을 소개합니다. 속도(速度) 오버라이딩은 모션이 오버라이딩은 모션이 진행되고 있는 중에 작업 속도를 변경하는 것을 의미합니다. 위치(位置) 오버라이딩은 위치(位置) 오버라이딩은 상대좌표 모션 이송이나 절대좌표 모션이송과 같이 목적좌표를 향해 추종 모션을 향해 추종 모션을 수행하고 있는 중에 최종 목표 거리 또는 최종 목표 좌표를 수정하는 것을 의미 합니다.
의미 합니다.

9 고급 모션 제어 편

9.1 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 Move나 MoveTo와 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.

9.1.1 함수 요약

㈜ 커미조의속도 및 위치 오버라이딩에 관련된 함수는 다음과 같습니다.

Summary of Functions
❒ VT_I4 cmmOverrideSpeedSet ([in] VT_I4 Axis) 단축(單軸) 모션 작업이 진행되고 있는 중에 속도(速度)를 변경합니다.
❒ VT_I4 cmmOverrideSpeedSetAll ([in] VT_I4 NumAxes, [in] VT_PI4 AxisList) 다축(多軸) 모션 작업이 진행되고 있는 대상 모션 채널들에 대하여, 동시에 속도(速度)를 변경합니다.
❒ VT_I4 cmmOverrideMove ([in] VT_I4 Axis, [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 cmmOverrideMoveTo ([in] VT_I4 Axis, [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 이송 중에 적용되어 지는 함수이기 때문에, 이전에 수행된 모션 명령이 cmmSxMove 나 cmmSxMoveTo 같은 결과적으로 모션이송의 목표 위치에 대한 완료를 동반하여 반환되는 함수에서는 사용하는 것이 배제됩니다. 따라서 오버라이딩 함수의 이전함수는 cmmSxMoveStart 나 cmmSxMoveToStart 와 같은 함수를 통해 이송 명령이 설정된 후에 응용프로그램의 제어를 즉시 반환 받고, 오버라이딩을 수행하게 됩니다.

9.1.2 함수 설명

NAME

cmmOverrideSpeedSet
– 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 cmmOverrideSpeedSet ([in] VT_I4 Axis)

DESCRIPTION

이 함수는 단축 모션이 진행되고 있는 중에 속도를 오버라이딩하고자할 때 사용하는 함수입니다. 속도를 오버라이딩하기 위해서는 먼저 cmmCfgSetSpeedPattern() 속도 패턴 설정 함수를 통하여 변경하고자 하는 속도 또는 가속도값을 설정하고나서 이 함수를 수행해야합니다.

PARAMETER

▶ Axis: 축번호. 축번호는 상수값으로 [cmX1] 부터 0 번째 축을 기준 축으로 임의의 축을 설정할 수 있습니다.

RETURN VALUE

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

SEE ALSO

□ 여러축을 동시에 속도 오버라이딩(Overriding)하고자 한다면 이 함수 대신에 cmmOverrideSpeedSetAll() 함수를 사용하십시오.

□ 직선, 원호, 헬리컬 보간작업을 수행하는 경우에는 속도 오버라이딩을 사용할 수 없습니다.

EXAMPLE

본 예제는 cmmOverrideSpeedSet()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++

```
#include "Cmmsdk.h"
#include "CmmsdkDef.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_nNumAxes;

	cmmLoadDll();
	if(cmmGnDeviceLoad(cmTRUE, &m_nNumAxes) != cmERR_NONE)
	{
		//Handle 은 사용자가 생성한 폼의 핸들 값입니다.
		cmmErrShowLast(Handle);
		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 //
	cmmSxIsDone(cmX1, &nIsDone);
	if(nIsDone != cmTRUE) cmmSxStopEmg(cmX1);
	// 속도설정 => 시작은 LOW 속도로 시작 //
	cmmCfgSetSpeedPattern(cmX1, cmSMODE_S, V_LOW, 50000, 50000);
	// V-Move start //
	if(cmmSxVMoveStart(cmX1, cmDIR_P)){
		cmmErrShowLast(Handle);
		return;
	}
}
/*************************************************************
* OnHighButtonClick() : "HIGH" 버튼 콜백함수 (가상함수)
* "HIGH"버튼이 클릭되면 속도를 V_HIGH 속도로 오버라이드한다.
*************************************************************/
void OnHighButtonClick()
{
	// V_HIGH 속도로 오버라이딩 //
	cmmSxSetSpeedRatio(cmX1, cmSMODE_S, , 200, 100, 100);
	//아래 코드로 대체가 가능합니다.
	//cmmCfgSetSpeedPattern(cmX1, cmSMODE_S, V_HIGH, 50000,  50000);

	if(cmmOverrideSpeedSet (cmX1) != cmERR_NONE){
		cmmErrShowLast(Handle);
		return;
	}
}
/*************************************************************
* OnLowButtonClick() : "LOW" 버튼 콜백함수 (가상함수)
* "LOW"버튼이 클릭되면 속도를 V_LOW 속도로 오버라이드한다.
*************************************************************/
void OnLowButtonClick()
{
	// V_LOW 속도로 오버라이딩 //
	cmmSxSetSpeedRatio(cmX1, cmSMODE_S, , 100, 100, 100);
	//아래 코드로 대체가 가능합니다.
	//cmmCfgSetSpeedPattern(cmX1, cmSMODE_S, V_LOW, 50000, 50000);

	if(cmmOverrideSpeedSet (cmX1) != cmERR_NONE){
		cmmErrShowLast(Handle);
		return;
	}
}
/*************************************************************
* OnStop() : "Stop"명령시에 호출되는 가상의 함수
*************************************************************/
void OnStop()
{
	cmmSxStopEmg(cmX1);
```

```
}
```

Visual Basic

```
'/*************************************************************
'* 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 cmmSxIsDone(cmX1, nIsDone)
    If (nIsDone <> cmTRUE) Then
        cmmSxStopEmg (cmX1)
    End If

    '시작은 LOW 속도로 시작
    Call cmmCfgSetSpeedPattern(cmX1, cmSMODE_S, 10000, 50000, 50000)

    If (cmmSxVMoveStart(cmX1, cmDIR_P)) Then
       Call cmmErrShowLast(thisform.Hwnd)
    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 cmmSxSetSpeedRatio(cmX1, cmSMODE_S, 200, 100, 100)

    '아래 코드로 대체가 가능합니다.
    'Call cmmCfgSetSpeedPattern(cmX1, cmSMODE_S, 20000, 50000, 50000)

    If (cmmOverrideSpeedSet(cmX1) <> cmERR_NONE) Then
       Call cmmErrShowLast(thisform.Hwnd)
    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 cmmCfgSetSpeedPattern(cmX1, cmSMODE_S, 10000, 50000, 50000)

    If (cmmOverrideSpeedSet(cmX1) <> cmERR_NONE) Then
       Call cmmErrShowLast(thisform.Hwnd)

End Sub

`/*************************************************************
`* OnStop() : “Stop”명령시에 호출되는 가상의 함수
`*************************************************************/
Private Sub OnStop()

    Call cmmSxStopEmg(cmX1)

End Sub
```

Delphi

```
{ 다음 예제는 단축 속도 이송 수행 중에 속도 오버라이드 함수를 통해
 속도를 변경하는 예제 입니다. }

procedure OnMove ();
begin
        // 속도 환경을 설정합니다.
        cmmCfgSetSpeedPattern (cmX1, cmSMODE_S, 1000, 10000, 10000 );

        // 단축 속도 이송을 수행합니다.
        cmmSxMoveStart (cmX1, cmDIR_P );
end;


procedure OnOverrideSpeedHigh ()
begin
        // 속도 오버라이드 할 축 및 속도를 지정합니다.
        cmmCfgSetSpeedPattern (cmX1, cmSMODE_KEEP, 2000, 10000, 10000 );
        // 또는 cmmCfgSetSpeedPattern (cmX1, cmSMODE_KEEP, 200, 100, 100 );

        // 진행 중인 이송 작업에 대해 속도를 오버라이드 합니다.
        cmmOverrideSpeedSet (cmX1 );

end;


procedure OnOverrideSpeedLow ();
begin
        // 속도 오버라이드 할 축 및 속도를 지정합니다.
        cmmCfgSetSpeedPattern (cmX1, cmSMODE_KEEP, 500, 10000, 10000 );
        // 또는 cmmCfgSetSpeedPattern (cmX1, cmSMODE_KEEP, 50, 100, 100 );

        //진행 중인 이송 작업에 대해 속도를 오버라이드 합니다.
        cmmOverrideSpeedSet (cmX1 );

end;
```

NAME

cmmOverrideSpeedSetAll
– 다축속도(多軸速度) 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 cmmOverrideSpeedSetAll ([in] VT_I4 NumAxes, [in] VT_PI4 AxisList)

DESCRIPTION

이 함수는 cmmOverrideSpeedSet()과 같이 각 축의 속도를 오버라이딩하는 함수이지만 여러 축을 동시에 오버라이딩할 수 있다는 것이 차이점입니다.

PARAMETER

▶ nNumAxes : 동시에 작업을 수행할 대상 축의 수

▶ AxisList : 동시에 작업을 수행할 대상 축의 배열 주소값. 이 배열의 크기는 NumAxes 값과 일치해야 합니다.

RETURN VALUE

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

EXAMPLE

본 예제는 cmmOverrideSpeedSetAll()함수를 사용하여 속도를 오버라이딩하는 것을 예로 보여주는 코드입니다. 본 예제는 "HIGH" 와 "LOW" 로 이름지어진 두 개의 버튼이 있다고 가정하고 "HIGH" 버튼이 눌리면 X1 축과 Y1 축의 속도를 20000 으로 설정하고 "LOW" 버튼이 눌리면 속도를 10000 으로 설정하는 예입니다.

C/C++

```
#include "Cmmsdk.h"
#include "CmmsdkDef.h"

#define V_LOW   10000       // 저속모드 속도
#define V_HIGH 20000       // 고속모드 속도
#define N_AXES 2             // 2 축

int nAxisList[N_AXES]={cmX1, cmY1};
int nDirList[N_AXES]={cmDIR_P, cmDIR_P};

/*************************************************************
* OnProgramInitial : 이 함수는 가상의 함수로서 프로그램 초기화 루틴이
* 적용되는 부분을 의미합니다.
 *************************************************************/
void OnProgramInitial()
{
```

```
	long m_nNumAexs;
	cmmLoadDll();
	if(cmmGnDeviceLoad(cmTRUE, &m_nNumAxes) != cmERR_NONE)
	{
		//Handle은 사용자가 생성한 폼의 핸들 값입니다.
		cmmErrShowLast(Handle);
		return;
	}
}

/************************************************************
* OnStart() : 이 함수는 가상의 함수로서 X축에 대하여
* V-MOVE 모션을 시작합니다.
************************************************************/
void OnStart()
{
	long nIsDone;
	// 해당축이 작업중이면 정지(停止)하고 다시 시작 //
	cmmSxIsDone(cmX1, &nIsDone);
	if(nIsDone != cmTRUE) cmmSxStopEmg(cmX1);

	cmmSxIsDone(cmY1, &nIsDone);
	if(nIsDone != cmTRUE) cmmSxStopEmg(cmY1);

	// 속도설정 => 시작은 LOW 속도로 시작 //
	cmmCfgSetSpeedPattern(cmX1, cmSMODE_S, V_LOW, 50000, 50000);
	cmmCfgSetSpeedPattern(cmY1, cmSMODE_S, V_LOW, 50000, 50000);
	// V-Move start (X & Y)//
	if(!cmmMxVMoveStart(N_AXES, nAxisList, nDirList)){
				cmmErrShowLast(Handle);
				return;
	}
}

/************************************************************
* OnHighButtonClick() : "HIGH" 버튼 콜백함수 (가상함수)
* "HIGH"버튼이 클릭되면 속도를 V_HIGH 속도로 오버라이드한다.
************************************************************/
void OnHighButtonClick()
{
	// V_HIGH 속도로 오버라이딩 //
	cmmSxSetSpeedRatio(cmX1, cmSMODE_S,  200, 100, 100);
	cmmSxSetSpeedRatio(cmY1, cmSMODE_S,  200, 100, 100);
	//아래 코드로 대체가 가능합니다.
	//cmmCfgSetSpeedPattern(cmX1, cmSMODE_S, V_HIGH, 50000, 50000);
	//cmmCfgSetSpeedPattern(cmY1, cmSMODE_S, V_HIGH, 50000, 50000);
	//
	if(cmmOverrideSpeedSetAll (N_AXES, nAxisList) != cmERR_NONE){
		cmmErrShowLast(Handle);
	}
}
/************************************************************
* OnLowButtonClick() : "LOW" 버튼 콜백함수 (가상함수)
* "LOW"버튼이 클릭되면 속도를 V_LOW 속도로 오버라이드한다.
************************************************************/
void OnLowButtonClick()
{
	// V_LOW 속도로 오버라이딩 //
	cmmSxSetSpeedRatio(cmX1, cmSMODE_S,  100, 100, 100);
	cmmSxSetSpeedRatio(cmY1, cmSMODE_S,  100, 100, 100);
	//아래 코드로 대체가 가능합니다.
	//cmmCfgSetSpeedPattern(cmX1, cmSMODE_S, V_LOW, 50000, 50000);
	//cmmCfgSetSpeedPattern(cmY1, cmSMODE_S, V_LOW, 50000, 50000);
	//
	if(cmmOverrideSpeedSetAll (N_AXES, nAxisList) != cmERR_NONE){
		cmmErrShowLast(Handle);
	}
}

/************************************************************
* OnStop() : "Stop"명령시에 호출되는 가상의 함수
```

```
*************************************************************/
void OnStop()
{
	cmmMxStopEmg(N_AXES, nAxisList);
}
```

Visual Basic

```
'/*************************************************************
'* 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 cmmSxIsDone(cmX1, nIsDone)
	If nIsDone <> cmTRUE Then
	   Call cmmSxStopEmg(cmX1)
	End If

	Call cmmSxIsDone(cmY1, nIsDone)
	If nIsDone <> cmTRUE Then
	   Call cmmSxStopEmg(cmY1)
	End If


	'시작은 LOW 속도로 시작
	Call cmmCfgSetSpeedPattern(cmX1, cmSMODE_S, 10000, 50000, 50000)
	Call cmmCfgSetSpeedPattern(cmY1, cmSMODE_S, 10000, 50000, 50000)

	'V-Move Start ( X, Y )
	If (Not (cmmMxVMoveStart(2, nAxisList(0), nDirList(0)))) Then
	   Call cmmErrShowLast(thisform.Hwnd)
	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 cmmSxSetSpeedRatio(cmX1, cmSMODE_S, 200, 100, 100)
	Call cmmSxSetSpeedRatio(cmY1, cmSMODE_S, 200, 100, 100)

	'//아래 코드로 대체가 가능합니다.
	'//Call cmmCfgSetSpeedPattern(cmX1, cmSMODE_S, 20000, 50000, 50000)
	'//Call cmmCfgSetSpeedPattern(cmY1, cmSMODE_S, 20000, 50000, 50000)
	'//

	If cmmOverrideSpeedSetAll(2, nAxisList(0)) <> cmERR_NONE Then
	   Call cmmErrShowLast(thisform.Hwnd)
	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 cmmSxSetSpeedRatio(cmX1, cmSMODE_S, 100, 100, 100)
	Call cmmSxSetSpeedRatio(cmY1, cmSMODE_S, 100, 100, 100)

	'//아래 코드로 대체가 가능합니다.
```

```
        '//Call cmmCfgSetSpeedPattern(cmX1, cmSMODE_S, 10000, 50000, 50000)
        '//Call cmmCfgSetSpeedPattern(cmY1, cmSMODE_S, 10000, 50000, 50000)
        '//

        If cmmOverrideSpeedSetAll(2, nAxisList(0)) <> cmERR_NONE Then
           Call cmmErrShowLast(thisform.Hwnd)
        End If

End Sub
```

```
Delphi

Const V_LOW =  10000;       // 저속모드 속도
Const V_HIGH = 20000;       // 고속모드 속도
Const N_AXES = 2;           // 2 축
Const Handle = 1;

var
        nAxisList : Array[0..2] of LongInt;
        nDirList : Array[0..2] of LongInt;

//***************************************************************
// OnProgramInitial : 이 함수는 가상의 함수로서 프로그램 초기화 루틴이
// 적용되는 부분을 의미합니다.
 //***************************************************************/

procedure OnProgramInitial();
var
        m_nNumAxes : LongInt;

begin
        if(cmmGnDeviceLoad(cmTRUE, @m_nNumAxes) <> cmERR_NONE) then
        begin
                //Handle 은 사용자가 생성한 폼의 핸들 값입니다.
                cmmErrShowLast(Handle);
                exit;
        end;
end;

//*************************************************************
// OnStart() : 이 함수는 가상의 함수로서 X 축에 대하여
// V-MOVE 모션을 시작합니다.
//*************************************************************/
procedure OnStart();
var
        nIsDone : LongInt;
begin
        // 해당축이 작업중이면 정지(停止)하고 다시 시작 //
        cmmSxIsDone(cmX1, @nIsDone);
        if(nIsDone <> cmTRUE) then
        begin
cmmSxStopEmg(cmX1);
        end;

        cmmSxIsDone(cmY1, @nIsDone);

        if(nIsDone <> cmTRUE) then
        begin
cmmSxStopEmg(cmY1);
        end;

        // 속도설정 => 시작은 LOW 속도로 시작 //
        cmmCfgSetSpeedPattern(cmX1, cmSMODE_S, V_LOW, 50000, 50000);
        cmmCfgSetSpeedPattern(cmY1, cmSMODE_S, V_LOW, 50000, 50000);
        // V-Move start (X & Y)//
        if(cmmMxVMoveStart(N_AXES, @nAxisList, @nDirList)<>cmFALSE)then
        begin
                cmmErrShowLast(Handle);
                exit;
        end;
end;
```

```
//************************************************************
//OnHighButtonClick() : "HIGH" 버튼 콜백함수 (가상함수)
//"HIGH"버튼이 클릭되면 속도를 V_HIGH 속도로 오버라이드한다.
//************************************************************/
procedure OnHighButtonClick();
begin
	// V_HIGH 속도로 오버라이딩 //
	cmmSxSetSpeedRatio(cmX1, cmSMODE_S,  200, 100, 100);
	cmmSxSetSpeedRatio(cmY1, cmSMODE_S,  200, 100, 100);
	//아래 코드로 대체가 가능합니다.
	//cmmCfgSetSpeedPattern(cmX1, cmSMODE_S, V_HIGH, 50000, 50000);
	//cmmCfgSetSpeedPattern(cmY1, cmSMODE_S, V_HIGH, 50000, 50000);

	if(cmmOverrideSpeedSetAll (N_AXES, @nAxisList) <> cmERR_NONE)then
	begin
		cmmErrShowLast(Handle);
	end;
end;
//************************************************************
//OnLowButtonClick() : "LOW" 버튼 콜백함수 (가상함수)
//"LOW"버튼이 클릭되면 속도를 V_LOW 속도로 오버라이드한다.
//************************************************************/
procedure OnLowButtonClick();
begin
	// V_LOW 속도로 오버라이딩 //
	cmmSxSetSpeedRatio(cmX1, cmSMODE_S,  100, 100, 100);
	cmmSxSetSpeedRatio(cmY1, cmSMODE_S,  100, 100, 100);
	//아래 코드로 대체가 가능합니다.
	//cmmCfgSetSpeedPattern(cmX1, cmSMODE_S, V_LOW, 50000, 50000);
	//cmmCfgSetSpeedPattern(cmY1, cmSMODE_S, V_LOW, 50000, 50000);

	if(cmmOverrideSpeedSetAll (N_AXES, @nAxisList) <> cmERR_NONE) then
	begin
		cmmErrShowLast(Handle);
	end;
end;

//************************************************************
// OnStop() : "Stop"명령시에 호출되는 가상의 함수
//************************************************************/
procedure OnStop();
begin
	cmmMxStopEmg(N_AXES, @nAxisList);
end;
```

NAME

cmmOverrideMove
– 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 cmmOverrideMove ([in] VT_I4 Axis, [in] VT_R8 NewDistance, [out] VT_PI4 IsIgnored)

DESCRIPTION

이 함수는 cmmSxMoveStart() 이송 함수를 통하여 수행되는 상대좌표 In-position 모션에 대하여 상대좌표값, 즉 목표 거리값을 오버라이딩하는 함수입니다.

PARAMETER

▶ Axis: 축번호. 축번호는 상수값으로 [cmX1] 부터 0 번째 축을 기준 축으로 임의의 축을 설정할 수 있습니다.

▶ NewDistance : 새로운 목표 거리값을 지정합니다. 이 값의 기준 위치는 오버라이드하고자 하는 대상이 되는 cmmSxMoveStart() 작업에서 사용한 기준점과 같습니다. 즉, 새로운 목표 거리는 cmmSxMoveStart() 함수를 실행하기 바로직전의 위치를 기준으로 계산하여야 합니다.

▶ IsIgnored : cmmOverrideMove 의 적용 성공/실패 여부를 반환 합니다.

Value	Meaning
0	성공. 위치 오버라이드가 적용됨.
1	실패. 모션에러가 발생하였거나 이미 이송이 완료되어 위치 오버라이드가 적용되지 않음.

RETURN VALUE

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

REFERENCE

□ 위치 오버라이드를 수행하려는 시점에 이미 이송이 완료되어 버린 경우에는 위치 오버라이드는 무시되고 Ignored 파라미터 값을 1 로 반환합니다. 따라서 사용자는 Ignored 파라미터 값이 1 인 경우에는 이미 이송이 완료되어 오버라이드가 적용되지 않은 것으로 인지하여야 하며, 그럼에도 불구하고 목표좌표를 수정해야하는 경우에는 cmmSxMove() 또는 cmmSxMoveTo() 함수를 추가적으로 수행해야 합니다. 이러한 경우에는 오버라이드라는 개념보다는 추가 이송의 개념을 의미합니다.

NAME

cmmOverrideMoveTo
– 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 cmmOverrideMoveTo ([in] VT_I4 Axis, [in] VT_R8 NewPosition, [out] VT_PI4 IsIgnored)

DESCRIPTION

이 함수는 cmmSxMoveToStart() 함수를 통하여 수행되는 절대좌표 In-position 모션에 대하여 목표 절대좌표값을 오버라이딩하는 함수입니다.

PARAMETER

▶ Axis: 축번호. 축번호는 상수값으로 [cmX1] 부터 0 번째 축을 기준 축으로 임의의 축을 설정할 수 있습니다.

▶ NewPosition : 새로운 목표 절대좌표값을 지정합니다.

▶ IsIgnored : cmmOverrideMoveTo 의 적용 성공/실패 여부를 반환 합니다.

Value	Meaning
0	성공. 위치 오버라이드가 적용됨.
1	실패. 모션에러가 발생하였거나 이미 이송이 완료되어 위치 오버라이드가 적용되지 않음.

RETURN VALUE

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

REFERENCE

□ 위치 오버라이드를 수행하려는 시점에 이미 이송이 완료되어 버린 경우에는 위치 오버라이드는 무시되고 Ignored 파라미터 값을 1 로 반환합니다. 따라서 사용자는 Ignored 파라미터 값이 1 인 경우에는 이미 이송이 완료되어 오버라이드가 적용되지 않은 것으로 인지하여야 하며, 그럼에도 불구하고 목표좌표를 수정해야하는 경우에는 cmmSxMoveTo() 함수를 추가적으로 수행해야합니다.

9.2 Master/Slave 동기제어

Master/Slave 동기제어는 두 개의 모터축이 하나의 모터처럼 완전한 동기제어가 필요한 시스템에 사용되는 기능입니다. ㈜커미조아 모션컨트롤러에서 제공하는 Master/Slave 동기제어 기능은 Manual Pulsar 모드를 응용한 것입니다. 이 기능을 사용하면 두 축을 하나의 축처럼 제어하는 것이 가능합니다. Master/Slave 동기제어 모드 상태에서 Slave축은 자동적으로 Master축과 동기되어서 동작하므로 사용자는 Master축만 제어하면 됩니다. 이 것은 제어의 편리성을 제공하는 동시에 Master/Slave 축을 하나의 가상축으로 하여 다른 축과 원호보간제어를 수행하는 것을 가능하게 합니다.

9.2.1 Master/Slave 하드웨어 스위치 설정

Master/Slave 기능을 사용하기 위해서는 하드웨어적인 "SYNC" 스위치를 적절하게 설정하여야 합니다. 스위치 설정에 대한 자세한 내용은 PDF 파일로 제공되는 각 제품별 하드웨어 매뉴얼을 참조하시기 바랍니다.

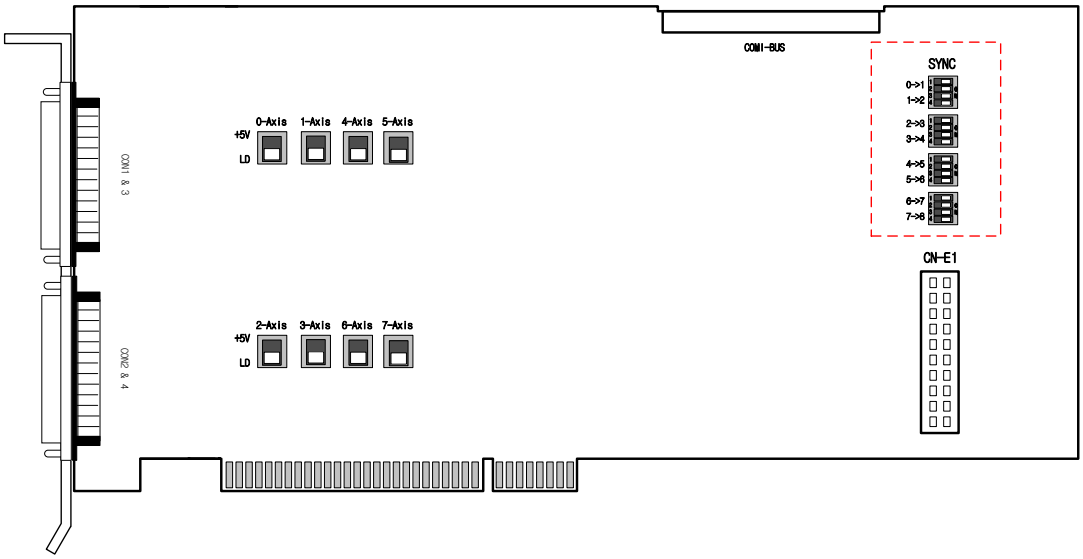


그림 9-1 LX508 기준 Master/Slave 모드 설정 하드웨어 스위치 위치 안내

9.2.2 함수 요약

Master/Slave 동기제어에 관련된 함수들은 다음과 같습니다.

Summary of Functions
❒ VT_I4 cmmMsRegisterSlave ([in] VT_I4 Axis, [in] VT_R8 MaxSpeed, [in] VT_I4 IsInverse) 대상 모션 채널에 대해서, Master / Slave 동기(同期) 구동(驅動)의 Slave 축으로 등록합니다. 해당 축의 이전 축은 묵시적으로 Master 축이 되며, Slave 축은 Master 축과 동기(同期) 되어 구동(驅動)됩니다.
❒ VT_I4 cmmMsUnregisterSlave ([in] VT_I4 Axis) 대상 모션 채널에 대해서, Master / Slave 동기(同期) 구동(驅動)의 Slave 축으로서의, 자격을 명시적으로 소멸(消滅)합니다. 이 함수의 호출 후에 대상 모션 채널은 Slave 모드가 해제됩니다.
❒ VT_I4 cmmMsCheckSlaveState ([in] VT_I4 SlaveAxis, [out] VT_PI4 SlaveState) 대상 모션 채널에 대해서, Master / Slave 동기(同期) 구동(驅動)의 Slave 축으로서의, 자격에 대해서 조회합니다. 이 함수를 통해 대상 모션 채널이 Slave 축으로서 정상 등록되어 있는지에 대한 여부에 대한 상태를 알 수 있습니다.
❒ VT_I4 cmmMsGetMasterAxis ([in] VT_I4 SlaveAxis, [out] VT_PI4 MasterAxis) 대상 모션 채널에 대해서, Master / Slave 동기(同期) 구동(驅動)의 Master 축을 조회하여, 해당 Slave 축의 Master 축의 번호를 반환(返還)합니다.

9.2.3 함수 설명

NAME

cmmMsRegisterSlave
cmmMsUnregisterSlave
– 지정한 축을 동기(同期) 제어 대상 (對象) 축으로 설정(設定)

INFORMATION

- Master/Slave 동기 제어
- VC++/VB
- BCB/Delphi/.NET
- Level 5
- 위험 요소 없음

SYNOPSIS

❒ VT_I4 cmmMsRegisterSlave

([in] VT_I4 Axis, [in] VT_R8 MaxSpeed, [in] VT_I4 IsInverse)

❒ VT_I4 cmmMsUnregisterSlave

([in] VT_I4 Axis)

DESCRIPTION

지정한 축을 Slave 축으로 등록/해제 합니다. 이때 마스터축은 자동으로 결정되며, 축 번호상 앞의 축이 자동적으로 Master 축으로 됩니다. 단, 해당 축의 Master/Slave 모드를 설정하는 하드웨어 스위치가 연결되지 않은 경우에는 Slave 축이 등록되어도 Master/Slave 동작이 정상적으로 동작하지 않습니다.

PARAMETER

▶ Axis: Slave 축으로 지정/해제할 축번호. 축 번호는 0 부터 시작합니다.

▶ MaxSpeed : Slave 축이 구동될 수 있는 최대속도. Master 축이 이 값보다 큰 속도로 구동되면 오동작할 수 있습니다. 속도의 단위는 "Unit speed"에 의해 정의되는 논리적 속도입니다.

▶ IsInverse : Slave 축의 구동방향과 Master 축의 구동방향을 반대로 할 것인지를 결정합니다.

Value	Meaning
0 또는 cmFALSE	Master 축과 Slave 축의 구동방향을 동일하게 함.
1 또는 cmTRUE	Master 축과 Slave 축의 구동방향을 반대로 함.

RETURN VALUE

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

REFERENCE

□ 이 함수가 수행되면 지정한 축은 Manual Pulsar 모드로 전환되게 됩니다. Maser/Slave 에 관련된 하드웨어 스위치 설정이 적절하게 되었으면 지정한 축의 PA/PB 입력핀에 Master 축의 Command 펄스 신호가 자동으로 입력되고, 따라서 두 축이 동기제어됩니다

□ Master 축의 Command 출력모드는 반드시 "CW & CCW 모드 (출력모드 4 ~ 5 번)"로 설정되어 있어야 합니다. Master 축의 Command 출력모드가 "Pulse & Direction 모드 (출력모드 0 ~ 3 번)"로 설정된 경우에는 Slave 축이 한쪽방향으로만 구동되게 됩니다. Command 출력모드의 설정에 관해서는 cmmCfgSetOutMode() 함수 설명편을 참조하시기 바랍니다.

□ Master 축의 Command 출력 모드가 "Pulse & Direction 모드" 로 설정된 경우 (출력모드 0~3) 에는 Master / Slave 기능을 양방향에서 원할하게 사용할 수 없습니다.

□ 이 함수가 성공적으로 수행되었으면 cmmStReadMotionState() 함수의 반환값이 8(cmMST_WAIT_PLSR)이 됩니다. 이를 확인(確認)하면 지정한 축이 Slave 모드로 전환되었는지를 확인(確認)할 수 있습니다. 그러나 하드웨어 스위치 설정은 소프트웨어적으로 확인(確認)되지 않습니다.

□ Slave 축은 CLR(Counter Clear) 입력과 INP 입력이 자동으로 Disable 됩니다. 따라서 이 함수를 호출한 이후에 CLR 입력과 INP 입력을 Enable 시키는 환경설정을 수행해서는 안됩니다.

□ Slave 모드가 해제되면 해당 축의 CLR 입력모드와 INP 입력모드는 cmmMsRegisterSlave() 함수가 호출되기 이전의 설정으로 복원됩니다.

EXAMPLE 1

본 예제는 1 번(Y1) 축을 Slave 축으로 설정하여 Master/Slave 동기제어를 수행하는 예제입니다. 1 번축의 마스터축은 자동으로 0 번축으로 설정됩니다. 본 예제에서는 원점복귀 하는 함수를 두 가지 방식으로 예시합니다. 사용자는 두 가지 방식중에 하나를 선택적으로 사용하시면 됩니다.

★. 하드웨어적인 SYNC 스위치 중에서 "0->1" 로 표기된 두 개의 스위치를 연결하여 합니다.

★. 마스터축인 0 번 축의 Command 출력모드는 반드시 "CW & CCW 모드 (출력모드 4 ~ 5 번)"로 설정되어 있어야 합니다.

C/C++

```
#include "Cmmsdk.h"
#include "CmmsdkDef.h"
#define N_AXES 2              // 2 축

/************************************************************
* OnProgramInitial : 이 함수는 가상의 함수로서 프로그램 초기화 루틴이
* 적용되는 부분을 의미합니다.
************************************************************/
void OnProgramInitial()
{

        long m_nNumAxes;

        cmmLoadDll();
        if(cmmGnDeviceLoad(cmTRUE, &m_nNumAxes) != cmERR_NONE)
        {
                //Handle 은 사용자가 생성한 폼의 핸들 값입니다.
                cmmErrShowLast(Handle);
                return;
        }

        // 1 번(Y1) 축을 Slave 축으로 등록합니다. => 마스터는 0 번(X1) 축 //
        if(cmmMsRegisterSlave (cmY1, 500000, cmFALSE) != cmERR_NONE){
                cmmErrShowLast(Handle);
        }
}

/************************************************************
* OnHomeReturn1() : 이 함수에서는 원점복귀시에는 1 번축의 Slave 모드를 해제한 상태에서 0 번과 1 번축이 동시에
원점복귀를 수행한 후, 원점복귀가 완료되면 다시 1 번축의 Slave 모드를 복원하는 방식을 취하였습니다.
* OnHomeReturn1() 대신에 OnHomeReturn2() 방식을 사용할 수도 있습니다.
************************************************************/
void OnHomeReturn1()
{
        ////////////////////////////////////////////////////////
        // 원점복귀를 하기 위해 1 번축을 Slave 모드에서 해제한다.
        if(cmmMsUnregisterSlave (cmY1) != cmERR_NONE){
                cmmErrShowLast(Handle);
                return;
        }

        //X1 축과 Y1 축의 홈복귀 속도 패턴을 설정 합니다.  참고적으로 역방향 속도는 100 으로 설정하였습니다.
        cmmHomeSetSpeedPattern(cmX1, cmSMODE_S, 10000, 30000, 30000, 100);
```

```
	cmmHomeSetSpeedPattern(cmY1, cmSMODE_S, 10000, 30000, 30000, 100);

	// X1&Y1 축 원점복귀 동시 수행//
	long nAxes[2]={cmX1, cmY1};
	long nDirList[2]={cmDIR_N, cmDIR_N};
	if(cmmHomeMoveAll(2, nAxes, nDirList, cmFALSE) != cmERR_NONE){
		cmmErrShowLast(Handle);
		return;
	}

	// 1 번(Y1) 축을 Slave 축으로 다시 등록합니다. //
	if(cmmMsRegisterSlave (cmY1, 500000, cmFALSE) != cmERR_NONE){
		cmmErrShowLast(Handle);
	}
}

/*************************************************************
* OnHomeReturn2() : 이 함수는 Master/Slave 시스템에서의 원점복귀를 수행하는 함수입니다. 이 함수에서는
원점복귀시에도 Master/Slave 모드를 유지한채 0 번축만 원점복귀를 수행합니다. 1 번축은 Slave 이므로 0 번축의
모션을 똑같이 추종합니다. OnHomeReturn2() 대신에 OnHomeReturn1() 방식을 사용할 수도 있습니다.
*************************************************************/
void OnHomeReturn2()
{
	// X1 축 홈복귀 속도패턴 설정 //
	cmmHomeSetSpeedPattern(cmX1, cmSMODE_S, 10000, 30000, 30000);

	// X1 축 원점복귀 시작 //
	if(cmmHomeMoveStart(cmX1, cmDIR_N) != cmERR_NONE){
		cmmErrShowLast(Handle);
		return;
	}
	if(cmmHomeWaitDone(cmX1, cmFALSE) != cmERR_NONE){
		cmmErrShowLast(Handle);
		return;
	}
}
/*************************************************************
* OnDoMotion() : 모션을 수행하는 가상의 함수. 이 함수에서는 0 번축만 제어
* 하지만 실제제어되는 것은 0 번축과 1 번축이 함께 제어됩니다.
*************************************************************/
void OnDoMotion()
{
	cmmCfgSetSpeedPattern(cmX1, cmSMODE_S, 10000, 100000, 100000);

	cmmSxMove(cmX1, 1000, cmFALSE); // Move 1000
	cmmSxMove(cmX1, 1000, cmFALSE); // Move 1000
	cmmSxMove(cmX1, -1000, cmFALSE); // Move -1000
	cmmSxMove(cmX1, -1000, cmFALSE); // Move -1000
}

/*************************************************************
* OnStop() : "Stop"명령시에 호출되는 가상의 함수
*************************************************************/
void OnStop()
{
	cmmMxStopEmg(N_AXES, nAxisList);
}
```

Visual Basic

```
'/*************************************************************
'* OnProgramInitial : 이 함수는 가상의 함수로서 프로그램 초기화 루틴이
'* 적용되는 부분을 의미합니다.
'*************************************************************

Private Sub OnProgramInitial()
	If cmmGnDeviceLoad(cmTRUE, m_nNumAxes) <> cmERR_NONE Then
	'Handle 은 사용자가 생성한 폼의 핸들 값입니다.
	   Call cmmErrShowLast(thisform.Hwnd)
	End
```

```
End If

        '1 번(Y1) 축을 Slave 축으로 등록합니다. => 마스터는 0 번(X1) 축
        If cmmMsRegisterSlave(cmY1, 500000, cmFALSE) <> cmERR_NONE Then
           Call cmmErrShowLast(thisform.Hwnd)
        End If

End Sub

'/*************************************************************
'* OnHomeReturn1() : 이 함수에서는 원점복귀시에는
'* 1 번축의 Slave 모드를 해제한 상태에서 0 번과 1 번축이 동시에 원점복귀를
'* 수행한 후, 원점복귀가 완료되면 다시 1 번축의 Slave 모드를 복원하는
'* 방식을 취하였습니다.
'* OnHomeReturn1() 대신에 OnHomeReturn2() 방식을 사용할 수도 있습니다.
'*************************************************************

Private Sub OnHomeReturn1()

        ' // 원점복귀를 하기 위해 1 번축을 Slave 모드에서 해제한다.
        If cmmMsUnregisterSlave(cmY1) <> cmERR_NONE Then
           Call cmmErrShowLast(thisform.Hwnd)
           End
        End If

        '//X1 축과 Y1 축의 홈복귀 속도 패턴을 설정 합니다.
        Call cmmHomeSetSpeedPattern(cmX1, cmSMODE_S, 10000, 30000, 30000, 100)
        Call cmmHomeSetSpeedPattern(cmY1, cmSMODE_S, 10000, 30000, 30000, 100)

        '// X1&Y1 축 원점복귀 동시 수행//
        Dim nAxes(2) As Long
        Dim ndirList(2) As Long

        nAxes(0) = cmX1
        nAxes(1) = cmY1

        ndirList(0) = cmDIR_N
        ndirList(1) = cmDIR_N

        If cmmHomeMoveAll(2, nAxes(0), ndirList(0), cmFALSE) <> cmERR_NONE Then
           Call cmmErrShowLast(thisform.Hwnd)
           End
        End If

        '// 1 번(Y1) 축을 Slave 축으로 다시 등록합니다. //
        If cmmMsRegisterSlave(cmY1, 500000, cmFALSE) <> cmERR_NONE Then
           Call cmmErrShowLast(thisform.Hwnd)
        End If
End Sub

'/*************************************************************
'* OnHomeReturn2(): 이 함수는 Master/Slave 시스템에서의 원점복귀를 수행
'* 하는 함수입니다. 이 함수에서는 원점복귀시에도 Master/Slave 모드를 유지한
'* 채 0 번축만 원점복귀를 수행합니다.1 번축은 Slave 이므로 0 번축의 모션을 똑
'* 같이 추종합니다. OnHomeReturn2()대신에 OnHomeReturn1() 방식을 사용할
'* 수도 있습니다.
'*************************************************************/
Private Sub OnHomeReturn2()

        '// X1 축 홈복귀 속도패턴 설정 //
        Call cmmHomeSetSpeedPattern(cmX1, cmSMODE_S, 10000, 30000, 30000)

        '// X1 축 원점복귀 시작 //
        If cmmHomeMoveStart(cmX1, cmDIR_N) <> cmERR_NONE Then
           cmmErrShowLast (thisform.Hwnd)
           End
        End If

        If cmmHomeWaitDone(cmX1, cmFALSE) <> cmERR_NONE Then
           cmmErrShowLast (thisform.Hwnd)
           End
```

```
        End If

End Sub

'/*************************************************************
'* OnDoMotion() : 모션을 수행하는 가상의 함수. 이 함수에서는 0 번축만 제어
'* 하지만 실제제어되는 것은 0 번축과 1 번축이 함께 제어됩니다.
'*************************************************************/
Private Sub OnDoMotion()

        Call cmmCfgSetSpeedPattern(cmX1, cmSMODE_S, 10000, 100000, 100000)

        Call cmmSxMove(cmX1, 1000, cmFALSE)  '// Move 1000
        Call cmmSxMove(cmX1, 1000, cmFALSE)  '// Move 1000
        Call cmmSxMove(cmX1, -1000, cmFALSE) '// Move -1000
        Call cmmSxMove(cmX1, -1000, cmFALSE) '// Move -1000

End Sub

'/*************************************************************
'* OnStop() : "Stop"명령시에 호출되는 가상의 함수
'*************************************************************/
Private Sub OnStop()

        Call cmmMxStopEmg(2, nAxisList(0))

End Sub
```

```
Delphi


// ****************************************************************************
// Master/Slave 동기 구동을 하기 위해 슬레이브 축을 등록 합니다.
// ****************************************************************************
procedure OnSetMsRegisterSalve ();
var
        nSlaveState : LongInt;                  // 슬레이브 축 등록 상태 정보.
        nMasterAxis : LongInt;                          // 마스터 축 정보.

begin

        // 1 번 축을 슬레이브 축으로 등록합니다. 자동으로 마스터 축은 0(cmX1)번 축이 됩니다.
        if cmmMsRegisterSlave (cmY1, 655350, cmFALSE          ) = ceERR_NONE then
        begin
                // 슬레이브 축의 등록 상태를 확인 합니다.
                cmmMsCheckSlaveState (cmY1, @nSlaveState );

                if nSlaveState <> cmTRUE then
                begin
                        ShowMessage ( 'Slave axis registered failed' );
                end;

                // 슬레이브 축의 마스터 축을 확인합니다.
                cmmMsGetMasterAxis (cmY1, @nMasterAxis );

                if nMasterAxis <> cmX1 then
                begin
                        ShowMessage ( 'Slave axis registered failed' );
                end;
        end;

end;


// ****************************************************************************
// 마스터 축에 대해 단축 이송 작업을 수행합니다. 슬레이브 축이 등록
// 되어 있으므로 마스터 축 제어 시 슬레이브 축이 함께 제어됩니다.
// ****************************************************************************
procedure OnMove ();
begin
```

```
    // 마스터 축 속도 환경을 설정합니다.
    cmmCfgSetSpeedPattern (cmX1, cmSMODE_S, 5000, 30000, 30000 );

    // 마스터 축 단축 이송 수행 시 슬레이브 축이 함께 이송합니다.
    cmmSxMove (cmX1, 15000, cmFALSE );

end;
```

NAME

cmmMsCheckSlaveState
– 동기 (同期) 제어 대상 (對象) 축 상태 확인(確認)

INFORMATION

Master/Slave 동기 제어
VC++/VB
BCB/Delphi/.NET
Level 5
위험 요소 없음

SYNOPSIS

❒ VT_I4 cmmMsCheckSlaveState

([in] VT_I4 SlaveAxis, [out] VT_PI4 SlaveState)

DESCRIPTION

지정한 축이 현재 Slave 축으로 정상 등록 되어 있는지에 대한 상태를 반환합니다.

PARAMETER

▶ SlaveAxis : Slave 상태를 확인(確認)할 대상 축 번호. 축 번호는 0 부터 시작합니다.
▶ SlaveState : Slave 를 반환합니다. Slave 축의 상태는 다음과 같이 반환됩니다.

Value	Meaning
-1	해당 축이 Slave 축으로 등록되어 있지만 모션 에러가 발생하였음을 의미합니다. 이 때는 Slave 축을 다시 등록할 필요는 없지만 에러 상황은 해제 시켜주어야 합니다.
0	해당 축이 Slave 축으로 등록되어 있지 않습니다.
1	해당 축이 Slave 축으로서 등록되어 있으며 정상 동작을 하고 있습니다.

RETURN VALUE

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

NAME

cmmMsGetMasterAxis
– 동기 (同期) 제어 대상 (對象) 축의 원본 축의 번호 반환 (返還)

INFORMATION

- Master/Slave 동기 제어
- VC++/VB
- BCB/Delphi/.NET
- Level 5
- 위험 요소 없음

SYNOPSIS

❒ VT_I4 cmmMsGetMasterAxis

([in] VT_I4 SlaveAxis, [out] VT_PI4 MasterAxis)

DESCRIPTION

지정한 Slave 축의 마스터축으로 동작하는 축의 번호를 반환합니다.

PARAMETER

▶ SlaveAxis : Slave 축 번호. 축 번호는 0 부터 시작합니다.

▶ MasterAxis : 얻어온 마스터 축 번호.

RETURN VALUE

지정한 Slave 축의 마스터축으로 동작하는 축의 번호를 반환합니다.

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

REFERENCE

□ Master/Slave 기능을 사용할 때 Master 축 번호는 Slave 축에 따라서 자동으로 결정되면, 축 번호상 앞의 축이 자동적으로 Master 축으로 됩니다.

단, 해당 축의 Master/Slave 모드를 설정하는 하드웨어 스위치가 연결되지 않은 경우에는 Slave 축이 등록되어도 Master/Slave 동작이 정상적으로 동작하지 않습니다. 그리고 하드웨어적인 스위치 설정 상태는 소프트웨어적으로 확인(確認)할 수 없습니다. 따라서 사용자가 하드웨어 스위치 설정 상태는 반드시 직접 보드상에 스위치 위치를 육안으로 확인(確認)하셔야 합니다.

9.3 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 또는 2 축 직선보간이 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 9-2 와 같이 나사산 모양의 작업을 수행할 수 있습니다.

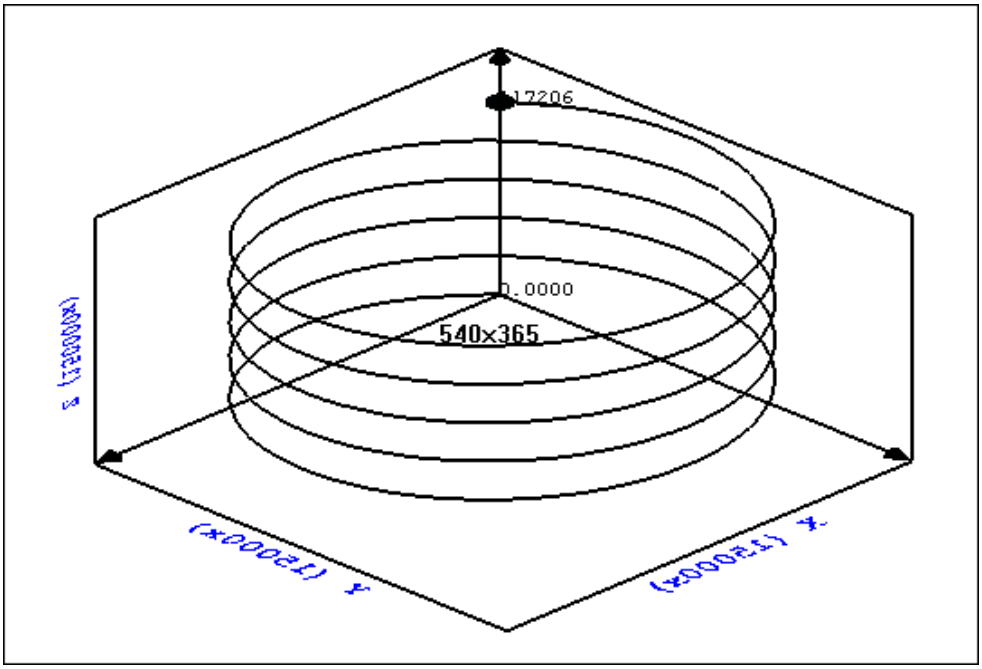


그림 9-2 연속적인 헬리컬 보간작업

□ 헬리컬 보간 제어에서 U축의 역할

헬리컬 보간 동작 시에 U축은 항상 원호 보간의 원주 길이에 해당하는 거리만큼 직선 이송을 자동으로 수행하게 됩니다. 그리고 원호 보간 이송은 U축에 동기되어서 수행됩니다. 따라서 U축의 속도가 원호 보간의 속도를 결정하며, 원호보간에서도 가감속을 수행할 수 있습니다. 단 이때 U축의 이송량은 원주 길이에 해당하는 거리이며 자동으로 설정됩니다.

그림 9-2 과 같이 원호 보간을 임의 거리의 직선 이송과 동기하여 수행하려면 대부분의 경우에 U축 대신에 Z축을 추가적으로 사용하여야 할 것입니다. 왜냐하면 U축의 이송량은 자동으로 계산되며, 사용자가 임의로 지정할 수 없기 때문입니다. 이때에 U축은 X,Y축의 원호보간과 Z축의 직선 이송이 동기될 수 있도록 매개축 역할을 수행합니다. 이때에도 U축에서는 원주 길이에 해당하는 양의 펄스가 출력된다는 것을 주의해야 합니다.

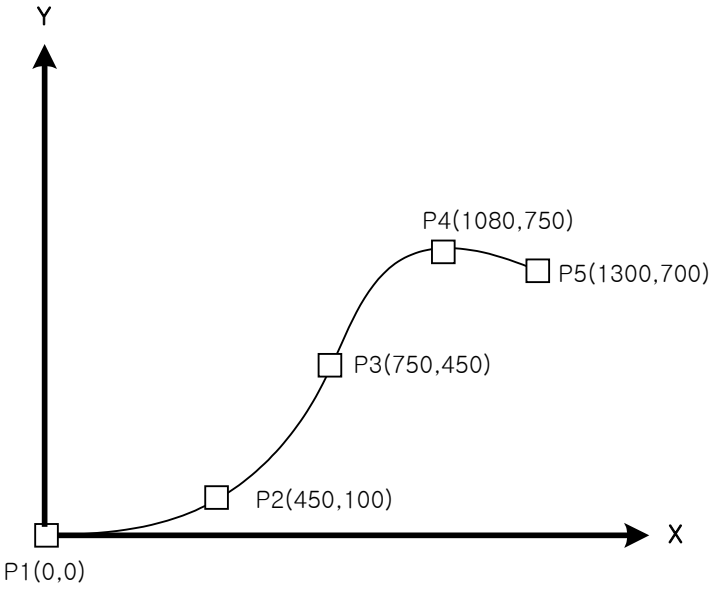
□ 헬리컬 보간 기능 사용 시의 축 맵핑(Mapping)

하나의 헬리컬 보간 이송을 수행하는데 있어서 관여되는 축들을 정의하는 것을 축 맵핑이라고 합니다. 헬리컬 보간 이송에서 축 맵핑은 동일 모션칩(Motion-chip)에 속한 축들로 구성되어야 합니다.

따라서 헬리컬 보간의 축 맵핑은 동일 보드내에 있는 축들로만 구성되어야 합니다. 그리고 8 축 보드인 경우에는 4 축 제어용 모션칩이 2 개 사용된 것이므로 Axis0~3 사이의 축들로만 구성되거나, Axis4~7 사이의 축들로만 구성되어야 합니다.

나) 스플라인 보간제어

㈜커미조아 모션제어 보드는 Cubic spline interpolation 기능을 지원합니다. Cublic spline interpolation은 아래 그림과 같이 사용자가 지정하는 데이터 포인트를 자유곡선으로 보간해주는 기능입니다.



본 라이브러리에서 제공하는 스플라인 보간 기능은 사용자가 지정한 포인트들을 기반으로 Cubic spline 보간기법을 사용하여 자유곡선 포인트들을 생성해주는 기능입니다. 따라서 이를 직접 모션으로 구동하기 위해서는 리스트모션 기능과 함께 사용하여야 합니다. 스플라인 보간 기능과 리스트모션 기능을 함께 사용하면 효과적으로 스플라인 보간 구동을 수행할 수 있습니다.

9.3.1 함수 요약

헬리컬 보간제어와 관련된 함수들은 다음의 표와 같습니다.

Summary of Functions
❒ VT_I4 cmmIxxHelOnceSetSpeed ([in] VT_I4 HelId, [in] VT_I4 Master, [in] VT_I4 SpeedMode, [in] VT_R8 WorkSpeed, [in] VT_R8 Acc, [in] VT_R8 Dec) 대상(對象) 모션 채널에 대해서, 헬리컬보간(補間) 작업을 수행할 때의 속도 패턴 및 관련 환경(環境)을 설정(設定)합니다.
❒ VT_I4 cmmIxxHelOnceGetSpeed ([in] VT_I4 HelId, [out] VT_PI4 SpeedMaster, [out] VT_PI4 SpeedMode, [out] VT_PR8 WorkSpeed, [out] VT_PR8 Acc, [out] VT_PR8 Dec) 대상(對象) 모션 채널에 대해서, 헬리컬 보간(補間) 작업을 수행할 때의 속도 패턴 및 관련 환경(環境)을 반환(返還)합니다.
❒ VT_I4 cmmIxxHelOnce ([in] VT_I4 HelId, [in] VT_I4 NumAxes, [in] VT_PI4 AxisList, [in] VT_PR8 CoordList, [in] VT_R8 ArcAngle, [out] VT_PR8 DistU, [in] VT_I4 IsBlocking) 대상(對象) 모션 채널에 대해서, 2 축 원호간과 1 축또는 2 축 직선 보간 (補間)을 동시에 시작하고 동시에 종료하는 헬러킬 보간(補間) 구동을 동작합니다. 이 구동 함수는 구동 시작 후 바로 반환(返還)되지 않습니다.
❒ VT_I4 cmmIxxHelOnceStart ([in] VT_I4 HelId, [in] VT_I4 NumAxes, [in] VT_PI4 AxisList, [in] VT_PR8 CoordList, [in] VT_R8 ArcAngle, [out] VT_PR8 DistU) 대상(對象) 모션 채널에 대해서, 2 축 원호간과 1 축또는 2 축 직선 보간 (補間)을 동시에 시작하고 동시에 종료하는 헬러킬 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 cmmIxxSplineBuild ([in] VT_PR8 InArray, [in] VT_I4 NumInArray, [out] VT_PR8 OutArray, [in] VT_I4 NumOutArray) Cubic Spline 보간 구동의 목표 좌표를 생성하기 위해, 전달된 샘플 좌표를 통해 생성 후 최종 좌표를 반환(返還)합니다. 이 함수는 순수한 Cublic Spline 보간법의 좌표만을 생성합니다.

9.3.2 함수 설명

NAME

cmmIxxHelOnceSetSpeed
cmmIxxHelOnceGetSpeed
– 헬리컬 보간 이송 속도설정(速度設定)

INFORMATION

Extended Interpolation Motion
VC++/VB
BCB/Delphi/.NET
Level 5
위험 요소 없음

SYNOPSIS

❒ VT_I4 cmmIxxHelOnceSetSpeed

([in] VT_I4 HelId, [in] VT_I4 Master, [in] VT_I4 SpeedMode, [in] VT_R8 WorkSpeed, [in] VT_R8 Acc, [in] VT_R8 Dec)

❒ VT_I4 cmmIxxHelOnceGetSpeed

([in] VT_I4 HelId, [out] VT_PI4 Master, [out] VT_PI4 SpeedMode, [out] VT_PR8 WorkSpeed, [out] VT_PR8 Acc, [out] VT_PR8 Dec)

DESCRIPTION

확장 보간제어 상에서 헬리컬보간 작업을 수행시의 속도패턴을 설정 하거나 설정된 값을 읽어옵니다.

PARAMETER

▶ HelId : 속도 설정을 하고자 하는 헬리컬 보간 이송 작업의 아이디를 지정합니다. CMMSDK 는 통합라이브러리이므로 여러 개의 보드가 장착되어 있는 경우에 여러 개의 헬리컬 보간 작업을 동시에 수행할 수도 있습니다. 따라서 각각의 작업을 구분지어줄 아이디가 필요합니다. cmmIxxHelOnce() / cmmIxxHelOnceStart() 의 헬리컬 이송 함수를 실행할 때에도 HelId 를 입력하게 되어 있는데, 여기서 지정한 속도 패턴은 동일한 HelId 가 지정된 헬리컬 이송 함수에서 반영됩니다.

▶ Master: cmmIxxHelOnceSetSpeed 함수의 인자이며, 속도 설정의 마스터를 설정합니다. 이 값에 따라서 지정한 속도패턴의 의미가 다음과 같이 달라집니다.

Value	Meaning
0	이 함수를 통하여 설정하는 속도 패턴은 Z 축과 U 축의 직선 보간에 대한 벡터 속도로 적용됩니다(U 축의 이송량은 원호보간의 원주 길이에 해당함). 만일 Z 축이 포함되지 않은 헬리컬 보간이라면 이 모드는 Master=1 로 설정한 것과 동일한 것이 되며 이 함수를 통하여 설정하는 속도 패턴은 U 축의 속도로 적용됩니다.
1	이 함수를 통하여 설정하는 속도 패턴은 U 축의 속도로 적용됩니다. U 축은 원주 길이에 해당하는 이송을 하게 되므로 U 축의 속도는 곧 원호보간의 속도입니다. Z 축의 속도는 원주 길이에 대한 상대적인 거리비에 따라 자동으로 결정됩니다.
2	이 함수를 통하여 설정하는 속도 패턴은 Z 축의 속도로 적용됩니다. 원호보간 속도는 Z 축 이동 거리에 대한 상대적인 거리비에 따라 자동으로 결정됩니다. 만일 Z 축이 포함되지 않은 헬리컬 보간이라면 이 모드는 Master=1 로 설정한 것과 동일한 것이 되며 U 축의 속도로 적용됩니다.

▶ Master: cmmIxxHelOnceGetSpeed 함수의 인자이며, 마스터 속도 설정을 반환합니다.

Value	Meaning
0	Z 축과 U 축의 직선 보간에 대한 벡터 속도를 마스터 속도로 설정된 상태입니다.

1	이 함수를 통하여 설정하는 속도 패턴은 U 축의 속도를 마스터 속도로 설정된 상태입니다.
2	이 함수를 통하여 설정하는 속도 패턴은 Z 축의 속도를 마스터 속도로 설정된 상태입니다.

▶ SpeedMode : cmmIxxHelOnceSetSpeed 함수의 인자이며, 속도 모드를 설정합니다.

Value	Meaning
0 또는 cmSMODE_C	CONSTANT 속도모드 => 가감속을 수행하지 않습니다.
1 또는 cmSMODE_T	TRAPEZOIDAL 속도모드 => 사다리꼴 가감속을 수행합니다.
2 또는 cmSMODE_S	S-CURVE 속도모드 => S-CURVE 가감속을 수행합니다.

▶ SpeedMode : cmmIxxHelOnceGetSpeed 함수의 인자이며, 속도 모드를 반환합니다.

Value	Meaning
0 또는 cmSMODE_C	CONSTANT 속도모드 => 가감속을 수행하지 않습니다.
1 또는 cmSMODE_T	TRAPEZOIDAL 속도모드 => 사다리꼴 가감속을 수행합니다.
2 또는 cmSMODE_S	S-CURVE 속도모드 => S-CURVE 가감속을 수행합니다.

▶ WorkSpeed : cmmIxxHelOnceSetSpeed 함수의 인자이며, 정속도를 지정합니다.

▶ WorkSpeed : cmmIxxHelOnceGetSpeed 함수의 인자이며, 지정된 정속도를 반환합니다.

▶ Acc : cmmIxxHelOnceSetSpeed 함수의 인자이며, 가속도를 지정합니다. 단, 이 값을 0 으로 설정하면 가속구간이 없이 즉시 작업속도로 올라갑니다(실제로는 가속값이 무한대).

▶ Acc : cmmIxxHelOnceGetSpeed 함수의 인자이며, 가속도를 반환합니다.

▶ Dec : cmmIxxHelOnceSetSpeed 함수의 인자이며, 감속도를 지정합니다. 단, 이 값을 0 으로 설정하면 감속구간이 없이 즉시 정지(停止)합니다(실제로는 감속값이 무한대).

▶ Dec : cmmIxxHelOnceGetSpeed 함수의 인자이며, 감속도를 반환합니다.

EXAMPLE

아래의 코드는 360 도 원을 그리면서 Z 축도 +3000 만큼 이송하는 헬리컬 보간을 수행하는 예입니다. 이때 Z 축 속도를 V=5000, ACC=DEC=25000 으로 설정하고, 원호보간 속도는 원주 길이와 Z 축 이동거리의 비에 따라 자동으로 설정되도록 합니다.

```
C/C++
#define HEL_ID                  0  // Helical ID

        cmmIxxHelOnceSetSpeed(HEL_ID, 2, cmSMODE_S, 5000, 25000, 25000);

        long nAxes[4] = {0, 1, 2, 3};
        double fCoords[4] = {5000, 5000, 3000, cmDIR_P};

        cmmIxxHelOnce(HEL_ID, 4, nAxes, fCoords, 360, NULL, TRUE);
        …
```

```
Visual Basic
'HEL_ID 는 이미 선언되어 있다고 가정함.
Dim arrayAxes(4) As Long
Dim fCoords(4) As Double

        arrayAxes(0) = 0
        arrayAxes(1) = 1
        arrayAxes(2) = 2
        arrayAxes(3) = 3
        fCoords(0) = 5000
        fCoords(1) = 5000
        fCoords(2) = 3000
        fCoords(3) = cmDIR_P

        Call cmmIxxHelOnceSetSpeed(HEL_ID, cmSMODE_S, 5000, 25000, 25000)
```

```
Call cmmIxxHelOnce(HEL_ID, 4,  arrayAxes(0), fCoords(0), 360, 0, True)
```

Delphi

```
Const HEL_ID = 0;               // Helical ID

var
        nAxes : Array[0..3] of LongInt;
        fCoords : Array[0..3] of Double;

begin
        nAxes[0] := 0;
        nAxes[1] := 1;
        nAxes[2] := 2;
        nAxes[3] := 3;

        fCoords[0] := 5000;
        fCoords[1] := 5000;
        fCoords[2] := 3000;
        fCoords[3] := cmDIR_P;

        cmmIxxHelOnce(HEL_ID, 4, @nAxes, @fCoords, 360,  NIL , cmTRUE);

end;
```

NAME

cmmIxxHelOnce
cmmIxxHelOnceStart
– 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 cmmIxxHelOnce

([in] VT_I4 HelId, [in] VT_I4 NumAxes, [in] VT_PI4 AxisList, [in] VT_PR8 CoordList, [in] VT_R8 ArcAngle, [out] VT_PR8 DistU, [in] VT_I4 IsBlocking)

❒ VT_I4 cmmIxxHelOnceStart

([in] VT_I4 HelId, [in] VT_I4 NumAxes, [in] VT_PI4 AxisList, [out] VT_PR8 CoordList, [out] VT_R8 ArcAngle, [out] VT_PR8 DistU)

DESCRIPTION

2 축 원호보간과 1 축 또는 2 축 직선보간을 동시에 시작하고 동시에 종료하는 헬리컬보간 구동을 시작합니다. cmmIxxHelOnce() 함수는 모션이 완료되기 전까지 반환되지 않으며, cmmIxxHelOnceStart() 함수는 모션을 시작시킨 후에 바로 반환됩니다.

헬리컬보간 구동에서 원호보간 이동은 그 속도와 이동량이 U 축(4 축 모션보드인 경우 Axis3, 8 축 모션보드인 경우 Axis3 또는 Axis7)과 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.

원호보간을 위한 2 축과 Z 축은 4 축 모션보드인 경우에 CH0, CH1, CH2 그리고 8 축 모션보드인 경우에 CH0, CH1, CH2 축 중에서, 혹은 CH4, CH5, CH6 축 중에서 임의로 조합하여 사용할 수 있습니다. 이 때에 CH0~2, CH4~6 이 각각의 그룹으로서 서로 교차하여 조합할 수는 없습니다

PARAMETER

▶ HelId : 이송하고자 하는 헬리컬 보간 이송 작업의 아이디를 지정합니다. CMMSDK 는 통합라이브러리이므로 여러 개의 보드가 장착되어 있는 경우에 여러 개의 헬리컬 보간 작업을 동시에 수행할 수도 있습니다. 따라서 각각의 작업을 구분지어줄 아이디가 필요합니다. cmmIxxHelOnceSetSpeed()의 헬리컬 이송 속도를 설정하는 함수도 HelId 를 입력하게 되어 있는데, 동일한 HelId 로 설정한 속도 패턴이 적용되게 됩니다.

▶ nNumAxes : 헬리컬보간에 사용되는 축 수. 이 값은 3 또는 4 이어야 합니다.

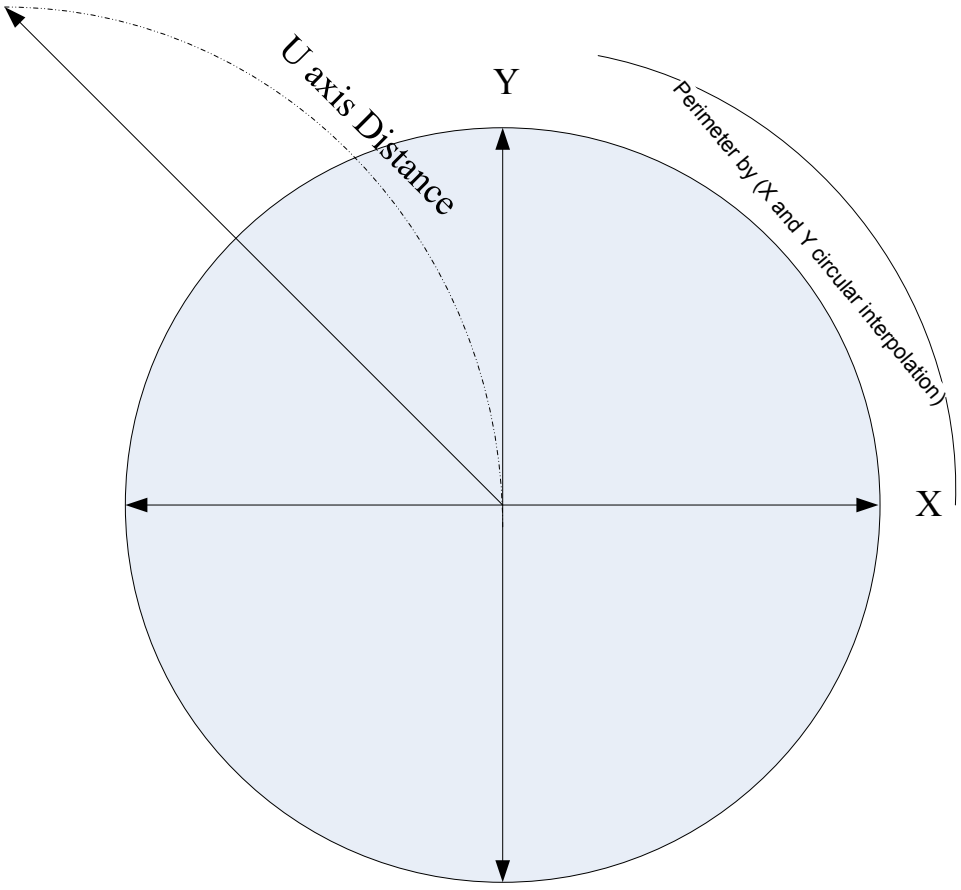
▶ nAxisList : 헬리컬보간에 사용되는 축 배열의 주소값. 이 배열의 마지막 Element 값은 4 축 모션 보드인 경우에는 반드시 3 이어야 하며 8 축 모션 보드인 경우에는 반드시 3 또는 7 이어야 합니다. 이 배열의 구성은 다음과 같이 하면 됩니다.
nAxisList[0] : 원호보간의 X 축.
nAxisList[1] : 원호보간의 Y 축.
nAxisList[2] : Z 축 (3 축만 사용하는 경우에는 U 축)
nAxisList[3] : U 축 (3 축만 사용하는 경우에는 이 매개 변수(媒介變數)는 사용안함)

▶ CoordList : 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)

□ 4 축을 사용하는 경우
nCoordList[0] : 원호 중심의 X 상대좌표값
nCoordList[1] : 원호 중심의 Y 상대좌표값
nCoordList[2] : Z 축 이동거리(상대좌표)
nCoordList[3] : 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 절대값이 360 보다 클수 없습니다.

▶ DistU : 헬리컬 보간은 X 축과 Y 축의 원호보간과 U 축의 직선 보간이 동기 되어 동작되는 보간법입니다. 이 보간법을 통해 동작하는 X 축과 Y 축의 원호 보간에 의해 이송되는 원의 둘레에 대한 거리는 U 축의 이송거리와 동일하게 됩니다. DistU 매개 변수는 U 축의 이송 거리를 반환하는 매개 변수입니다. U 축의 실제 이송한 거리를 알기 위하실 경우 유용하게 사용하실 수 있습니다.



▶ IsBlocking : 함수의 동작시에 블록킹 처리에 대한 매개변수를 요구합니다. 블록킹(Blocking) 처리는 함수의 동작시에 윈도우 메시지를 처리 여부를 결정하게 됩니다.

EXAMPLE 1

아래의 코드는 하나의 360 도 원을 그리면서 Z 축도 +3000 만큼 이송하는 헬리컬 보간을 수행하는 예입니다. 이때 Z 축 속도를 V=5000, ACC=DEC=25000 으로 설정하고, 원호보간 속도는 원주 길이와 Z 축 이동거리의 비에 따라 자동으로 설정되도록 합니다.

C/C++

```
#define HEL_ID                  0  // Helical ID

        cmmIxxHelOnceSetSpeed(HEL_ID, 2, cmSMODE_S, 5000, 25000, 25000);

        // 대상 축 설정
        long nAxes[4] = {0, 1, 2, 3};

        // 보간 이송 위치 결정
        double fCoords[4] = {5000, 5000, 3000, cmDIR_P};

        // 헬리컬 보간 이송
        cmmIxxHelOnce(HEL_ID, 4, nAxes, fCoords, 360, NULL, TRUE);
        …
```

Visual Basic

```
'HEL_ID 는 이미 선언되어 있다고 가정함.
Call cmmIxxHelOnceSetSpeed(HEL_ID,2, cmSMODE_S, 5000, 25000, 25000)
Dim nAxes(4) As Long
Dim fCoords(4) As Double

        ' 대상 축 설정
        nAxes(0) = 0
        nAxes(1) = 1
        nAxes(2) = 2
        nAxes(3) = 3

        ' 보간 이송 위치 설정
        fCoords(0) = 5000
        fCoords(1) = 5000
        fCoords(2) = 3000
        fCoords(3) = cmDIR_P

        ' 헬리컬 보간 이송
        Call cmmIxxHelOnce(HEL_ID, 4, nAxes(0), fCoords(0), 360, Null, True)
        …
```

Delphi

```
Const HEL_ID = 0;               // Helical ID

var
        nAxes : Array[0..3] of LongInt;
        fCoords : Array[0..3] of Double;

begin

        cmmIxxHelOnceSetSpeed(HEL_ID, 2, cmSMODE_S, 5000, 25000, 25000);

        // 대상 축 설정
        nAxes[0] := 0;
        nAxes[1] := 1;
        nAxes[2] := 2;
        nAxes[3] := 3;

        // 보간 이송 위치 결정
        fCoords[0] := 5000;
        fCoords[1] := 5000;
        fCoords[2] := 3000;
        fCoords[3] := cmDIR_P;
```

```
        // 헬리컬 보간 이송
        cmmIxxHelOnce(HEL_ID, 4, @nAxes, @fCoords, 360,  NIL , cmTRUE);

        …

end;
```

EXAMPLE 2: 연속적인 헬리컬 보간

□ 아래 그림 9-3 과 같이 X,Y 축에 대하여 5 회의 원호보간을 수행하면서 동시에 Z 축을 이동하면서 헬리컬보간을 수행하는 예제입니다. 이때 마지막 ARC 의 각도는 90 도로 하여 처음 시작점을 기준으로 90 도 원호를 그리고 종료하도록 합니다. 5 회의 원호보간 이송이 연속적으로 수행되도록 하기 위해서 리스트모션을 사용하였습니다.

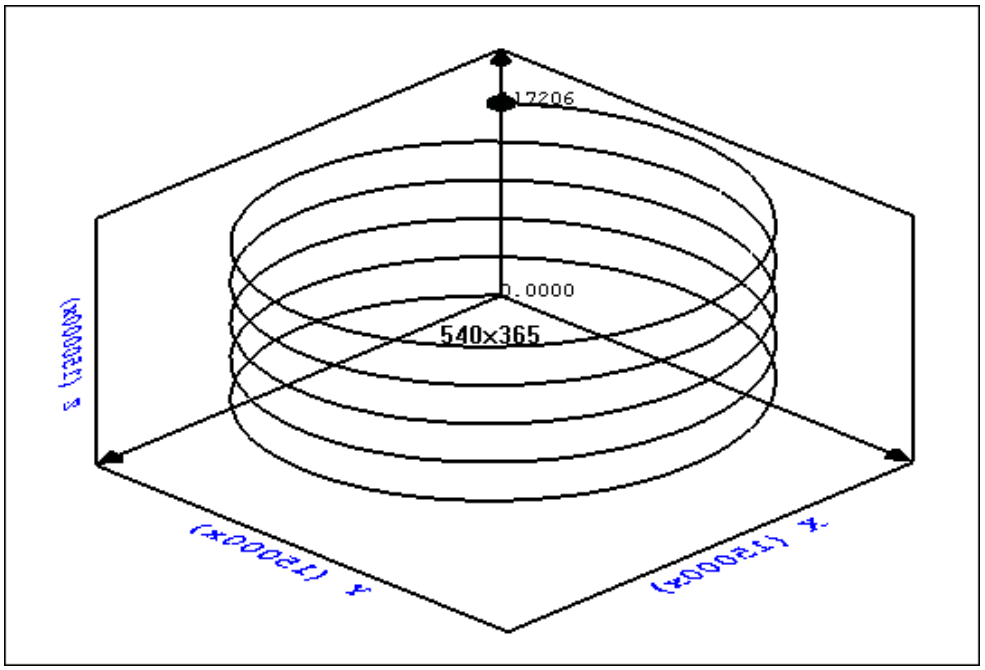


그림 9-3 헬리컬 보간

```
C/C++

#define LM_MAP  0
#define HEL_ID  0
long nAxisList[4] = {cmX1, cmY1, cmZ1, cmU1};
double fCoords[4] = {5000, 5000, 3000, cmDIR_P};

        // X1&Y1&Z1&U1 축을 리스트모션에 관계된 축으로 설정 //
        cmmLmMapAxes (LM_MAP, 0xf, 0x0);

        // 모션 리스트 등록 시작 //
        cmmLmBeginList (LM_MAP);

        // 처음 이동은 가속=25000, 감속구간 없음(0)으로 설정 //
        // 헬리컬보간 이동 :  원호중심=(5000,5000), 원호이동각도=360,    //
        cmmIxxHelOnceSetSpeed(HEL_ID, 2, cmSMODE_T, 5000, 25000, 0)

        // Z 축 이동량=3000, U 축 Direction=+방향                    //
        fCoords[0] = 5000;  fCoords[1] = 5000;
```

```
        fCoords[2] = 3000;  fCoords[3] = cmDIR_P;

        // 헬리컬 보간 이송 시작
        cmmIxxHelOnceStart(HEL_ID, 4, nAxisList, fCoords,360, NULL);

        // 두번째 부터는 Constant 속도 모드로 설정 //
        cmmIxxHelOnceSetSpeed (HEL_ID, 2, cmSMODE_C, 5000, 0, 0)
        cmmIxxHelOnceStart (HEL_ID, 4, nAxisList, fCoords,360, NULL);
        cmmIxxHelOnceStart (HEL_ID, 4, nAxisList, fCoords,360, NULL);
        cmmIxxHelOnceStart (HEL_ID, 4, nAxisList, fCoords,360, NULL);
        cmmIxxHelOnceStart (HEL_ID, 4, nAxisList, fCoords,360, NULL);

        // 모션리스트 등록을 마침 //
        cmmLmEndList (LM_MAP);

        cmmLmStartMotion (LM_MAP);// 리스트모션 수행 시작 //

        long lIsDone = FALSE;


        while(!lIsDone){

                        long nCurOperIdx=0;

                        cmmLmIsDone(LM_MAP, &lIsDone);

                        // 현재 수행되는 작업 번호를 확인(確認)하여 화면에 표시          //
                        // 아래 DisplaySequence 함수는 가상의 함수입니다. //
                        cmmLmCurSequence (LM_MAP, (long *)&nCurOperIdx);

                        DisplaySequence(nCurOperIdx);
                        Sleep(1);

                        }

}
```

```
Visual Basic
'LM_MAP 은 이미 선언되어 있다고 가정함.
Dim nAxisList(4) As Long
Dim fCoords(4) As Double

        ' 대상 축 설정
        nAxisList(0) = cmX1
        nAxisList(1) = cmY1
        nAxisList(2) = cmZ1
        nAxisList(3) = cmU1

        ' 이송 거리 설정
        fCoords(0) = 5000
        fCoords(1) = 5000
        fCoords(2) = 3000
        fCoords(3) = cmDIR_P

        '// X1&Y1&Z1&U1 축을 리스트모션에 관계된 축으로 설정 //
        Call cmmLmMapAxes(0, &HF, &H0)

        '// 모션 리스트 등록 시작 //
        Call cmmLmBeginList(0)

        '// 처음 이동은 가속=25000, 감속구간 없음(0)으로 설정 //
        Call cmmIxxHelOnceSetSpeed(0, 2, cmSMODE_T, 5000, 25000, 0)

        '// 헬리컬보간 이동 : 원호중심=(5000,5000), 원호이동각도=360,    //
        '// Z 축 이동량=3000, U 축 Direction=+방향                    //
        Call cmmIxxHelOnceStart(4, nAxisList(0), fCoords(0), 360, 0)

        '// 두번째 부터는 Constant 속도 모드로 설정 //
        Call cmmIxxHelOnceSetSpeed(0, 2, cmSMODE_C, 5000, 0, 0)
```

```
        Call cmmIxxHelOnceStart(0, 4, nAxisList(0), fCoords(0), 360, Null)
        Call cmmIxxHelOnceStart(0, 4, nAxisList(0), fCoords(0), 360, Null)
        Call cmmIxxHelOnceStart(0, 4, nAxisList(0), fCoords(0), 360, Null)
        Call cmmIxxHelOnceStart(0, 4, nAxisList(0), fCoords(0), 360, Null)
        Call cmmLmEndList(0) '// 모션리스트 등록을 마침 //
        Call cmmLmStartMotion(0)  '// 리스트모션 수행 시작 //

Dim nIsDone As Long
Dim nCurOperIdx As Long
nIsDone = False

        Do While(Not (nIsDone))

            Call cmmLmIsDone(LM_MAP, nIsDone)
            '// 현재 수행되는 작업 번호를 확인(確認)하여 화면에 표시  //
            '//아래의 DisplaySequence 함수는 가상의 함수입니다.
            Call cmmLmCurSequence(LM_MAP, nCurOperIdx)
            DisplaySequence(nCurOperIdx)

        Loop
```

```
Delphi

Const LM_MAP = 0;
Const HEL_ID = 0;               // Helical ID

var
        nAxisList : Array[0..3] of LongInt;
        fCoords : Array[0..3] of Double;
        lIsDone : LongInt;
        nCurOperIdx : LongInt;

begin
        nAxisList [0] := cmX1;
        nAxisList [1] := cmY1;
        nAxisList [2] := cmZ1;
        nAxisList [3] := cmU1;

        fCoords[0] := 5000;
        fCoords[1] := 5000;
        fCoords[2] := 3000;
        fCoords[3] := cmDIR_P;

        // X1&Y1&Z1&U1 축을 리스트모션에 관계된 축으로 설정 //
        cmmLmMapAxes (LM_MAP, $f, $0);

        // 모션 리스트 등록 시작 //
        cmmLmBeginList (LM_MAP);

        // 처음 이동은 가속=25000, 감속구간 없음(0)으로 설정 //
        // 헬리컬보간 이동 :  원호중심=(5000,5000), 원호이동각도=360,    //
        cmmIxxHelOnceSetSpeed(HEL_ID, 2, cmSMODE_T, 5000, 25000, 0);

        // Z 축 이동량=3000, U 축 Direction=+방향                    //
        fCoords[0] := 5000;
        fCoords[1] := 5000;
        fCoords[2] := 3000;
        fCoords[3] := cmDIR_P;

        // 헬리컬 보간 이송 시작
        cmmIxxHelOnceStart(HEL_ID, 4,@nAxisList, @fCoords,360, NIL);

        // 두번째 부터는 Constant 속도 모드로 설정 //
        cmmIxxHelOnceSetSpeed (HEL_ID, 2, cmSMODE_C, 5000, 0, 0);
        cmmIxxHelOnceStart (HEL_ID, 4, @nAxisList, @fCoords,360, NIL);
        cmmIxxHelOnceStart (HEL_ID, 4, @nAxisList, @fCoords,360, NIL);
        cmmIxxHelOnceStart (HEL_ID, 4, @nAxisList, @fCoords,360, NIL);
        cmmIxxHelOnceStart (HEL_ID, 4, @nAxisList, @fCoords,360, NIL);

        // 모션리스트 등록을 마침 //
```

```
        cmmLmEndList (LM_MAP);

        cmmLmStartMotion (LM_MAP);// 리스트모션 수행 시작 //

        lIsDone := cmFALSE;

        while lIsDone =0 do
        begin
                nCurOperIdx := 0;

                cmmLmIsDone(LM_MAP, @lIsDone);

                // 현재 수행되는 작업 번호를 확인(確認)하여 화면에 표시      //
                // 아래 DisplaySequence 함수는 가상의 함수입니다. //
                cmmLmCurSequence (LM_MAP,@nCurOperIdx);

                //DisplaySequence(nCurOperIdx);
                Sleep(1);

        end;


end;
```

NAME

cmmIxxSplineBuild
– 스플라인(Spline) 보간(補間)을 위한 좌표(座標) 생성

INFORMATION

Extended Interpolation Motion

VC++/VB BCB/Delphi/.NET

Level 5

☺ 위험 요소 없음

이 함수는 이송 함수가 아닙니다. 단순한 Spline 포인트를 생성하는 함수입니다.

SYNOPSIS

❒ VT_I4 cmmIxxSplineBuild

([in] VT_PR8 InArray, [in] VT_I4 NumInArray, [out] VT_PR8 OutArray, [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. 따라서 2 축뿐만 아니라 3 축, 4 축의 스플라인 보간 데이터도 생성가능합니다.

PARAMETER

▶ InArray : 스플라인 보간을 수행할 샘플 데이터 포인트 배열.

▶ NumInArray : 샘플 데이터의 수.

▶ OutArray : 스플라인 보간을 수행하여 자동 생성된 자유곡선 데이터를 전달할 배열의 주소값. 이 배열의 크기는 NumOutPoints 보다 크거나 같아야 합니다.

▶ NumOutArray : 스플라인 보간을 수행하여 자동 생성할 데이터 수. 이 값은 전체 곡선을 몇 개의 데이터 포인트로 곡선화할 것인지를 결정합니다.

REFERENCE

□ 이 함수는 스플라인 보간 기법을 이용하여 데이터를 생성해주는 역할만 하며 실제로 모션을 구동하지는 않습니다. 스플라인 보간 구동을 하기 위해서는 여기서 생성된 각 데이터 포인트를 IxLineTo() 함수 등을 이용하여 구동해주어야 합니다. 이 때 리스트모션(Listed Motion) 기법과 함께 사용하면 효과적으로 구동할 수 있습니다.

9.4 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 Move나 MoveTo 함수와 같은 In-Position 함수를 사용할 때에도 작업 속도의 연속성을 확보할 수 있습니다.

▣ 리스트 모션의 적용 예 1

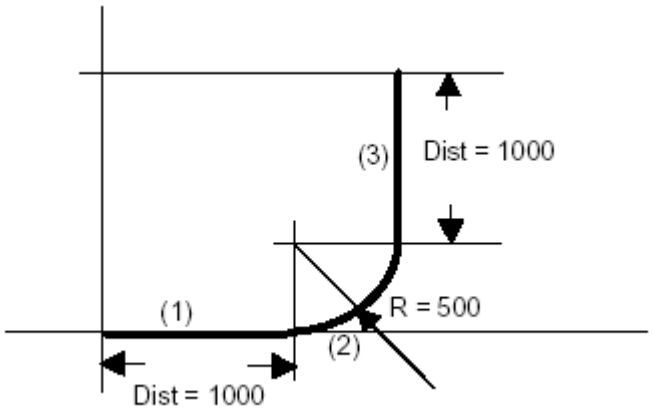


그림 9-4 리스트 모션의 적용예

그림 9-4 는 3 회의 Coordinated Motion으로 이루어지는 작업의 예 입니다. 그림과 같이 3 회의 Coordinated Motion을 연속적으로 수행하고자할 때 다음과 같이 리스트 모션을 적용하면 각 작업간의 Delay가 최소화되어 연속적인 작업을 수행할 수 있습니다.

```
//LMAP0 은 이미 선언되어 있다고 가정함.
//MAP0 은 이미 선언되어 있다고 가정함.

double fDistList[2];

// 리스트 모션 대상 축 맵 설정
cmmLmMapAxes(LMAP0, 0x3, 0x0);

// 모션 리스트 등록 시작 //
cmmLmBeginList(LMAP0);

  // X1&Y1 축을 인터폴레이션 축으로 맵핑//
  cmmIxMapAxes(MAP0, 0x3, 0x0);

  // Set Speed Pattern of interpolation //
  cmmIxSetSpeedPattern(MAP0, cmFALSE,  cmSMODE_S, 1000, 5000, 5000);

  // (1000,0)만큼 직선 보간 이동 //
  fDistList[0]=1000;
  fDistList[1]=0;

  cmmIxLine(MAP0, fDistList, cmFALSE);

  // 중심점 Offset = (0, 500), End Angle = 90 DEG  의 조건으로 원호보간 이동
  cmmIxArcA(MAP0, 0, 500, 90, cmFALSE);

  // (0,1000)만큼 직선 보간 이동 //
  fDistList[0]=0;
  fDistList[1]=1000;
```

```
   cmmIxLine(MAP0, fDistList, cmFALSE);

// 모션 리스트 등록을 완료 //
cmmLmEndList(LMAP0);

// 리스트 모션 수행 //
cmmLmStartMotion(LMAP0);

// 리스트 모션이 모두 완료될 때까지 기다림 //
while(cmTRUE)
{
        LONG IsDone;
        cmmLmIsDone(LMAP0, &IsDone)

        if ( IsDone == cmTRUE ) break;
        else continue;
}
```

▣ 리스트 모션의 적용 예 2

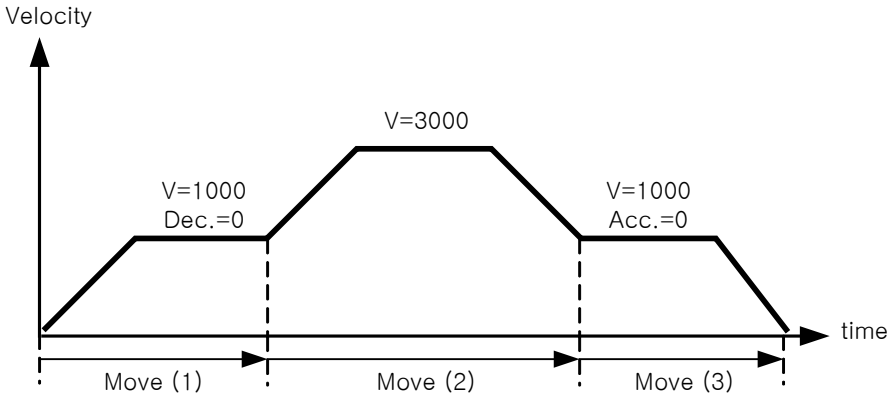


그림 9-5 리스트 모션의 적용 예

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

표 10 속도의 연속성을 가지는 연속적인 In-Position 모션

리스트 모션을 이용하면 표 10 과 같이 In-Position의 경우에도 작업 속도의 연속성을 확보할 수 있습니다. 다음의 코드는 표 10 의 작업을 리스트 모션을 적용하여 구현하는 예입니다.

```
//LMAP0 은 이미 선언되어 있다고 가정함.
cmmLmBeginList(LMAP0); // 모션 리스트 등록 시작 //

// Move (1) //
cmmCfgSetSpeedPattern(cmX1, cmSMODE_T, 1000, 2000, 0);
cmmSxSetSpeedRatio(cmX1, cmSMODE_T, 100, 100, 100);
cmmSxMoveTo(cmX1, 3000,cmFALSE);

// Move (2) //
cmmCfgSetSpeedPattern(cmX1, cmSMODE_T, 3000, 2000, 2000);
cmmSxSetSpeedRatio(cmX1, cmSMODE_T, 100, 100, 100);
cmmSxMoveTo(cmX1, 8000,cmFALSE);

// Move (3) //
cmmCfgSetSpeedPattern(cmX1, cmSMODE_T, 1000, 0, 2000);
```

```
    cmmSxSetSpeedRatio(cmX1, cmSMODE_T, 100, 100, 100);
    cmmSxMoveTo(cmX1, 11000,cmFALSE);

    cmmLmEndList(LMAP0); // 모션 리스트 등록을 마침 //
    cmmLmStartMotion(LMAP0); // 리스트 모션 수행 //

    // 리스트 모션이 모두 완료될 때까지 기다림 //
    while(cmTRUE)
    {
            LONG IsDone;
            cmmLmIsDone(LMAP0, &IsDone)

            if ( IsDone == cmTRUE ) break;
            else continue;
     }
}
```

9.4.1 함수 요약

리스트 모션에 관련된 함수들은 다음과 같습니다.

Summary of Functions
❒ VT_I4 cmmLmMapAxes ([in] VT_I4 LmIndex, [in] VT_I4 MapMask1, [in] VT_I4 MapMask2) Listed Motion 에서 사용되는 모든 축들을 지정한 Index 로 등록합니다. 32Bit 형 전달 인자 2 개를 통해 최대 64 개의 모션 채널을 대상(對象)으로 리스트 모션에 사용될 수 있습니다.
❒ VT_I4 cmmLmBeginList ([in] VT_I4 LmIndex) Listed Motion 에서 사용되는 함수의 등록(登錄)을 시작합니다. 이 함수가 시작 된 이후(以後)로는 리스트 모션을 지원하는 함수는 cmmLmEndList 함수를 사용할 때까지 리스트 모션 대상(對象) 함수로서의 자격으로 추가(追加)됩니다.
❒ VT_I4 cmmLmEndList ([in] VT_I4 LmIndex) Listed Motion 에서 사용되는 함수의 등록을 종료(終了)합니다. cmmLmBeginList 함수와 cmmLmEndList 함수 사이에 있는 존재하는 리스트 모션을 지원하는 함수의 등록을 명시적으로 완료(完了)하게 됩니다.
❒ VT_I4 cmmLmIsDone ([in] VT_I4 LmIndex, [out] VT_PI4 IsDone) Listed Motion 의 동작의 완료(完了) 상태를 확인(確認)합니다.
❒ VT_I4 cmmLmWaitDone ([in] VT_I4 LmIndex, [in] VT_I4 IsBlocking) Listed Motion 의 동작의 완료(完了)를 위해 대기(待機) 합니다.
❒ VT_I4 cmmLmCurSequence ([in] VT_I4 LmIndex, [out] VT_PI4 SeqIndex) Listed Motion 에서 등록(登錄)된 작업 중에서 현재 수행되고 있는 작업의 번호를 반환합니다.
❒ VT_I4 cmmLmImmediacySet ([in] VT_I4 LmIndex) Listed Motion 의 추후(追後)의 추가(追加) 기능을 위해 예약(豫約)된 함수입니다.
❒ VT_I4 cmmLmStartMotion ([in] VT_I4 LmIndex) Listed Motion 에서 등록(登錄)된 작업을 대상으로 실제 Listed Motion 을 수행(遂行)합니다.
❒ VT_I4 cmmLmAbortMotion ([in] VT_I4 LmIndex) Listed Motion 을 수행(遂行)하고 있는 상황에서 현재 실행중인 Listed Motion 을 중단(中斷)합니다.
❒ VT_I4 cmmLmDoPutOne ([in] VT_I4 LmIndex, [in] VT_HANDLE hDoDevice, [in] VT_I4 Channel, [in] VT_I4 OutState) Listed Motion 에서 단일(單一) 채널을 대상으로 디지털 출력(出力) 명령을 추가합니다. 실제 Listed Motion 수행(遂行)시에 한 개의 채널에 대해서 지정한 디지털 출력(出力) 명령을 수행(遂行)합니다. 이 함수에서 사용되는 출력(出力) 채널은 모션 보드 상의 지역(地域) 출력(出力) 채널입니다.
❒ VT_I4 cmmLmDoPutMulti ([in] VT_I4 LmIndex, [in] VT_HANDLE hDoDevice, [in] VT_I4 ChannelGroup, [in] VT_I4 Mask, [in] VT_I4 OutStates) Listed Motion 에서 다중(多重) 채널을 대상으로 디지털 출력 명령을 추가합니다. 실제 Listed Motion 수행(遂行)시에 다수의 채널에 대해서 지정한 디지털 출력 명령을 수행합니다. 이 함수에서 사용되는 출력 채널은 모션 보드 상의 지역(地域) 출력 채널입니다.

❒ VT_I4 cmmLmDoPulseOne ([in] VT_I4 LmIndex, [in] VT_HANDLE hDoDevice, [in] VT_I4 Channel, [in] VT_I4 OutState, [in] VT_I4 Duration)
Listed Motion 에서 단일(單一) 채널을 대상으로 디지털 펄스 신호 출력(出力) 명령을 추가합니다. 실제 Listed Motion 수행(遂行)시에 단일의 채널에 대해서 지정한 디지털 펄스 신호 출력(出力) 명령을 수행합니다. 이 함수에서 사용되는 출력(出力) 채널은 모션 보드 상의 지역(地域) 출력 채널입니다.

❒ VT_I4 cmmLmDoPulseMulti ([in] VT_I4 LmIndex, [in] VT_HANDLE hDoDevice, [in] VT_I4 ChannelGroup, [in] VT_I4 Mask, [in] VT_I4 OutStates, [in] VT_I4 Duration)
Listed Motion 에서 다중(多重) 채널을 대상으로 디지털 펄스 신호 출력(出力) 명령을 추가합니다. 실제 Listed Motion 수행시에 다중의 채널에 대해서 지정한 디지털 펄스 신호 출력(出力) 명령을 수행합니다. 이 함수에서 사용되는 출력(出力) 채널은 모션 보드 상의 지역(地域) 출력 채널입니다.

❒ VT_I4 cmmLmxStart ([in] VT_I4 LmIndex,[in] VT_I4 AxisMask1,[in] VT_I4 AxisMask2)
Extend Listed Motion 에서 등록(登錄)된 작업을 대상으로 실제 Listed Motion 을 수행(遂行)합니다.

❒ VT_I4 cmmLmxPause ([in] VT_I4 LmIndex)
Extend Listed Motion 수행 중 명령 시퀀스를 일시 정지하고 싶을 때 사용되는 함수입니다.

❒ VT_I4 cmmLmxResume ([in] VT_I4 LmIndex, [in] VT_I4 IsClearQue)
Extend Listed Motion 수행 중 cmmLmxPause() 명령에 의해 일시 정지된 명령 시퀀스를 재개하고자 할 때 사용합니다.

❒ VT_I4 cmmLmxEnd ([in] VT_I4 LmIndex)
Extend Listed Motion 작업을 종료 합니다.

❒ VT_I4 cmmLmxSetSeqMode ([in] VT_I4 LmIndex, [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 cmmLmxGetSeqMode ([in] VT_I4 LmIndex, [in] 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 cmmLmxSetNextItemId ([in] VT_I4 LmIndex, [in] VT_I4 SeqId)
Extend Listed Motion 에서 수행할 명령(Item)에 대해 Sequence Item Id 를 설정합니다.

❒ VT_I4 cmmLmxGetNextItemId ([in] VT_I4 LmIndex, [in] VT_PI4 SeqId)
Extend Listed Motion 에서 다음 수행할 명령(Item)에 해당하는 Sequence Item Id 를 반환합니다.

❒ VT_I4 cmmLmxSetNextItemParam ([in] VT_I4 LmIndex, [in] VT_I4 ParamIdx, [in] VT_I4 ParamData)
Extend Listed Motion 에서 다음 수행 예정인 명령에 대한 함수 파라미터 설정 값 설정합니다.

❒ VT_I4 cmmLmxGetNextItemParam ([in] VT_I4 LmIndex, [in] VT_I4 ParamIdx, [out] VT_PI4 ParamData)
Extend Listed Motion 에서 다음 수행 예정인 명령에 대한 함수 파라미터 설정 값 반환합니다.

❒ VT_I4 cmmLmxGetRunItemParam ([in] VT_I4 LmIndex, [in] VT_I4 ParamIdx, [out] VT_PI4 ParamData)
Extend Listed Motion 에서 현재 수행 중인 명령에 대한 함수 파라미터 설정 값 반환합니다.

❒ VT_I4 cmmLmxGetRunItemStaPos([in] VT_I4 LmIndex,[in] VT_I4 Axis, [out] VT_PR8 Position)
Extend Listed Motion 수행 중에 현재 수행 중인 명령(Current Sequence Item Id) 이 수행되기 직전에 해당 축의 Command Pulse Position 값을 반환 합니다.

❒ VT_I4 cmmLmxGetRunItemTargPos([in] VT_I4 LmIndex, [in] VT_I4 Axis, [out] VT_PR8 Position)
Extend Listed Motion 수행 중에 현재 수행 중인 명령(Current Sequence Item Id) 에 대해 해당 축의 목표 좌표에 해당하는 Command Pulse Position 값을 반환 합니다.

❒ VT_I4 cmmLmxGetSts([in] VT_I4 LmIndex,[in] VT_I4 LmxStsId, [out] VT_PI4 LmxStsVal)
Extend Listed Motion 에서 현재 수행 중인 Sequence 명령에 대한 상태 값을 Status Id 값을 통해 확인이 가능한 함수 입니다.

9.4.2 함수 설명

NAME

cmmLmMapAxes
- Listed Motion 대상(對象) 축 그룹(Group) 설정(設定)

INFORMATION

Listed Motion
VC++/VB
BCB/Delphi/.NET
Level 6
위험 요소 없음

SYNOPSIS

❒ VT_I4 cmmLmMapAxes ([in] VT_I4 LmIndex, [in] VT_I4 MapMask1, [in] VT_I4 MapMask2)

DESCRIPTION

이 함수는 리스트모션에서 사용되는 모든 축들을 등록하는 함수입니다. 이는 리스트모션을 수행하면서 독립적으로 다른 축들을 구동하고자할 때 리스트모션 작업이 독립적으로 구동되는 축에 영향을 주지 않도록 하기 위함입니다. 예를 들어, 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.

cmmLmBeginList() 함수를 호출하기 전에 먼저 cmmLmMapAxes(MAP_INDEX, 0x3, 0) 과 같이 리스트 모션에서는 X,Y 축만 영향을 주도록 하면 Z 축은 오류없이 리스트모션과 독립적으로 구동할 수 있습니다.

PARAMETER

▶ LmIndex : 리스트모션의 Map Index 를 의미합니다. CMMSDK2 라이브러리는 동일 PC 에 장착된 모든 보드의 축들을 통합관리하는 통합라이브러리이므로 장치의 수만큼의 리스트모션 작업이 각각 동시에 수행될 수 있습니다. 그러므로 이들을 서로 구분해줄 인자가 필요한데, LmIndex 가 바로 그 역할을 하는 인자입니다. LmIndex 값은 0 ~ 15 의 수를 사용할 수 있습니다.

▶ MapMask1 : 리스트모션에 포함시킬 축에 대한 하위 32 비트 Mask 값입니다. 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.

▶ MapMask 2: 리스트모션에 포함시킬 축에 대한 상위 32 비트 Mask 값입니다. 이 값의 bit0 ~ bit31 은 각각 Axis32 ~ Axis63 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
cmERR_NONE	수행 성공

REFERENCE

□ CMMSDK2 라이브러리는 동일 PC 에 장착된 모든 보드의 축들을 통합관리하는 통합라이브러리이므로 장치의 수만큼의 리스트모션 작업이 동시에 수행될 수 있습니다. 사용자는 LmIndex 매개 변수(媒介變數)를 사용하여 각각의 리스트모션 작업을 구분합니다. 모든 리스트모션 관련 함수는 LmIndex 를 매개 변수(媒介變數)로 취하고 있습니다.

□ 동일한 LmIndex 를 인자로 하여 cmmLmMapAxes() 함수에 의해서 맵핑되는 축들은 모두 동일한 장치내에 속한 축들이어야 합니다. 왜냐하면 리스트모션은 장치의 하드웨어 인터럽트(Interrupt)를 사용하므로 서로 다른 장치에 있는 축들의 리스트모션은 지원할 수 없습니다. 또한 동일한 이유로 하나의 장치에 대해서 하나의 리스트 모션만이 실행될 수 있습니다.

□ 동일 LmIndex 를 인자로 하여 MapMask1 과 MapMask2 에 의해서 리스트모션에 포함된 축들의 이전 이송 작업은 cmmLmStartMotion() 함수가 실행되면서 정지하게 됩니다.

□ 이 함수가 맵마스크를 지정한다고 해서 리스트모션이 꼭 해당 맵마스크를 이용하는 보간구동만 지원하는 것은 아닙니다. 맵마스크에 포함된 축의 단축 구동, 다축 구동, 보간 구동을 모두 사용할 수 있습니다. 여기서 맵마스크를 지정하는 것은 단지 리스트모션이 영향을 주지 말아야하는 축을 알기 위해서입니다.

NAME

cmmLmBeginList
- Listed Motion 대상(對象) 함수 추가(追加) 시작

INFORMATION

Listed Motion

VC++/VB

BCB/Delphi/.NET

Level 6

다소 주의

Listed Motion 에서 Begin List 는 End List 와 서로 짝을 이룹니다.

SYNOPSIS

❒ VT_I4 cmmLmBeginList ([in] VT_I4 LmIndex)

DESCRIPTION

이 함수는 List Motion 에서 수행할 작업들의 등록을 시작합니다. 이 함수를 호출한 후 cmmLmEndList() 함수가 호출되기 전까지 수행되는 모든 모션작업은 실제 모션을 수행하는 것이 아니라 리스트모션으로 등록만 되는 것이며, cmmLmStartMotion() 함수가 호출될 때 비로소 등록된 모션이 실제로 수행됩니다.

PARAMETER

▶ LmIndex : 리스트모션의 Map Index 를 의미합니다.

RETURN VALUE

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

REFERENCE

cmmLmEndList

SEE ALSO

□ 리스트모션으로 등록가능한 함수들의 리스트는 아래 표와 같습니다.

리스트모션으로 등록가능한 함수	인덱스 부여
VT_I4 cmmCfgSetSpeedPattern ([in] VT_I4 Channel, [in] VT_I4 SpeedMode, [in] VT_R8 WorkSpeed, [in] VT_R8 Accel, [in] VT_R8 Decel)	NO
VT_I4 cmmSxOptSetIniSpeed ([in] VT_I4 Channel, [in] VT_R8 IniSpeed)	NO
VT_I4 cmmSxMoveStart ([in] VT_I4 Channel, [in] VT_R8 Distance)	YES
VT_I4 cmmSxMove ([in] VT_I4 Channel, [in] VT_R8 Distance, [in] VT_I4 IsBlocking)	YES
VT_I4 cmmSxMoveToStart ([in] VT_I4 Channel, [in] VT_R8 Position)	YES
VT_I4 cmmSxMoveTo ([in] VT_I4 Channel, [in] VT_R8 Position, [in] VT_I4 IsBlocking)	YES
VT_I4 cmmIxMapAxes ([in] VT_I4 MapIndex, [in] VT_I4 MapMask)	NO
VT_I4 cmmIxSetSpeedPattern ([in] VT_I4 MapIndex, [in] VT_I4 IsVectorSpeed, [in] VT_I4 SpeedMode, [in] VT_R8 Vel, [in] VT_R8 Acc, [in] VT_R8 Dec)	NO
VT_I4 cmmIxLine ([in] VT_I4 MapIndex, [in] VT_PR8 DistList, [in] VT_I4 IsBlocking)	YES
VT_I4 cmmIxLineStart ([in] VT_I4 MapIndex, [in] VT_PR8 DistList)	YES
VT_I4 cmmIxLineTo ([in] VT_I4 MapIndex, [in] VT_PR8 PostList, VT_I4 IsBlocking)	YES
VT_I4 cmmIxLineToStart ([in] VT_I4 MapIndex, [in] VT_PR8 PostList)	YES

리스트모션으로 등록가능한 함수	인덱스 부여
VT_I4 cmmIxArcA ([in] VT_I4 MapIndex, [in] VT_R8 XCentOffset, [in] VT_R8 YCentOffset, [in] VT_R8 EndAngle, [in] VT_I4 IsBlocking)	YES
VT_I4 cmmIxArcAStart ([in] VT_I4 MapIndex, [in] VT_R8 XcentOffset, [in] VT_R8 YCentOffset, [in] VT_R8 EndAngle)	YES
VT_I4 cmmIxArcATo ([in] VT_I4 MapIndex, [in] VT_R8 XCent, [in] VT_R8 YCent, [in] VT_R8 EndAngle, [in] VT_I4 IsBlocking)	YES
VT_I4 cmmIxArcAToStart ([in] VT_I4 MapIndex, [in] VT_R8 XCent, [in] VT_R8 YCent, [in] VT_R8 EndAngle)	YES
VT_I4 cmmIxArcP ([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)	YES
VT_I4 cmmIxArcPStart ([in] VT_I4 MapIndex, [in] VT_R8 XCentOffset, [in] VT_R8 YCentOffset, [in] VT_R8 XEndPointDist, [in] VT_R8 YEndPointDist, [in] VT_I4 Direction)	YES
VT_I4 cmmIxArcPTo ([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)	YES
VT_I4 cmmIxArcPToStart ([in] VT_I4 MapIndex, [in] VT_R8 XCent, [in] VT_R8 YCent, [in] VT_R8 XEndPos, [in] VT_R8 YEndPos, [in] VT_I4 Direction)	YES
VT_I4 cmmIxxHelOnceSetSpeed ([in] VT_I4 HelId, [in] VT_I4 Master, [in] VT_I4 SpeedMode, [in] VT_R8 WorkSpeed, [in] VT_R8 Acc, [in] VT_R8 Dec)	NO
VT_I4 cmmIxxHelOnce ([in] VT_I4 HelId, [in] VT_I4 NumAxes, [in] VT_PI4 AxisList, [in] VT_PR8 CoordList, [in] VT_R8 ArcAngle, [out] VT_PR8 DistU, [in] VT_I4 IsBlocking)	YES
VT_I4 cmmIxxHelOnceStart ([in] VT_I4 NumAxes, [in] VT_PI4 AxisList, [in] VT_PR8 CoordList, [in] VT_R8 ArcAngle, [out] VT_PR8 DistU)	YES

리스트 모션에서 다축 모션 구동에 대한 안내

CMMSDK 에서는 리스트모션에서 MxMove 를 지원하지 않습니다. 따라서 다축 구동시에는 보간제어를 사용하십시요

NAME

cmmLmEndList
- Listed Motion 대상 함수 추가(追加) 종료

INFORMATION

Listed Motion

VC++/VB

BCB/Delphi/.NET

Level 6

다소 주의

Listed Motion 에서 Begin List 는 End List 와 서로 짝을 이룹니다.

SYNOPSIS

❒ VT_I4 cmmLmEndList ([in] VT_I4 LmIndex)

DESCRIPTION

이 함수는 List Motion 에서 수행할 작업들의 등록을 종료합니다.

PARAMETER

▶ LmIndex : 리스트모션의 Map Index 를 의미합니다.

RETURN VALUE

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

SEE ALSO

cmmLmBeginList

NAME

cmmLmIsDone
- Listed Motion 수행 완료확인(原點確認)

INFORMATION

Listed Motion
VC++/VB
BCB/Delphi/.NET
Level 6
위험 요소 없음

SYNOPSIS

❒ VT_I4 cmmLmIsDone

([in] VT_I4 LmIndex, [out] VT_PI4 IsDone)

DESCRIPTION

이 함수는 리스트 모션 수행이 완료를 확인하기 위한 용도로 사용되는 함수입니다.

PARAMETER

▶ LmIndex : 리스트모션의 Map Index 를 의미합니다.

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

Value	Meaning
0	리스트 모션이 완료되지 않음
1	등록된 모든 리스트 모션이 완료됨

RETURN VALUE

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

NAME

cmmLmWaitDone
- Listed Motion 수행 완료대기(原點復歸)

INFORMATION

Listed Motion
VC++/VB
BCB/Delphi/.NET
Level 6
위험 요소 없음

SYNOPSIS

❒ VT_I4 cmmLmWaitDone

([in] VT_I4 LmIndex, [in] VT_I4 IsBlocking)

DESCRIPTION

이 함수는 리스트 모션 수행이 완료(完了)시까지 대기하는 함수입니다.

PARAMETER

▶ LmIndex : 리스트모션의 Map Index 를 의미합니다.

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

Value	Meaning
cmFALSE	블록(Blocking)을 하지 않습니다. 따라서 해당 모션이 완료되는 동안에도 윈도우 이벤트를 처리합니다.
cmTRUE	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
cmERR_NONE	수행 성공

NAME

cmmLmCurSequence
– 현재 실행중인 리스트 모션 명령 번호 반환(返還)

INFORMATION

- Listed Motion
- VC++/VB
- BCB/Delphi/.NET
- Level 6
- 위험 요소 없음

SYNOPSIS

❒ VT_I4 cmmLmCurSequence

([in] VT_I4 LmIndex, [out] VT_PI4 SeqIndex)

DESCRIPTION

이 함수는 실행 중인 리스트 모션 명령어의 번호를 반환합니다. 모션 명령어의 번호라는 의미는 cmmLmStartMotion 가 실행되기 전에 Listed 모션을 수행할 명령어를 등록시작하는 cmmLmBeginList 명령어가 실행된 이후 cmmLmEndList 의 함수가 실행되기 전까지에 등록된 함수에 대한 번호를 의미합니다. 실행되어야 할 Listed 모션 명령어가 총 10 개라면, cmmLmCurSequence 는 10 개의 명령어 중에 현재 실행중인 모션 명령어가 어떤 명령어 인지를 시퀀스 인덱스(Sequence Index) 의 의미로 반환하게 됩니다.

일반적으로 Listed 모션이 실행 중에는 해당 명령 자체가 윈도우 운영체제의 사용자 메모리 공간(User Memory Space)이 아닌 운영체제 메모리 공간(Kernal Memory Space) 에서 수행되며, 현재 실행 중인 운영체제(Kernel Memory Space) 메모리 공간의 어떤 모션 명령어가 실행되는 지를 이 함수를 통해 알 수 있습니다.

아래의 표에 리스트되어 있는 함수들은 리스트모션에 포함되었을 때 시퀀스 번호가 부여되는 함수들입니다. 아래의 표에 포함되지 않은 다른 함수들은 시퀀스 번호에 영향을 미치지 않습니다.

리스트모션으로 등록가능한 함수	인덱스 부여
VT_I4 cmmCfgSetSpeedPattern ([in] VT_I4 Channel, [in] VT_I4 SpeedMode, [in] VT_R8 WorkSpeed, [in] VT_R8 Accel, [in] VT_R8 Decel)	NO
VT_I4 cmmSxOptSetIniSpeed ([in] VT_I4 Channel, [in] VT_R8 IniSpeed)	NO
VT_I4 cmmSxMoveStart ([in] VT_I4 Channel, [in] VT_R8 Distance)	YES
VT_I4 cmmSxMove ([in] VT_I4 Channel, [in] VT_R8 Distance, [in] VT_I4 IsBlocking)	YES
VT_I4 cmmSxMoveToStart ([in] VT_I4 Channel, [in] VT_R8 Position)	YES
VT_I4 cmmSxMoveTo ([in] VT_I4 Channel, [in] VT_R8 Position, [in] VT_I4 IsBlocking)	YES
VT_I4 cmmIxMapAxes ([in] VT_I4 MapIndex, [in] VT_I4 MapMask)	NO
VT_I4 cmmIxSetSpeedPattern ([in] VT_I4 MapIndex, [in] VT_I4 IsVectorSpeed, [in] VT_I4 SpeedMode, [in] VT_R8 Vel, [in] VT_R8 Acc, [in] VT_R8 Dec)	NO
VT_I4 cmmIxLine ([in] VT_I4 MapIndex, [in] VT_PR8 DistList, [in] VT_I4 IsBlocking)	YES
VT_I4 cmmIxLineStart ([in] VT_I4 MapIndex, [in] VT_PR8 DistList)	YES
VT_I4 cmmIxLineTo ([in] VT_I4 MapIndex, [in] VT_PR8 PostList, VT_I4 IsBlocking)	YES
VT_I4 cmmIxLineToStart ([in] VT_I4 MapIndex, [in] VT_PR8 PostList)	YES
VT_I4 cmmIxArcA ([in] VT_I4 MapIndex, [in] VT_R8 XCentOffset, [in] VT_R8 YCentOffset, [in] VT_R8 EndAngle, [in] VT_I4 IsBlocking)	YES
VT_I4 cmmIxArcAStart ([in] VT_I4 MapIndex, [in] VT_R8 XcentOffset, [in] VT_R8 YCentOffset, [in] VT_R8 EndAngle)	YES
VT_I4 cmmIxArcATo ([in] VT_I4 MapIndex, [in] VT_R8 XCent, [in] VT_R8 YCent, [in] VT_R8 EndAngle, [in] VT_I4 IsBlocking)	YES
VT_I4 cmmIxArcAToStart ([in] VT_I4 MapIndex, [in] VT_R8 XCent, [in] VT_R8 YCent, [in] VT_R8 EndAngle)	YES
VT_I4 cmmIxArcP ([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)	YES
VT_I4 cmmIxArcPStart ([in] VT_I4 MapIndex, [in] VT_R8 XCentOffset, [in] VT_R8 YCentOffset, [in] VT_R8 XEndPointDist, [in] VT_R8 YEndPointDist, [in] VT_I4 Direction)	YES
VT_I4 cmmIxArcPTo ([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)	YES

리스트모션으로 등록가능한 함수	인덱스 부여
VT_I4 cmmIxArcPToStart ([in] VT_I4 MapIndex, [in] VT_R8 XCent, [in] VT_R8 YCent, [in] VT_R8 XEndPos, [in] VT_R8 YEndPos, [in] VT_I4 Direction)	YES
VT_I4 cmmIxxHelOnceSetSpeed ([in] VT_I4 HelId, [in] VT_I4 Master, [in] VT_I4 SpeedMode, [in] VT_R8 WorkSpeed, [in] VT_R8 Acc, [in] VT_R8 Dec)	NO
VT_I4 cmmIxxHelOnce ([in] VT_I4 HelId, [in] VT_I4 NumAxes, [in] VT_PI4 AxisList, [in] VT_PR8 CoordList, [in] VT_R8 ArcAngle, [out] VT_PR8 DistU, [in] VT_I4 IsBlocking)	YES
VT_I4 cmmIxxHelOnceStart ([in] VT_I4 NumAxes, [in] VT_PI4 AxisList, [in] VT_PR8 CoordList, [in] VT_R8 ArcAngle, [out] VT_PR8 DistU)	YES

표 11 리스트 모션 포함시 시퀀스 번호가 부여되는 함수

PARAMETER

▶ LmIndex : 리스트모션의 Map Index 를 의미합니다.

▶ SeqIndex : 현재 실행 중인 Listed 모션 명령어의 번호를 반환합니다.

RETURN VALUE

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

NAME

cmmLmImmediacySet
- 리스트 모션 특수 기능 설정(設定)

INFORMATION

Listed Motion

VC++/VB

BCB/Delphi/.NET

Level 6

위험 요소 없음

SYNOPSIS

❒ VT_I4 cmmLmImmediacySet

([in] VT_I4 LmIndex)

DESCRIPTION

이 함수는 현재 버전의 라이브러리에서는 사용할 수 없으며, 향후를 위해서 예약된 함수 기능입니다.

PARAMETER

▶ LmIndex : 리스트모션의 Map Index 를 의미합니다.

RETURN VALUE

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

NAME

cmmLmStartMotion
- Listed Motion 시작(始作)

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 cmmLmStartMotion ([in] VT_I4 Lmindex)

DESCRIPTION

이 함수는 List Motion 으로 등록된 작업들에 대하여 실제로 모션을 시작합니다.

PARAMETER

▶ LmIndex : 리스트모션의 Map Index 를 의미합니다.

RETURN VALUE

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

SEE ALSO

cmmLmAbortMotion

NAME

cmmLmAbortMotion
- Listed Motion 중단(中斷)

INFORMATION

Listed Motion

VC++/VB

BCB/Delphi/.NET

Level 6

☺ 모션 중단 함수

리스트 모션이 수행 도중 정지(停止)하게 됩니다. 작업 안전사항을 필히 확인(確認)하십시요.

SYNOPSIS

❒ VT_I4 cmmLmAbortMotion ([in] VT_I4 LmIndex)

DESCRIPTION

이 함수는 cmmLmStartMotion() 을 이용하여 리스트 모션을 수행하고 있는 중에 리스트 모션을 중지하고자 할 때 사용합니다. 리스트 모션으로 등록된 작업들의 수행이 모두 완료되면 자동으로 리스트모션이 종료됩니다. cmmLmAbortMotion() 함수는 리스트 모션이 진행되고 있는 도중에 리스트 모션을 중지해야할 때 사용합니다.

PARAMETER

▶ LmIndex : 리스트모션의 Map Index 를 의미합니다.

SEE ALSO

cmmLmStartMotion

RETURN VALUE

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

NAME

cmmLmAbortMotionEx
Listed Motion 중단(中斷)

INFORMATION

Listed Motion

VC++/VB

BCB/Delphi/.NET

Level 6

☺ 모션 중단 함수

리스트 모션이 수행 도중 정지(停止)하게 됩니다. 작업 안전사항을 필히 확인(確認)하십시요.

SYNOPSIS

❒ VT_I4 cmmLmAbortMotionEx([in] VT_I4 LmIndex, [in] VT_R8 DecelT_sec)

DESCRIPTION

이 함수는 cmmLmStartMotion() 을 이용하여 리스트 모션을 수행하고 있는 중에 리스트 모션을 중지하고자 할 때 사용합니다. 리스트 모션으로 등록된 작업들의 수행이 모두 완료되면 자동으로 리스트모션이 종료됩니다. cmmLmAbortMotion() 함수는 리스트 모션이 진행되고 있는 도중에 리스트 모션을 중지해야할 때 사용합니다 cmmLmAbortMotion() 함수와의 차이점은, Listed Motion 중지 시 감속 시간을 초(sec) 단위로 설정 가능하다는 것입니다.

PARAMETER

▶ LmIndex : 리스트모션의 Map Index 를 의미합니다.

▶ DecelT_sec : 리스트 모션 중지 시 감속 시간을 초(sec) 단위로 설정합니다.

SEE ALSO

cmmLmAbortMotion

RETURN VALUE

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

EXAMPLE

C/C++

```
/*********************************************************************
* OnBtnStop() : "Stop" Button's event handler
* : This button stops motion.
*********************************************************************/
#define              DEC_TIME_SEC      2 // Listed Motion 중지 시 2 초의 감속 시간 후에 정지 시키도록 한다.

void CListedMotionDlg::OnBtnStop()
{
        cmmLmAbortMotionEx(LM_MAP_IDX, DEC_TIME_SEC);
}
```

```
/************************************************************************
* OnDestroy() : 프로그램 종료 시에 수행해야할 루틴들
************************************************************************/
void CListedMotionDlg::OnDestroy()
{
	CDialog::OnDestroy();

	cmmLmAbortMotionEx(LM_MAP_IDX, DEC_TIME_SEC);
	cmmSxStop(AXIS, FALSE, FALSE);
	cmmGnDeviceUnload(); // Unload Device

	cmmUnloadDll(); // Unload DLL
}
```

Visual Basic

```
' ************************************************************************
' BtnStop_Click ()
' : This button stops motion.
' ************************************************************************

Const DEC_TIME_SEC = 2
Private Sub BtnStop_Click ()

	Call cmmLmAbortMotionEx(LM_MAP_IDX, DEC_TIME_SEC);

End Sub

' ************************************************************************
' Form_Unload () : 프로그램 종료 시에 수행해야할 루틴들
' ************************************************************************

Private Sub Form_Unload ()

	Call cmmLmAbortMotionEx(LM_MAP_IDX, DEC_TIME_SEC)
	Call cmmSxStop(AXIS, FALSE, FALSE)
	Call cmmGnDeviceUnload()

End Sub
```

Delphi

```
//************************************************************************
// OnBtnStop() : "Stop" Button's event handler
// : This button stops motion.
//************************************************************************

procedure OnBtnStop ();
begin

	cmmLmAbortMotionEx(LM_MAP_IDX, DEC_TIME_SEC);

end;

//************************************************************************
// OnDestroy() : 프로그램 종료 시에 수행해야할 루틴들
//************************************************************************

procedure OnDestroy ();
begin

	cmmLmAbortMotionEx(LM_MAP_IDX, DEC_TIME_SEC);
	cmmSxStop(AXIS, FALSE, FALSE);
	cmmGnDeviceUnload(); // Unload Device

end;
```

NAME

cmmLmDoPutOne
- Listed Motion 전용 단일(單一) DO 출력(出力)

INFORMATION

Listed Motion
VC++/VB
BCB/Delphi/.NET
Level 6
위험 요소 없음

SYNOPSIS

❒ VT_I4 cmmLmDoPutOne

([in] VT_I4 LmIndex, [in] VT_HANDLE hDoDevice, [in] VT_I4 Channel, [in] VT_I4 OutState)

DESCRIPTION

이 함수는 리스트 모션에서 디지털 출력 명령을 추가합니다. 한 개의 채널에 대해서 지정한 디지털 출력 명령을 수행합니다.

PARAMETER

▶ LmIndex : 리스트모션의 Map Index 를 의미합니다.

▶ hDoDevice : 디지털 출력의 장치 핸들 값입니다. 만일 리스트 모션 대상 모션제어 보드의 동일한 장치이면, 이 값을 NULL 로 설정합니다.

▶ Channel: 범용 디지털 출력으로 사용할 디지털 출력 채널을 설정합니다. 이 채널은 반드시 해당 모션 보드의 로컬(Local) 채널로 설정해야 합니다. 로컬(Local) 채널이라는 것은 CMMSDK 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 Channel 매개 변수(媒介變數)에 사용될 수 있는 번호는 장치의 순서에 관계없이 0 ~ 5 가 되는 것입니다.

▶ OutState : 출력할 출력할 상태를 의미합니다. 상태에 따라서 '0' 혹은 '1'로 설정됩니다.

RETURN VALUE

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

NAME

cmmLmDoPutMulti
- Listed Motion 전용 다중(多重) DO 출력(出力)

INFORMATION

Listed Motion
VC++/VB
BCB/Delphi/.NET
Level 6
위험 요소 없음

SYNOPSIS

❒ VT_I4 cmmLmDoPutMulti

([in] VT_I4 LmIndex, [in] VT_HANDLE hDoDevice, [in] VT_I4 ChannelGroup, [in] VT_I4 Mask, [in] VT_I4 OutStates)

DESCRIPTION

이 함수는 리스트 모션에서 디지털 출력 명령을 추가합니다. 여러 개의 채널에 대해서 지정한 디지털 출력 명령을 수행합니다.

PARAMETER

▶ LmIndex : 리스트모션의 Map Index 를 의미합니다.

▶ hDoDevice : 디지털 출력의 장치 핸들 값입니다. 만일 리스트 모션 대상 모션제어 보드의 동일한 장치이면, 이 값을 NULL 로 설정합니다.

▶ ChannelGroup: 범용 디지털 출력으로 사용할 디지털 출력 채널 그룹을 설정합니다. 이 채널은 반드시 해당 모션 보드의 로컬(Local) 채널로 설정해야 합니다. 로컬(Local) 채널이라는 것은 CMMSDK 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 ChannelGroup 매개 변수(媒介變數)에 사용될 수 있는 번호는 장치의 순서에 관계없이 0 ~ 5 가 되는 것입니다.

▶ Mask : 출력 대상 채널에 대한 비트 마스크(Bit Mask) 값입니다. 이 값을 통해 비트가 1 인 채널들이 Digital Output 대상 채널이 됩니다.

▶ OutStates : 출력할 상태를 의미합니다. 상태에 따라서 '0' 혹은 '1'로 설정됩니다.

RETURN VALUE

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

NAME

cmmLmDoPulseOne
- Listed Motion 단일(單一) 채널 단발 Pulse 출력(出力)

INFORMATION

Listed Motion

VC++/VB

BCB/Delphi/.NET

Level 6

위험 요소 없음

SYNOPSIS

❒ VT_I4 cmmLmDoPulseOne

([in] VT_I4 LmIndex, [in] VT_HANDLE hDoDevice, [in] VT_I4 Channel, [in] VT_I4 OutState, [in] VT_I4 Duration)

DESCRIPTION

이 함수는 리스트 모션에서 디지털 출력 명령을 추가합니다. 이 출력은 개별 채널(One-Channel)에 대해서 설정된 시간(Duration) 동안 ONE-SHOT(단발) 펄스를 출력하게 됩니다.

PARAMETER

▶ LmIndex : 리스트모션의 Map Index 를 의미합니다.

▶ hDoDevice : 디지털 출력의 장치 핸들 값입니다. 만일 리스트 모션 대상 모션제어 보드의 동일한 장치이면, 이 값을 NULL 로 설정합니다.

▶ Channel: 범용 디지털 출력으로 사용할 디지털 출력 채널을 설정합니다. 이 채널은 반드시 해당 모션 보드의 로컬(Local) 채널로 설정해야 합니다. 로컬(Local) 채널이라는 것은 CMMSDK 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 Channel 매개 변수(媒介變數)에 사용될 수 있는 번호는 장치의 순서에 관계없이 0 ~ 5 가 되는 것입니다.

▶ OutState : 출력할 출력할 상태를 의미합니다. 상태에 따라서 '0' 혹은 '1'로 설정됩니다.

▶ Duration : 단일 채널로 출력되는 펄스의 펄스 폭을 밀리초(millisecond) 단위 설정합니다.

RETURN VALUE

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

NAME

cmmLmDoPulseMulti
Listed Motion 다중(多重) 채널 단발 Pulse 출력(出力)

INFORMATION

- Listed Motion
- VC++/VB
- BCB/Delphi/.NET
- Level 6
- 위험 요소 없음

SYNOPSIS

❒ VT_I4 cmmLmDoPulseMulti

([in] VT_I4 LmIndex, [in] VT_HANDLE hDoDevice, [in] VT_I4 ChannelGroup, [in] VT_I4 Mask, [in] VT_I4 OutStates, [in] VT_I4 Duration)

DESCRIPTION

이 함수는 리스트 모션에서 디지털 출력 명령을 추가합니다. 이 출력은 다중 채널(Multi-Channel)에 대해서 설정된 시간(Duration) 동안 ONE-SHOT(단발) 펄스를 출력하게 됩니다.

PARAMETER

▶ LmIndex : 리스트모션의 Map Index 를 의미합니다.

▶ hDoDevice : 디지털 출력의 장치 핸들 값입니다. 만일 리스트 모션 대상 모션제어 보드의 동일한 장치이면, 이 값을 NULL 로 설정합니다.

▶ ChannelGroup: 범용 디지털 출력으로 사용할 디지털 출력 채널 그룹을 설정합니다. 이 채널은 반드시 해당 모션 보드의 로컬(Local) 채널로 설정해야 합니다. 로컬(Local) 채널이라는 것은 CMMSDK 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 ChannelGroup 매개 변수(媒介變數)에 사용될 수 있는 번호는 장치의 순서에 관계없이 0 ~ 5 가 되는 것입니다.

▶ Mask : 출력 대상 채널에 대한 비트 마스크(Bit Mask) 값입니다. 이 값을 통해 비트가 1 인 채널들이 Digital Output 대상 채널이 됩니다.

▶ OutStates : : 출력할 상태를 의미합니다. 상태에 따라서 '0' 혹은 '1'로 설정됩니다.

▶ Duration : 다중 채널로 출력되는 펄스의 펄스 폭을 밀리초(millisecond) 단위 설정합니다.

RETURN VALUE

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

NAME

cmmLmxStart
- Extend Listed Motion 작업 시작

INFORMATION

Extends List Motion

VC++/VB

BCB/Delphi/.NET

Level 6

다소 주의

Extend Listed Motion 에서 cmmLmxStart 는 cmmLmxEnd 와 서로 짝을 이룹니다.

SYNOPSIS

❒ VT_I4 cmmLmxStart([in] VT_I4 LmIndex,[in] VT_I4 AxisMask1,[in] VT_I4 AxisMask2)

DESCRIPTION

이 함수 호출을 통해 Extend Listed Motion 작업을 시작 합니다. 이 함수를 호출한 후 cmmLmxEnd() 함수가 호출되기 전까지 Sequence Item Id 로 설정된 명령들이 연속 적으로 수행됩니다.
연속 이송 작업 도중, cmmLmxPause() 및 cmmLmxResume() 함수를 통해 진행 중인 현재 Sequence Id 까지 수행 후 Pause 하거나 다시 재개(resume) 할 수 있습니다.
Extend Listed Motion 이 일반 Listed Motion 함수 기능과 가장 큰 차이점은 실행 도중에도 순차적 Sequence Id 를 사용해서 아직 수행되지 않은 다음 Sequence 명령을 예약, 추가 및 변경이 가능하다는 점입니다.

PARAMETER

▶ LmIndex : 리스트모션의 Map Index 를 의미합니다.
▶ AxisMask1 : 리스트모션에 포함시킬 축에 대한 하위 32 비트 Mask 값입니다. 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.
▶ AxisMask2 : 리스트모션에 포함시킬 축에 대한 상위 32 비트 Mask 값입니다. 이 값의 bit0 ~ bit31 은 각각 Axis32 ~ Axis63 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 * 참고: cmERR_LMX_ADD_ITEM_FAIL (-5200) 은 Extend Listed Motion 에 새로운 명령 Item 을 추가하는데 실패했다는 의미 입니다.
cmERR_NONE	수행 성공

REFERENCE

□ 리스트 모션(Listed Motion) 에 대한 REFERENCE 를 참고 하세요

EXAMPLE

C/C++

```
*********************************************************************************************************************
* OnBtnStart() : 확장 리스트 모션 동작 이벤트 핸들러 함수 (사용자 제작)
* Pick & Place 작업을 수행하는 X(좌우이송, 주행 축) 및 Z(상하이송)축 연속 모션 이송 예제 입니다.
* 하나의 쓰레드 함수를 사용해서 확장 리스트 모션을 수행합니다. VC6.0 기준으로 제작되었음에 유의하세요.
* 또한 예제에서 사용한 전역 변수 등은 임의로 사용하였으므로 유저 설정 값으로 사용하시기 바랍니다.
```

```
*****************************************************************************************************************************/
void CYourClass::OnBtnStart()
{
	AfxBeginThread(&ThfuncListMotion, this);
}

long PickAndPlaceWorks()
{
	long nResult;
	long nSeqId;
	long nStaIsBusy;
	double afDist[2];

	// step 1. Move To Position (1)
	afDist[0] = START_POS_X;
	afDist[1] = 0;
	cmmLmxSetNextItemId(LM_MAP_IDX, 0);
	nResult = cmmIxSetSpeedPattern(IX_MAP_IDX, cmFALSE, cmSMODE_T, g_nVelSpeedMultiply*100,
	   g_nAccSpeedMultiply*100, g_nAccSpeedMultiply*100);
	cmmIxLineTo(IX_MAP_IDX, afDist, cmTRUE);

	// step 2. (1) -> (2) Move
	afDist[0] = START_POS_X;
	afDist[1] = -5;
	cmmLmxSetNextItemId(LM_MAP_IDX, 1);
	nResult = cmmCfgSetSpeedPattern(cmY1, cmSMODE_T, g_nVelSpeedMultiply*1000,
	   g_nAccSpeedMultiply*1000/0.01, g_nAccSpeedMultiply*1000/0.01);
	nResult = cmmSxMoveTo(cmY1, afDist[1], cmTRUE);
	while(1){
		cmmLmxGetSts(LM_MAP_IDX, cmLMX_RUN_ITEM_ID, &nSeqId);
		cmmLmxGetSts(LM_MAP_IDX, cmLMX_BUSY, &nStaIsBusy);
		if(nSeqId == 1 && nStaIsBusy == 0){
			break;
		}
		if(g_bListMotionThreadStop){
			return;
		}
	}
	// step 3. Pick-Up Delay, 50ms
	Sleep(50); // with 1mm/sec Vel, 0.05sec Acc Time and 0.05sec Dec Time

	// step 4. (2) -> (1) Move
	afDist[0] = START_POS_X;
	afDist[1] = 0;
	cmmLmxSetNextItemId(LM_MAP_IDX, 2);
	nResult = cmmCfgSetSpeedPattern(cmY1, cmSMODE_T, g_nVelSpeedMultiply*1000,
	   g_nAccSpeedMultiply*1000/0.02, g_nAccSpeedMultiply*0);
	nResult = cmmSxMoveTo(cmY1, afDist[1], cmTRUE);

	// step 5. (1) -> (4) Move
	afDist[0] = START_POS_X + 120;
	afDist[1] = 50-5;
	cmmLmxSetNextItemId(LM_MAP_IDX, 3);
	nResult = cmmIxSetSpeedPattern(IX_MAP_IDX, cmFALSE, cmSMODE_T, g_nVelSpeedMultiply*100,
	   g_nAccSpeedMultiply*0, g_nAccSpeedMultiply*0);
	nResult = cmmIxLineTo(IX_MAP_IDX, afDist, cmTRUE);

	// step 6. (4) -> (5) Move
	afDist[0] = START_POS_X + 150;
	afDist[1] = 50-5;
	cmmLmxSetNextItemId(LM_MAP_IDX, 4);
	nResult = cmmIxSetSpeedPattern(IX_MAP_IDX, cmFALSE, cmSMODE_T, g_nVelSpeedMultiply*100,
	   g_nAccSpeedMultiply*0, g_nAccSpeedMultiply*100);
	nResult = cmmIxLineTo(IX_MAP_IDX, afDist, cmTRUE);

	// step 7. Place on Tray, (5) -> (6) Move
	afDist[0] = START_POS_X + 150;
	afDist[1] = 50-5-5;
	cmmLmxSetNextItemId(LM_MAP_IDX, 5);
	nResult = cmmCfgSetSpeedPattern(cmY1, cmSMODE_T, g_nVelSpeedMultiply*1000,
	   g_nAccSpeedMultiply*1000/0.05, g_nAccSpeedMultiply*1000/0.05);
	nResult = cmmSxMoveTo(cmY1, afDist[1], cmTRUE);
	while(1){
		cmmLmxGetSts(LM_MAP_IDX, cmLMX_RUN_ITEM_ID, &nSeqId);
		cmmLmxGetSts(LM_MAP_IDX, cmLMX_BUSY, &nStaIsBusy);
```

```
                if(nSeqId == 5 && nStaIsBusy == 0){
                        break;
                }
                if(g_bListMotionThreadStop){
                        return;
                }
        }

        // step 8. Place-Down Delay, 80ms
        Sleep(80); // with 1mm/sec Vel, 0.08sec Acc Time and 0.08sec Dec Time

        // step 9. (6) -> (5) Move
        afDist[0] = START_POS_X + 150;
        afDist[1] = 50-5;
        cmmLmxSetNextItemId(LM_MAP_IDX, 6);
        //nResult = cmmCfgSetSpeedPattern(cmY1, cmSMODE_T, g_nVelSpeedMultiply*1000,
           g_nAccSpeedMultiply*1000/0.05, g_nAccSpeedMultiply*1000/0.05); // the same as the previous step #7
        nResult = cmmSxMoveTo(cmY1, afDist[1], cmTRUE);

        // step 10. (5) -> (4) Move
        afDist[0] = START_POS_X + 120;
        afDist[1] = 50-5;
        cmmLmxSetNextItemId(LM_MAP_IDX, 7);
        nResult = cmmIxSetSpeedPattern(IX_MAP_IDX, cmFALSE, cmSMODE_T, g_nVelSpeedMultiply*100,
           g_nAccSpeedMultiply*100, g_nAccSpeedMultiply*0);
        nResult = cmmIxLineTo(IX_MAP_IDX, afDist, cmTRUE);

        // step 11. (4) -> (1) Move
        afDist[0] = START_POS_X;
        afDist[1] = 0;
        cmmLmxSetNextItemId(LM_MAP_IDX, 8);
        nResult = cmmIxSetSpeedPattern(IX_MAP_IDX, cmFALSE, cmSMODE_T, g_nVelSpeedMultiply*100,
           g_nAccSpeedMultiply*0, g_nAccSpeedMultiply*0);
        nResult = cmmIxLineTo(IX_MAP_IDX, afDist, cmTRUE);
}

UINT ThfuncListMotion(LPVOID pParam)
{
           // Extend List Motion Starts
        cmmLmxStart(LM_MAP_IDX, 0x5, 0x0); // Axis-X and Axis-Z
        cmmLmxSetSeqMode(LM_MAP_IDX, cmLMX_SEQM_WAIT_RUN);
        cmmIxMapAxes(IX_MAP_IDX, 0x5, 0x0);
        cmmCfgSetSpeedPattern(cmX1, cmSMODE_T, 2000, 2000/0.03, 2000/0.03);
        cmmCfgSetSpeedPattern(cmY1, cmSMODE_T, 1000, 1000/0.02, 1000/0.02);
        cmmIxSetSpeedPattern(IX_MAP_IDX, cmFALSE, cmSMODE_T, 100, 100, 100);

        PickAndPlaceWorks();

        // Extend List Motion Ends
        cmmLmxEnd(LM_MAP_IDX);
}
```

Visual Basic

```
'*************************************************************************************************************************
*
'*  BtnStart_Click() : 확장 리스트 모션 동작 이벤트 핸들러 함수 (사용자 제작)
'*  Pick & Place 작업을 수행하는 X(좌우이송, 주행 축) 및 Z(상하이송)축 연속 모션 이송 예제 입니다.
'*  하나의 쓰레드 함수를 사용해서 확장 리스트 모션을 수행합니다. VB6.0 기준으로 제작되었음에 유의하세요.
'*  또한 예제에서 사용한 전역 변수 등은 임의로 사용하였으므로 유저 설정 값으로 사용하시기 바랍니다.
'*  우선 프로젝트에서 Class Module 을 추가한 다음에 아래의 코드를 복사해 붙여 넣으세요.
'*  다음 프로젝트에 Module 을 추가한 후 그 아래 코드를 작성합니다.
'*  마지막으로 메인폼에서 버튼을 생성한 후 코드를 작성합니다.
'*************************************************************************************************************************
*

' Class Module(Module 명은 이 소스에서는 clsThreads)
' ------------------------------------------------------------------------------------------------------------------------
Option Explicit
Option Compare Text
```

```
Option Base 0

Private Type udtThread
   Handle As Long
   Enabled As Boolean
End Type

Private uThread As udtThread

Private Const CREATE_SUSPENDED As Long = &H4
Private Const THREAD_BASE_PRIORITY_IDLE  As Long = -15
Private Const THREAD_BASE_PRIORITY_LOWRT  As Long = 15
Private Const THREAD_BASE_PRIORITY_MAX As Long = 2
Private Const THREAD_BASE_PRIORITY_MIN As Long = -2
Private Const THREAD_PRIORITY_HIGHEST  As Long = THREAD_BASE_PRIORITY_MAX
Private Const THREAD_PRIORITY_LOWEST  As Long = THREAD_BASE_PRIORITY_MIN
Private Const THREAD_PRIORITY_ABOVE_NORMAL As Long = (THREAD_PRIORITY_HIGHEST - 1)
Private Const THREAD_PRIORITY_BELOW_NORMAL  As Long = (THREAD_PRIORITY_LOWEST + 1)
Private Const THREAD_PRIORITY_IDLE  As Long = THREAD_BASE_PRIORITY_IDLE
Private Const THREAD_PRIORITY_NORMAL As Long = 0
Private Const THREAD_PRIORITY_TIME_CRITICAL As Long = THREAD_BASE_PRIORITY_LOWRT

Private Declare Function CreateThread Lib "kernel32" (ByVal lpThreadAttributes As Any, ByVal dwStackSize As Long, ByVal
lpStartAddress As Long, lpParameter As Any, ByVal dwCreationFlags As Long, lpThreadId As Long) As Long
Private Declare Function ResumeThread Lib "kernel32" (ByVal hThread As Long) As Long
Private Declare Function SetThreadPriority Lib "kernel32" (ByVal hThread As Long, ByVal nPriority As Long) As Long
Private Declare Function GetThreadPriority Lib "kernel32" (ByVal hThread As Long) As Long
Private Declare Function SuspendThread Lib "kernel32" (ByVal hThread As Long) As Long
Private Declare Function TerminateThread Lib "kernel32" (ByVal hThread As Long, ByVal dwExitCode As Long) As Long

Public Sub Initialize(ByVal lpfnBasFunc As Long)
   Dim lStackSize As Long, lCreationFlags As Long, lpThreadId As Long, lNull As Long

On Error Resume Next
   lNull = 0
   lStackSize = 0
   lCreationFlags = CREATE_SUSPENDED
   uThread.Handle = CreateThread(lNull, lStackSize, lpfnBasFunc, lNull, lCreationFlags, lpThreadId)

   If uThread.Handle = lNull Then MsgBox "Thread 생성 실패~~"
End Sub

Public Property Get Enabled() As Boolean
On Error Resume Next
   Enabled = uThread.Enabled
End Property

Public Property Let Enabled(ByVal vNewValue As Boolean)
On Error Resume Next
   If vNewValue And (Not uThread.Enabled) Then
      ResumeThread uThread.Handle
      uThread.Enabled = True
   ElseIf uThread.Enabled Then
      SuspendThread uThread.Handle
      uThread.Enabled = False
   End If
End Property

Public Property Get Priority() As Long
On Error Resume Next
   Priority = GetThreadPriority(uThread.Handle)
End Property

Public Property Let Priority(ByVal vNewValue As Long)
On Error Resume Next
   If vNewValue = -2 Then
      Call SetThreadPriority(uThread.Handle, THREAD_PRIORITY_LOWEST)
   ElseIf vNewValue = -1 Then
      Call SetThreadPriority(uThread.Handle, THREAD_PRIORITY_BELOW_NORMAL)
   ElseIf vNewValue = 0 Then
      Call SetThreadPriority(uThread.Handle, THREAD_PRIORITY_NORMAL)
   ElseIf vNewValue = 1 Then
      Call SetThreadPriority(uThread.Handle, THREAD_PRIORITY_ABOVE_NORMAL)
   ElseIf vNewValue = 2 Then
      Call SetThreadPriority(uThread.Handle, THREAD_PRIORITY_HIGHEST)
```

```
    End If
End Property


Private Sub Class_Terminate()
On Error Resume Next
    Call TerminateThread(uThread.Handle, 0)     'Thread 종료 함수
End Sub
' ------------------------------------------------------------------------------------------------------------

' Module(Module 명은 사용자 임의로)
' ------------------------------------------------------------------------------------------------------------
Public Sub ThfuncListMotion()

    ' Extend List Motion Starts
    Call cmmLmxStart(LM_MAP_IDX, &H5, &H0) ' Axis-X and Axis-Z
    Call cmmLmxSetSeqMode(LM_MAP_IDX, cmLMX_SEQM_WAIT_RUN)
    Call cmmIxMapAxes(IX_MAP_IDX, &H5, &H0)
    Call cmmCfgSetSpeedPattern(cmX1, cmSMODE_T, 2000, 2000 / 0.03, 2000 / 0.03)
    Call cmmCfgSetSpeedPattern(cmY1, cmSMODE_T, 1000, 1000 / 0.02, 1000 / 0.02)
    Call cmmIxSetSpeedPattern(IX_MAP_IDX, cmFALSE, cmSMODE_T, 100, 100, 100)

    Call PickAndPlaceWorks

    ' Extend List Motion Ends
    Call cmmLmxEnd(LM_MAP_IDX)

End Sub

Public Sub PickAndPlaceWorks()

    Dim nResult As Long
    Dim nSeqId As Long
    Dim nStaIsBusy As Long
    Dim afDist(2) As Double

    ' step 1. Move To Position (1)
    afDist(0) = START_POS_X
    afDist(1) = 0
    Call cmmLmxSetNextItemId(LM_MAP_IDX, 0)
    nResult = cmmIxSetSpeedPattern(IX_MAP_IDX, cmFALSE, cmSMODE_T, g_nVelSpeedMultiply * 100,
g_nAccSpeedMultiply * 100, g_nAccSpeedMultiply * 100)
    Call cmmIxLineTo(IX_MAP_IDX, afDist, cmTRUE)

    ' step 2. (1) -> (2) Move
    afDist(0) = START_POS_X
    afDist(1) = -5
    Call cmmLmxSetNextItemId(LM_MAP_IDX, 1)
    nResult = cmmCfgSetSpeedPattern(cmY1, cmSMODE_T, g_nVelSpeedMultiply * 1000, g_nAccSpeedMultiply * 1000 / 0.01,
g_nAccSpeedMultiply * 1000 / 0.01)
    nResult = cmmSxMoveTo(cmY1, afDist(1), cmTRUE)
    While 1
        Call cmmLmxGetSts(LM_MAP_IDX, cmLMX_RUN_ITEM_ID, nSeqId)
        Call cmmLmxGetSts(LM_MAP_IDX, cmLMX_BUSY, nStaIsBusy)
        If nSeqId = 1 & nStaIsBusy = 0 Then
            break
        End If
        If g_bListMotionThreadStop Then
            Return
        End If
    Wend
    ' step 3. Pick-Up Delay, 50ms
    Call Sleep(50) ' with 1mm/sec Vel, 0.05sec Acc Time and 0.05sec Dec Time

    ' step 4. (2) -> (1) Move
    afDist(0) = START_POS_X
    afDist(1) = 0
    Call cmmLmxSetNextItemId(LM_MAP_IDX, 2)
    nResult = cmmCfgSetSpeedPattern(cmY1, cmSMODE_T, g_nVelSpeedMultiply * 1000, g_nAccSpeedMultiply * 1000 / 0.02,
g_nAccSpeedMultiply * 0)
    nResult = cmmSxMoveTo(cmY1, afDist(1), cmTRUE)

    ' step 5. (1) -> (4) Move
    afDist(0) = START_POS_X + 120
```

```
    afDist(1) = 50 - 5
    Call cmmLmxSetNextItemId(LM_MAP_IDX, 3)
    nResult = cmmIxSetSpeedPattern(IX_MAP_IDX, cmFALSE, cmSMODE_T, g_nVelSpeedMultiply * 100,
g_nAccSpeedMultiply * 0, g_nAccSpeedMultiply * 0)
    nResult = cmmIxLineTo(IX_MAP_IDX, afDist, cmTRUE)

    ' step 6. (4) -> (5) Move
    afDist(0) = START_POS_X + 150
    afDist(1) = 50 - 5
    Call cmmLmxSetNextItemId(LM_MAP_IDX, 4)
    nResult = cmmIxSetSpeedPattern(IX_MAP_IDX, cmFALSE, cmSMODE_T, g_nVelSpeedMultiply * 100,
g_nAccSpeedMultiply * 0, g_nAccSpeedMultiply * 100)
    nResult = cmmIxLineTo(IX_MAP_IDX, afDist, cmTRUE)

    ' step 7. Place on Tray, (5) -> (6) Move
    afDist(0) = START_POS_X + 150
    afDist(1) = 50 - 5 - 5
    Call cmmLmxSetNextItemId(LM_MAP_IDX, 5)
    nResult = cmmCfgSetSpeedPattern(cmY1, cmSMODE_T, g_nVelSpeedMultiply * 1000, g_nAccSpeedMultiply * 1000 / 0.05,
g_nAccSpeedMultiply * 1000 / 0.05)
    nResult = cmmSxMoveTo(cmY1, afDist(1), cmTRUE)
    While 1
        Call cmmLmxGetSts(LM_MAP_IDX, cmLMX_RUN_ITEM_ID, nSeqId)
        Call cmmLmxGetSts(LM_MAP_IDX, cmLMX_BUSY, nStaIsBusy)
        If nSeqId = 5 & nStaIsBusy = 0 Then
            break
        End If
        If g_bListMotionThreadStop Then
            Return
        End If
    Wend

    ' step 8. Place-Down Delay, 80ms
    Call Sleep(80) ' with 1mm/sec Vel, 0.08sec Acc Time and 0.08sec Dec Time

    ' step 9. (6) -> (5) Move
    afDist(0) = START_POS_X + 150
    afDist(1) = 50 - 5
    Call cmmLmxSetNextItemId(LM_MAP_IDX, 6)
    ' nResult = cmmCfgSetSpeedPattern(cmY1, cmSMODE_T, g_nVelSpeedMultiply*1000, g_nAccSpeedMultiply*1000/0.05,
g_nAccSpeedMultiply*1000/0.05) ' the same as the previous step #7
    nResult = cmmSxMoveTo(cmY1, afDist(1), cmTRUE)

    ' step 10. (5) -> (4) Move
    afDist(0) = START_POS_X + 120
    afDist(1) = 50 - 5
    Call cmmLmxSetNextItemId(LM_MAP_IDX, 7)
    nResult = cmmIxSetSpeedPattern(IX_MAP_IDX, cmFALSE, cmSMODE_T, g_nVelSpeedMultiply * 100,
g_nAccSpeedMultiply * 100, g_nAccSpeedMultiply * 0)
    nResult = cmmIxLineTo(IX_MAP_IDX, afDist, cmTRUE)

    ' step 11. (4) -> (1) Move
    afDist(0) = START_POS_X
    afDist(0) = 0
    Call cmmLmxSetNextItemId(LM_MAP_IDX, 8)
    nResult = cmmIxSetSpeedPattern(IX_MAP_IDX, cmFALSE, cmSMODE_T, g_nVelSpeedMultiply * 100,
g_nAccSpeedMultiply * 0, g_nAccSpeedMultiply * 0)
    nResult = cmmIxLineTo(IX_MAP_IDX, afDist, cmTRUE)

End Sub
' ----------------------------------------------------------------------------------------------------


' BtnStart_Click()
' ----------------------------------------------------------------------------------------------------

Private Sub BtnStart_Click()

        Dim ListMotionThread As New clsThreads

On Error Resume Next
        With ListMotionThread
                .Initialize AddressOf ThfuncListMotion
                .Enabled = True
        End With
```

```
        Set ListMotionThread = Nothing

End Sub
' ----------------------------------------------------------------------------------------------------
```

Delphi

```
//*****************************************************************************************************************
//*  TForm1.Button1Click(Sender: TObject) : 확장 리스트 모션 동작 이벤트 핸들러 함수 (사용자 제작)
//*  Pick & Place 작업을 수행하는 X(좌우이송, 주행 축) 및 Z(상하이송)축 연속 모션 이송 예제 입니다.
//*  하나의 쓰레드 함수를 사용해서 확장 리스트 모션을 수행합니다. Delphi7.0 기준으로 제작되었음에 유의하세요.
//*  또한 예제에서 사용한 전역 변수 등은 임의로 사용하였으므로 유저 설정 값으로 사용하시기 바랍니다.
//*****************************************************************************************************************

unit Unit1;

interface

uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
  Dialogs, StdCtrls, ComCtrls;

type
  TForm1 = class(TForm)
    Button1: TButton;
    procedure Button1Click(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;
  function ThfuncListMotion(P:Pointer) : LongInt; StdCall;
  function PickAndPlaceWorks():LongInt;
var
  Form1: TForm1;

implementation

uses CmmSdk;

{$R *.dfm}


function ThfuncListMotion(P:Pointer):LongInt;StdCall;
begin
        // Extend List Motion Starts
        cmmLmxStart(LM_MAP_IDX, $5, $0); // Axis-X and Axis-Z
        cmmLmxSetSeqMode(LM_MAP_IDX, cmLMX_SEQM_WAIT_RUN);
        cmmIxMapAxes(IX_MAP_IDX, $5, $0);
        cmmCfgSetSpeedPattern(cmX1, cmSMODE_T, 2000, 2000/0.03, 2000/0.03);
        cmmCfgSetSpeedPattern(cmY1, cmSMODE_T, 1000, 1000/0.02, 1000/0.02);
        cmmIxSetSpeedPattern(IX_MAP_IDX, cmFALSE, cmSMODE_T, 100, 100, 100);

        PickAndPlaceWorks();

        // Extend List Motion Ends
        cmmLmxEnd(LM_MAP_IDX);

end;


procedure TForm1.Button1Click(Sender: TObject);
var
        hThread : THandle;
        ThreadID : DWORD;
begin
        hThread := CreateThread(nil,0,@ThfuncListMotion,nil,0,ThreadID);

        if hThread = 0 then
                Application.MessageBox('FAIL','OK',0);
```

```
end;


function PickAndPlaceWorks():LongInt;
var
	nResult : LongInt;
	nSeqId : LongInt;
	nStaIsBusy : LongInt;
	afDist : Array[0..2] of Double;

begin

	// step 1. Move To Position (1)
	afDist[0] := START_POS_X;
	afDist[1] := 0;
	cmmLmxSetNextItemId(LM_MAP_IDX, 0);
	nResult := cmmIxSetSpeedPattern(IX_MAP_IDX, cmFALSE, cmSMODE_T, g_nVelSpeedMultiply*100,
g_nAccSpeedMultiply*100, g_nAccSpeedMultiply*100);
	cmmIxLineTo(IX_MAP_IDX, @afDist, cmTRUE);

	// step 2. (1) -> (2) Move
	afDist[0] := START_POS_X;
	afDist[1] := -5;
	cmmLmxSetNextItemId(LM_MAP_IDX, 1);
	nResult := cmmCfgSetSpeedPattern(cmY1, cmSMODE_T, g_nVelSpeedMultiply*1000, g_nAccSpeedMultiply*1000/0.01,
g_nAccSpeedMultiply*1000/0.01);
	nResult := cmmSxMoveTo(cmY1, afDist[1], cmTRUE);
	while True do
		cmmLmxGetSts(LM_MAP_IDX, cmLMX_RUN_ITEM_ID, @nSeqId);
		cmmLmxGetSts(LM_MAP_IDX, cmLMX_BUSY, @nStaIsBusy);
		if (nSeqId = 1) and (nStaIsBusy = 0) then ;
		if g_bListMotionThreadStop then return;

	// step 3. Pick-Up Delay, 50ms
	Sleep(50); // with 1mm/sec Vel, 0.05sec Acc Time and 0.05sec Dec Time

	// step 4. (2) -> (1) Move
	afDist[0] := START_POS_X;
	afDist[1] := 0;
	cmmLmxSetNextItemId(LM_MAP_IDX, 2);
	nResult := cmmCfgSetSpeedPattern(cmY1, cmSMODE_T, g_nVelSpeedMultiply*1000, g_nAccSpeedMultiply*1000/0.02,
g_nAccSpeedMultiply*0);
	nResult := cmmSxMoveTo(cmY1, afDist[1], cmTRUE);

	// step 5. (1) -> (4) Move
	afDist[0] := START_POS_X + 120;
	afDist[1] := 50-5;
	cmmLmxSetNextItemId(LM_MAP_IDX, 3);
	nResult := cmmIxSetSpeedPattern(IX_MAP_IDX, cmFALSE, cmSMODE_T, g_nVelSpeedMultiply*100,
g_nAccSpeedMultiply*0, g_nAccSpeedMultiply*0);
	nResult := cmmIxLineTo(IX_MAP_IDX, @afDist, cmTRUE);

	// step 6. (4) -> (5) Move
	afDist[0] := START_POS_X + 150;
	afDist[1] := 50-5;
	cmmLmxSetNextItemId(LM_MAP_IDX, 4);
	nResult := cmmIxSetSpeedPattern(IX_MAP_IDX, cmFALSE, cmSMODE_T, g_nVelSpeedMultiply*100,
g_nAccSpeedMultiply*0, g_nAccSpeedMultiply*100);
	nResult := cmmIxLineTo(IX_MAP_IDX, @afDist, cmTRUE);

	// step 7. Place on Tray, (5) -> (6) Move
	afDist[0] := START_POS_X + 150;
	afDist[1] := 50-5-5;
	cmmLmxSetNextItemId(LM_MAP_IDX, 5);
	nResult := cmmCfgSetSpeedPattern(cmY1, cmSMODE_T, g_nVelSpeedMultiply*1000, g_nAccSpeedMultiply*1000/0.05,
g_nAccSpeedMultiply*1000/0.05);
	nResult := cmmSxMoveTo(cmY1, afDist[1], cmTRUE);
	while true do
		cmmLmxGetSts(LM_MAP_IDX, cmLMX_RUN_ITEM_ID, @nSeqId);
		cmmLmxGetSts(LM_MAP_IDX, cmLMX_BUSY, @nStaIsBusy);
		if(nSeqId = 5) and (nStaIsBusy = 0) then break;
		if g_bListMotionThreadStop then return;

	// step 8. Place-Down Delay, 80ms
	Sleep(80); // with 1mm/sec Vel, 0.08sec Acc Time and 0.08sec Dec Time
```

```
        // step 9. (6) -> (5) Move
        afDist[0] := START_POS_X + 150;
        afDist[1] := 50-5;
        cmmLmxSetNextItemId(LM_MAP_IDX, 6);
        //nResult := cmmCfgSetSpeedPattern(cmY1, cmSMODE_T, g_nVelSpeedMultiply*1000,
g_nAccSpeedMultiply*1000/0.05, g_nAccSpeedMultiply*1000/0.05); // the same as the previous step #7
        nResult := cmmSxMoveTo(cmY1, afDist[1], cmTRUE);

        // step 10. (5) -> (4) Move
        afDist[0] := START_POS_X + 120;
        afDist[1] := 50-5;
        cmmLmxSetNextItemId(LM_MAP_IDX, 7);
        nResult := cmmIxSetSpeedPattern(IX_MAP_IDX, cmFALSE, cmSMODE_T, g_nVelSpeedMultiply*100,
g_nAccSpeedMultiply*100, g_nAccSpeedMultiply*0);
        nResult := cmmIxLineTo(IX_MAP_IDX, @afDist, cmTRUE);

        // step 11. (4) -> (1) Move
        afDist[0] := START_POS_X;
        afDist[1] := 0;
        cmmLmxSetNextItemId(LM_MAP_IDX, 8);
        nResult := cmmIxSetSpeedPattern(IX_MAP_IDX, cmFALSE, cmSMODE_T, g_nVelSpeedMultiply*100,
g_nAccSpeedMultiply*0, g_nAccSpeedMultiply*0);
        nResult := cmmIxLineTo(IX_MAP_IDX, @afDist, cmTRUE);

end;

end.
```

NAME

cmmLmxPause
- Extend Listed Motion 수행 중 현재 Sequence 명령을 완료 후 일시정지 합니다.

INFORMATION

- Extends List Motion
- VC++/VB
- BCB/Delphi/.NET
- Level 6
- ☺ 다소 주의

SYNOPSIS

❒ VT_I4 cmmLmxPause([in] VT_I4 LmIndex)

DESCRIPTION

Extend Listed Motion 수행 중 명령 시퀀스를 일시 정지하고 싶을 때 사용되는 함수이며, 현재 수행 중인 Sequence Item Id 에 해당하는 명령 까지는 진행된 후 Pause 가 됩니다.

PARAMETER

▶ LmIndex : 리스트모션의 Map Index 를 의미합니다.

RETURN VALUE

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

REFERENCE

□ 리스트 모션(Listed Motion) 에 대한 REFERENCE 를 참고 하세요

EXAMPLE

C/C++

cmmLmxStart() 함수 예제를 참고 하세요.

NAME

cmmLmxResume
- Extend Listed Motion 수행 중 일시 정지된 Sequence 명령을 재개 합니다.

INFORMATION

Extends List Motion
VC++/VB
BCB/Delphi/.NET
Level 6
다소 주의

SYNOPSIS

❒ VT_I4 cmmLmxResume([in] VT_I4 LmIndex, [in] VT_I4 IsClearQue)

DESCRIPTION

Extend Listed Motion 수행 중 cmmLmxPause() 명령에 의해 일시 정지된 명령 시퀀스를 재개하고자 할 때 사용 합니다.

PARAMETER

▶ LmIndex : 리스트모션의 Map Index 를 의미합니다.

▶ IsClearQue : 기존 명령 큐(Queue) 를 비울 것인지 여부를 설정합니다.

RETURN VALUE

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

REFERENCE

□ 리스트 모션(Listed Motion) 에 대한 REFERENCE 를 참고 하세요

EXAMPLE

C/C++

cmmLmxStart() 함수 예제를 참고 하세요.

NAME

cmmLmxEnd
- Extend Listed Motion 작업 종료

INFORMATION

Extends List Motion

VC++/VB

BCB/Delphi/.NET

Level 6

☺ 다소 주의

Extend Listed Motion 에서 cmmLmxEnd 는 cmmLmxStart 와 서로 짝을 이룹니다.

SYNOPSIS

❒ VT_I4 cmmLmxEnd([in] VT_I4 LmIndex)

DESCRIPTION

이 함수 호출을 통해 Extend Listed Motion 작업을 종료 합니다.

PARAMETER

▶ LmIndex : 리스트모션의 Map Index 를 의미합니다.

RETURN VALUE

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

REFERENCE

□ 리스트 모션(Listed Motion) 에 대한 REFERENCE 를 참고 하세요

EXAMPLE

C/C++

cmmLmxStart() 함수 예제를 참고 하세요.

NAME

cmmLmxSetSeqMode
- Extend Listed Motion 에서 이송명령 예약시 기존 명령 버퍼가 Full 인 경우 처리 방법 설정

INFORMATION

- Extends List Motion
- VC++/VB
- BCB/Delphi/.NET
- Level 6
- ☺ 다소 주의

SYNOPSIS

❒VT_I4 cmmLmxSetSeqMode

([in] VT_I4 LmIndex, [in] VT_I4 SeqMode)

DESCRIPTION

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.

PARAMETER

▶ LmIndex : 리스트모션의 Map Index 를 의미합니다.
▶ SeqMode : 예약하려는 명령 처리 방법

Value	Meaning
cmLMX_SEQM_SKIP_RUN (또는 0)	Extend Listed Motion 수행 중, 새로 예약하려는 명령이 SKIP 됩니다. 'cmERR_LMX_ADD_ITEM_FAIL' 에러를 발생하고 함수는 바로 반환됩니다.
cmLMX_SEQM_WAIT_RUN (또는 1)	Extend Listed Motion Buffer (명령 버퍼) 에 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 * 참고: cmERR_LMX_ADD_ITEM_FAIL (-5200) 은 Extend Listed Motion 에 새로운 명령 Item 을 추가하는데 실패했다는 의미 입니다.
cmERR_NONE	수행 성공

REFERENCE

□ 리스트 모션(Listed Motion) 에 대한 REFERENCE 를 참고 하세요

EXAMPLE

C/C++

cmmLmxStart() 함수 예제를 참고 하세요.

NAME

cmmLmxGetSeqMode
– Extend Listed Motion 에서 이송명령 예약시 기존 명령 버퍼가 Full 인 경우 처리 방법 반환

INFORMATION

- Extends List Motion
- VC++/VB
- BCB/Delphi/.NET
- Level 6
- ☺ 다소 주의

SYNOPSIS

❒VT_I4 cmmLmxGetSeqMode

([in] VT_I4 LmIndex, [in] VT_PI4 SeqMode)

DESCRIPTION

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.

PARAMETER

▶ LmIndex : 리스트모션의 Map Index 를 의미합니다.
▶ SeqMode : 예약하려는 명령 처리 방법

Value	Meaning
cmLMX_SEQM_SKIP_RUN (또는 0)	Extend Listed Motion 수행 중, 새로 예약하려는 명령이 SKIP 됩니다. 'cmERR_LMX_ADD_ITEM_FAIL' 에러를 발생하고 함수는 바로 반환됩니다.
cmLMX_SEQM_WAIT_RUN (또는 1)	Extend Listed Motion Buffer (명령 버퍼) 에 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 * 참고: cmERR_LMX_ADD_ITEM_FAIL (-5200) 은 Extend Listed Motion 에 새로운 명령 Item 을 추가하는데 실패했다는 의미 입니다.
cmERR_NONE	수행 성공

REFERENCE

□ 리스트 모션(Listed Motion) 에 대한 REFERENCE 를 참고 하세요

EXAMPLE

C/C++

cmmLmxStart() 함수 예제를 참고 하세요.

NAME

cmmLmxSetNextItemId
- Extend Listed Motion 에서 Sequence Item Id 설정

INFORMATION

- Extends List Motion
- VC++/VB
- BCB/Delphi/.NET
- Level 6
- ☺ 다소 주의

SYNOPSIS

❒VT_I4 cmmLmxSetNextItemId

([in] VT_I4 LmIndex, [in] VT_I4 SeqId)

DESCRIPTION

Extend Listed Motion 에서 수행할 명령(Item)에 대해 Sequence Item Id 를 설정합니다.
이 Sequence Item Id 는 cmmLmxGetSts() 함수를 통해 현재 동작 중인 Sequence 상태를 확인 하거나
cmmLmxGetNextItemId 로 다음 수행할 Sequence Item Id 를 확인하는 데 사용되므로 중복된 Id 를 사용하시면 안되며
고유의 순차적 Id 로 설정되어야 합니다. 자세한 내용은 cmmLmxStart() 함수에 소개된 예제를 참고 하세요.

PARAMETER

▶ LmIndex : 리스트모션의 Map Index 를 의미합니다.
▶ SeqId : 해당 Sequence 단계에 해당하는 이송 또는 설정 함수에 대해 설정할 순차적 Sequence Item Id

RETURN VALUE

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

REFERENCE

□ 리스트 모션(Listed Motion) 에 대한 REFERENCE 를 참고 하세요

EXAMPLE

C/C++

cmmLmxStart() 함수 예제를 참고 하세요.

NAME

cmmLmxGetNextItemId
– Extend Listed Motion 에서 다음 수행 예정인 Sequence Item Id 반환

INFORMATION

- Extends List Motion
- VC++/VB
- BCB/Delphi/.NET
- Level 6
- ☺ 다소 주의

SYNOPSIS

❒VT_I4 cmmLmxGetNextItemId

([in] VT_I4 LmIndex, [in] VT_PI4 SeqId)

DESCRIPTION

Extend Listed Motion 에서 다음 수행할 명령(Item)에 해당하는 Sequence Item Id 를 반환합니다.
이 Sequence Item Id 는 cmmLmxGetSts() 함수를 통해 현재 동작 중인 Sequence 상태를 확인 하거나 cmmLmxGetNextItemId 로 다음 수행할 Sequence Item Id 를 확인하는 데 사용되므로 중복된 Id 를 사용하시면 안되며 고유의 순차적 Id 로 설정되어야 합니다. 자세한 내용은 cmmLmxStart() 함수에 소개된 예제를 참고 하세요.

PARAMETER

▶ LmIndex : 리스트모션의 Map Index 를 의미합니다.
▶ SeqId : 해당 Sequence 단계에 해당하는 이송 또는 설정 함수에 대해 설정할 순차적 Sequence Item Id

RETURN VALUE

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

REFERENCE

□ 리스트 모션(Listed Motion) 에 대한 REFERENCE 를 참고 하세요

EXAMPLE

C/C++

cmmLmxStart() 함수 예제를 참고 하세요.

NAME

cmmLmxSetNextItemParam
- Extend Listed Motion 의 다음 수행 예정인 명령에 대한 함수 파라미터 설정값 설정

INFORMATION

- Extends List Motion
- VC++/VB
- BCB/Delphi/.NET
- Level 6
- ☺ 다소 주의

SYNOPSIS

❒VT_I4 cmmLmxSetNextItemParam

([in] VT_I4 LmIndex, [in] VT_I4 ParamIdx, [in] VT_I4 ParamData)

DESCRIPTION

Extend Listed Motion 에서 다음 수행 예정인 명령에 대한 함수 파라미터 설정 값 설정합니다.

PARAMETER

▶ LmIndex : 리스트모션의 Map Index 를 의미합니다.

▶ ParamIdx : 해당 명령에 대해 값 변경을 원하는 파라미터의 인덱스 값

▶ ParamData : 해당 명령에 대해 새로 설정하려는 파라미터 데이터 값

RETURN VALUE

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

REFERENCE

□ 리스트 모션(Listed Motion) 에 대한 REFERENCE 를 참고 하세요

EXAMPLE

C/C++

cmmLmxStart() 함수 예제를 참고 하세요.

NAME

cmmLmxGetNextItemParam
- Extend Listed Motion 의 다음 수행 예정인 명령에 대한 함수 파라미터 설정값 반환

INFORMATION

- Extends List Motion
- VC++/VB
- BCB/Delphi/.NET
- Level 6
- ☺ 다소 주의

SYNOPSIS

❒VT_I4 cmmLmxGetNextItemParam

([in] VT_I4 LmIndex, [in] VT_I4 ParamIdx, [out] VT_PI4 ParamData)

DESCRIPTION

Extend Listed Motion 에서 다음 수행 예정인 명령에 대한 함수 파라미터 설정 값 반환합니다.

PARAMETER

▶ LmIndex : 리스트모션의 Map Index 를 의미합니다.

▶ ParamIdx : 해당 명령에 대해 값을 알기를 원하는 파라미터의 인덱스 값

▶ ParamData : 해당 명령에 대해 설정된 파라미터 데이터 값

RETURN VALUE

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

REFERENCE

□ 리스트 모션(Listed Motion) 에 대한 REFERENCE 를 참고 하세요

EXAMPLE

C/C++

cmmLmxStart() 함수 예제를 참고 하세요.

NAME

cmmLmxGetRunItemParam
- Extend Listed Motion 의 현재 수행중인 명령에 대한 함수 파라미터 설정값 반환

INFORMATION

- Extends List Motion
- VC++/VB
- BCB/Delphi/.NET
- Level 6
- ☺ 다소 주의

SYNOPSIS

❒ VT_I4 cmmLmxGetRunItemParam

([in] VT_I4 LmIndex, [in] VT_I4 ParamIdx, [out] VT_PI4 ParamData)

DESCRIPTION

Extend Listed Motion 에서 현재 수행 중인 명령에 대한 함수 파라미터 설정 값 반환합니다.

PARAMETER

▶ LmIndex : 리스트모션의 Map Index 를 의미합니다.

▶ ParamIdx : 해당 명령에 대해 값을 알기를 원하는 파라미터의 인덱스 값

▶ ParamData : 해당 명령에 대해 설정된 파라미터 데이터 값

RETURN VALUE

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

REFERENCE

□ 리스트 모션(Listed Motion) 에 대한 REFERENCE 를 참고 하세요

EXAMPLE

C/C++

cmmLmxStart() 함수 예제를 참고 하세요.

NAME

cmmLmxGetRunItemStaPos
- Extend Listed Motion 의 현재 수행중인 명령이 수행되기 직전 해당 축 위치 반환

INFORMATION

- Extends List Motion
- VC++/VB
- BCB/Delphi/.NET
- Level 6
- ☺ 다소 주의

SYNOPSIS

❒ VT_I4 cmmLmxGetRunItemStaPos([in] VT_I4 LmIndex,[in] VT_I4 Axis,

[out] VT_PR8 Position)

DESCRIPTION

Extend Listed Motion 수행 중에 현재 수행 중인 명령(Current Sequence Item Id) 이 수행되기 직전에 해당 축의 Command Pulse Position 값을 반환 합니다.

PARAMETER

▶ LmIndex : 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

RETURN VALUE

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

REFERENCE

□ 리스트 모션(Listed Motion) 에 대한 REFERENCE 를 참고 하세요

EXAMPLE

C/C++

cmmLmxStart() 함수 예제를 참고 하세요.

NAME

cmmLmxGetRunItemTargPos
- Extend Listed Motion 의 현재 수행중인 명령에서 해당 축 이송 목표 위치 반환

INFORMATION

- Extends List Motion
- VC++/VB
- BCB/Delphi/.NET
- Level 6
- ☺ 다소 주의

SYNOPSIS

❒ VT_I4 cmmLmxGetRunItemTargPos([in] VT_I4 LmIndex, [in] VT_I4 Axis,

[out] VT_PR8 Position)

DESCRIPTION

Extend Listed Motion 수행 중에 현재 수행 중인 명령(Current Sequence Item Id) 에 대해 해당 축의 목표 좌표에 해당하는 Command Pulse Position 값을 반환 합니다.

PARAMETER

▶ LmIndex : 리스트모션의 Map Index 를 의미합니다.

▶ Axis : zero-based (0 번축 기준) 로 축 번호를 지정합니다.

▶ 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
cmERR_NONE	수행 성공

REFERENCE

□ 리스트 모션(Listed Motion) 에 대한 REFERENCE 를 참고 하세요

EXAMPLE

C/C++

cmmLmxStart() 함수 예제를 참고 하세요.

NAME

cmmLmxGetSts
- Extend Listed Motion 에서 현재 수행 중인 Sequence 명령에 대한 상태 값 확인

INFORMATION

- Extends List Motion
- VC++/VB
- BCB/Delphi/.NET
- Level 6
- ☺ 다소 주의

SYNOPSIS

❒ VT_I4 cmmLmxGetSts([in] VT_I4 LmIndex,[in] VT_I4 LmxStsId, [out] VT_PI4 LmxStsVal)

DESCRIPTION

Extend Listed Motion 에서 현재 수행 중인 Sequence 명령에 대한 상태 값을 Status Id 값을 통해 확인이 가능한 함수입니다.

PARAMETER

▶ LmIndex : 리스트모션의 Map Index 를 의미합니다.
▶ LmxStsId : 각종 Status ID

Value	Meaning
cmLMX_STARTED (또는 0)	Extend Listed Motion 기능이 활성화 되었는지를 나타내는 status
cmmLMX_BUSY (또는 1)	Extend Listed Motion 이 현재 실제로 이송을 진행 중인지를 나타내는 status
cmLMX_FREE_SPACE (또는 2)	Extend Listed Motion 버퍼의 여유 공간. 반환되는 값은 바이트 단위가 아니라 등록할 수 있는 Item 개수입니다.
cmLMX_RUN_ITEM_NO (또는 3)	현재 이송되고 있거나 마지막 이송된 Item 의 번호
cmLMX_RUN_ITEM_ID (또는 4)	현재 이송되고 있거나 마지막 이송된 Item 의 Id 값 (Id 는 cmmLmxSetNextItemId() 함수를 통해 설정된 값 입니다.
cmLMX_LAST_SET_ITEM_ID (또는 5)	마지막으로 예약된 Item 의 Id 값.

▶ LmxStsVal : 해당 Sequence 단계에 해당하는 이송 또는 설정 함수에 대해 설정할 순차적 Sequence Item Id

RETURN VALUE

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

REFERENCE

□ 리스트 모션(Listed Motion) 에 대한 REFERENCE 를 참고 하세요

EXAMPLE

C/C++

cmmLmxStart() 함수 예제를 참고 하세요.

Chapter

10

Input signals related to motion control by external signal

모션 제어에서의 외부 신호 입력은 다양(多樣)한 모션 제어로 응용될 수 있습니다. MPG 와 같은 펄스 출력(出力) 장치로부터 발생된 펄스 신호는 커미조아의 모션 하드웨어를 통해 입력되어, 다양한 모션 제어에 응용될 수 있습니다. 또한, 다양한 외부 신호의 응용으로 입력되는 신호를 통해 동작 방향이나 동작의 시작과 종료 시점을 제어할 수 있습니다.

본 단원에서는 Manual Pulsar 모드와 외부 스위치에 의한 모션제어 함수들을 소개합니다. Manual Pulsar 모드는 Pulsar 모드는 PA 입력(入力) 신호와 PB 입력(入力) 신호에 동기하여 모터를 제어하는 모드입니다. PA/PB 를 모드입니다. PA/PB 를 통하여 입력되는 신호는 지연시간 없이 Command 로 전달됩니다. 외부 신호에 의해 신호에 의해 동기(同期)되는 모션제어는 다양한 외부 신호 구성을 통해 모션제어의 동작의 속성을 속성을 정의(定義)할 수 있습니다.

10 외부신호 동기제어 편

10.1 Manual Pulsar (PA/PB) 모드 모션제어

이 단원에서는 Manual Pulsar 모드 모션에 관련된 함수들을 소개합니다. Manual Pulsar 모드는 PA 입력 신호와 PB 입력 신호에 동기하여 모터를 제어하는 모드입니다. PA/PB를 통하여 입력되는 신호는 지연시간 없이 Command로 전달됩니다.

Manual Pulsar 모드를 사용하면 로터리 엔코더(Rotary Encoder)와 같은 장치로부터 PA/PB 단자를 통하여 Plus 펄스와 Minus 펄스(CW/CCW) 또는 90°위상차를 갖는 A/B Phase 펄스를 입력받아 수동으로 모션을 제어할 수 있습니다. 또는 다른 모터나 외부장치와 동기제어를 하는데에도 유용하게 사용될 수 있습니다. 특히 Manual Pulsar 모드는 ㈜커미조아 모션제어보드의 "Master/Slave 모션제어"기능에 응용됩니다.

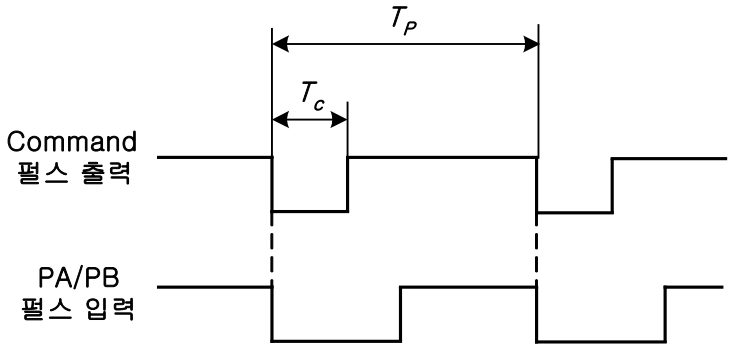
PA/PB 단자에 입력되는 신호의 형태는 Pulsar 입력모드 설정에 따라 달라지며 이는 cmmPlsrSetInMode() 함수에 의해 설정됩니다.

Manual Pulsar 모드 모션은 Velocity Motion은 물론 상대좌표/절대좌표 In-Position 모션에도 적용가능합니다. 그러나 Coordinated Motion에는 적용할 수 없습니다.

□ Manual Pulsar 모드 모션에서의 속도는 입력 펄스의 주파수에 의해 결정됩니다. 따라서 속도모드나 가/감속도는 무시됩니다. 그러나 Manual Pulsar 입력신호의 최대 주파수는 cmmCfgSetSpeedPattern() 함수에서 설정하는 작업속도에 의해서 결정됩니다. 따라서 Manual Pulsar 모션을 수행하기 전에 cmmCfgSetSpeedPattern() 함수를 이용하여 작업속도, 즉 입력신호의 최대속도를 적절한 값으로 설정하여야 합니다. 설정한 최대속도를 초과하는 PA/PB 신호가 입력되면 오동작하게 됩니다. 주의할 것은 작업속도는 논리속도로 설정되므로 PA/PB 단자로 입력될 수 있는 펄스의 최대 주파수(fmax)는 다음과 같습니다.

PA/PB 단자로 입력될 수 있는 펄스의 최대 주파수 산출
$f_{\max} = V_{work} \times f_u$ fmax : PA/PB로 입력될 수 있는 펄스의 최대 주파수(PPS) Vwork : 작업속도 fu : Unit speed 값

□ cmmCfgSetSpeedPattern() 함수에서 설정하는 작업속도는 PA/PB 단자로 입력될 수 있는 펄스의 최대주파수를 제한하는 동시에 해당축의 COMMAND 출력 펄스의 폭을 결정하기도 합니다. 따라서 작업속도를 너무 높게 설정하면 펄스폭이 너무 짧아서 모터 드라이버에서 받아들이지 못할 수 있습니다. 예를 들어 미쯔비시 MR-J2 서보드라이버는 기본적인 사양일 때 받아들일 수 있는 펄스의 최대 주파수는 500 KPPS 입니다. 따라서 COMMAND 펄스의 펄스폭은 (1000000/(500000*2)) = 1 μsec 이상이어야합니다. (최대 주파수에서 실제 입력 펄스의 시간은 ON/OFF 를 포함한 것이므로, COMMAND 펄스의 펄스폭에 대한 값은 실제 최대 주파수의 2 배가 되어야 합니다) 이러한 경우에, 만일 cmmCfgSetSpeedPattern() 함수에서 설정하는 작업속도를 500 KPPS보다 큰값으로 설정하면 펄스폭이 짧아서 Command 펄스를 서보드라이버가 받아들이지 못할 수 있습니다.



T_c : Pulsar 모드시에 Command 펄스의 폭
$T_c = 1000000 * (1/f_{max})/2$ (μsec)
T_p : PA/PB 입력펄스의 주기

□ PA/PB 입력에 대한 하드웨어적인 회로 및 신호연결 방법은 "User's Guide" 매뉴얼의 "Part II Chapter2 2.2 모션컨트롤러 인터페이스 신호"단원을 참조하시기 바랍니다.

□ Manual Pulsar 구동 명령이 시작되면 cmmStReadMotionState() 함수의 반환값이 8(cmMST_WAIT_PLSR)이 됩니다. 이를 확인(確認)하면 Manual Pulsar 구동 명령 시작여부를 확인(確認)할 수 있습니다.

10.1.1 함수 요약

Manual Pulsar 모드 모션에 관련된 함수들은 다음과 같습니다.

Summary of Functions
❒ VT_I4 cmmPlsrSetInMode ([in] VT_I4 Axis, [in] VT_I4 InputMode, [in] VT_I4 IsInverse) 대상(對象) 모션 채널에 대해서, Manual Pulsar 입력(入力) 신호에 대한 환경설정(環境設定)을 구성합니다.
❒ VT_I4 cmmPlsrGetInMode ([in] VT_I4 Axis, [out] VT_PI4 InputMode, [out] VT_PI4 IsInverse) 대상(對象) 모션 채널에 대해서, Manual Pulsar 입력(入力) 신호에 대한 환경설정(環境設定)을 반환(返還)합니다.
❒ VT_I4 cmmPlsrSetGain ([in] VT_I4 Axis, [in] VT_I4 GainFactor, [in] VT_I4 DivFactor) 대상(對象) 모션 채널에 대해서, Manual Pulsar 의 PA/PB 입력(入力) 펄스 대비 지령 펄스(Command Pulse)의 출력(出力) 펄스 수에 대한 비율을 설정합니다.
❒ VT_I4 cmmPlsrGetGain ([in] VT_I4 Axis, [out] VT_PI4 GainFactor, [out] VT_PI4 DivFactor) 대상(對象) 모션 채널에 대해서, Manual Pulsar 의 PA/PB 입력 펄스 대비 지령 펄스(Command Pulse)의 출력 펄스 수에 대한 비율을 반환(返還)합니다.
❒ VT_I4 cmmPlsrHomeMoveStart ([in] VT_I4 Axis, [in] VT_I4 HomeType) 대상(對象) 모션 채널에 대해서, Manual Pulsar 입력 신호에 의한 원점 복귀를 이송을 수행합니다. 이 구동 함수는 구동 시작 후 바로 반환(返還)됩니다.
❒ VT_I4 cmmPlsrMove ([in] VT_I4 Axis, [in] VT_R8 Distance, [in] VT_I4 IsBlocking) 대상(對象) 모션 채널에 대해서, Manual Pulsar 입력 신호에 맞추어 지정한 논리적(論理的) 거리를 통해 상대좌표이송(相對座標移送)을 수행합니다. 이 구동 함수는 구동 시작 후 바로 반환(返還)되지 않습니다.
❒ VT_I4 cmmPlsrMoveStart ([in] VT_I4 Axis, [in] VT_R8 Distance) 대상(對象) 모션 채널에 대해서, Manual Pulsar 입력 신호에 맞추어 지정한 논리적(論理的) 거리를 통해 상대좌표이송(相對座標移送)을 수행합니다. 이 구동 함수는 구동 시작 후 바로 반환(返還) 됩니다.
❒ VT_I4 cmmPlsrMoveTo ([in] VT_I4 Axis, [in] VT_R8 Position, [in] VT_I4 IsBlocking) 대상(對象) 모션 채널에 대해서, Manual Pulsar 입력 신호에 맞추어 지정한 논리적 거리를 통해 절대좌표이송(絶對座標移送)을 수행합니다. 이 구동 함수는 구동 시작 후 바로 반환(返還) 되지 않습니다.

❒ VT_I4 cmmPlsrMoveToStart ([in] VT_I4 Axis, [in] VT_R8 Position)
대상(對象) 모션 채널에 대해서, Manual Pulsar 입력 신호에 맞추어 지정한 논리적(論理的) 거리를 통해 절대좌표이송(絶對座標移送)을 수행합니다. 이 구동 함수는 구동 시작 후 바로 반환(返還) 됩니다.

❒ VT_I4 cmmPlsrVMoveStart ([in] VT_I4 Axis)
대상(對象) 모션 채널에 대해서, Manual Pulsar 입력 신호에 맞추어. 연속속도이송(連續速度移送) 을 시작합니다. 이 구동(驅動) 함수는 구동(驅動) 시작 후 바로 반환됩니다.

❒ VT_I4 cmmPlsrIsActive ([in] VT_I4 Axis, [out] VT_PI4 IsActive)
대상(對象) 모션 채널에 대해서, Manual Pulsar 신호의 입력 상태를 확인(確認)합니다.

10.1.2 함수 설명

NAME

cmmPlsrSetInMode
cmmPlsrGetInMode
– Pulsar 입력(入力) 신호 환경설정(環境設定)

INFORMATION

Manual Pulsar Motion

VC++/VB

BCB/Delphi/.NET

Level 5

다소 주의

Manual Pulsar 기능을 사용하는 모션 채널(Axis)에서는 내부적인 하드웨어 구조상으로 Master / Slave 기능을 사용할 수 없습니다.

SYNOPSIS

❒ VT_I4 cmmPlsrSetInMode ([in] VT_I4 Axis, [in] VT_I4 InputMode, [in] VT_I4 IsInverse)

❒ VT_I4 cmmPlsrGetInMode ([in] VT_I4 Axis, [out] VT_PI4 InputMode, [out] VT_PI4 IsInverse)

DESCRIPTION

cmmPlsrSetInMode() 함수는 Pulsar 입력 신호에 대한 환경을 설정하며, cmmPlsrGetInMode() 함수는 Pulsar 입력 신호의 환경설정값을 읽어옵니다. 설정되는 Pulsar 입력 신호 설정은 크기 A 상과 B 상을 검출하는 모드와 CW/CCW 신호 검출 모드가 있습니다. 신호의 성격에 맞게 입력 신호의 환경 설정을 해주시기 바랍니다.

PARAMETER

▶ Axis: 축번호. 축번호는 상수값으로 [cmX1] 부터 0 번째 축을 기준 축으로 임의의 축을 설정할 수 있습니다.

▶ InputMode : cmmPlsrSetInMode 함수의 인자이며, PA 와 PB 입력 단자를 통하여 입력되는 Pulsar 입력 신호의 입력모드를 설정합니다. 설정가능한 값은 다음과 같습니다.

Value	Meaning
0 또는 cmIMODE_AB1X	1X A/B (1 채배 Phase type 입력 모드)
1 또는 cmIMODE_AB2X	2X A/B (2 채배 Phase type 입력 모드)
2 또는 cmIMODE_AB4X	4X A/B (4 채배 Phase type 입력 모드)
3 또는 cmIMODE_CWCCW	CW/CCW (PA – Plus direction move, PB – Minus direction move)

▶ InputMode : cmmPlsrGetInMode 함수의 인자이며, 입력 신호의 입력모드를 반환합니다. 반환값은 다음과 같습니다.

Value	Meaning
0 또는 cmIMODE_AB1X	1X A/B (1 채배 Phase type 입력 모드)
1 또는 cmIMODE_AB2X	2X A/B (2 채배 Phase type 입력 모드)
2 또는 cmIMODE_AB4X	4X A/B (4 채배 Phase type 입력 모드)
3 또는 cmIMODE_CWCCW	CW/CCW (PA – Plus direction move, PB – Minus direction move)

▶ IsInverse : cmmPlsrSetInMode 함수의 인자이며, Pulsar 입력 신호에 의해 결정되는 방향(Direction)을 모션에 반대로 적용할 지를 결정합니다. 설정가능한 값은 다음과 같습니다.

Value	Meaning
0	Pulsar 입력 신호가 나타내는 방향과 모션의 방향 일치
1	Pulsar 입력 신호가 나타내는 방향과 모션의 방향을 반대로 적용

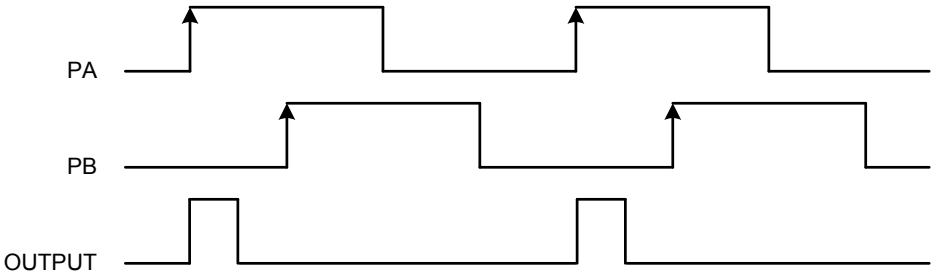
▶ IsInverse : cmmPlsrGetInMode 함수의 인자이며, Pulsar 입력 신호에 의해 결정되는 방향(Direction)을 모션에 반대로 적용되어 있는 지를 반환합니다. 반환되는 값은 다음과 같습니다.

Value	Meaning
0	Pulsar 입력 신호가 나타내는 방향과 모션의 방향 일치
1	Pulsar 입력 신호가 나타내는 방향과 모션의 방향이 반대

REFERENCE

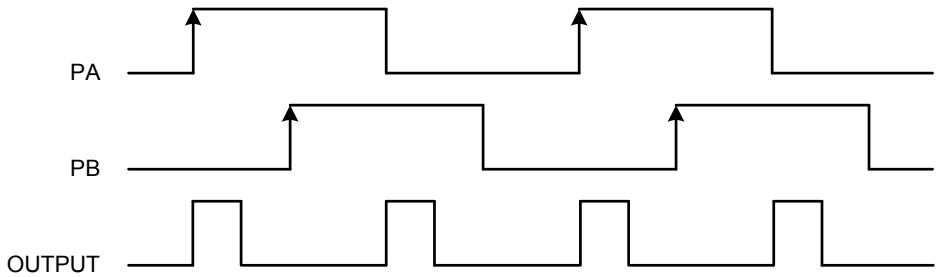
☐ InputMode 설정에 따라서 PA/PB 입력 신호와 COMMAND 출력 펄스의 관계는 다음과 같습니다.

• InputMode = cmIMODE_AB1X 일때



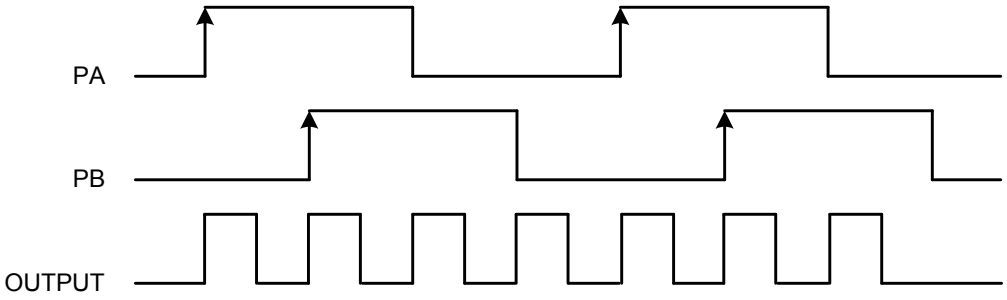
+/- 방향은 PA 신호와 PB 신호의 위상에 따라서 결정됩니다.

• InputMode = cmIMODE_AB2X 일때



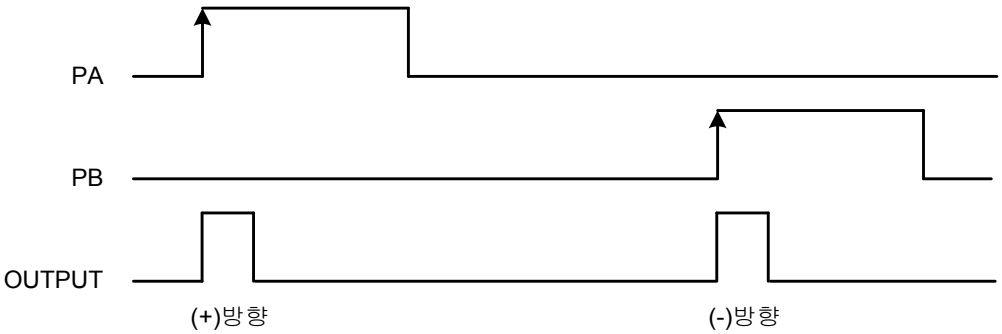
+/- 방향은 PA 신호와 PB 신호의 위상에 따라서 결정됩니다.

• InputMode = cmIMODE_AB4X 일때



+/- 방향은 PA 신호와 PB 신호의 위상에 따라서 결정됩니다.

• InputMode = cmIMODE_CWCCW 일때



RETURN VALUE

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

EXAMPLE

C/C++

```
#include "Cmmsdk.h"
#include "CmmsdkDef.h"

void OnSetPlsrInMode ()
{
	long nInputMode, nIsInverse;	// Pulsar 입력 모드 정보.

	// 설정된 Manual Pulsar 설정 값을 확인한 후, 입력 모드를 'CW/CCW' 모드로 설정합니다.
	if (cmmPlsrGetInMode (cmX1, &nInputMode, &nIsInverse ) == cmERR_NONE )
	{
		if ( nInputMode != cmIMODE_CWCCW )
		{
			cmmPlsrSetInMode (cmX1, cmIMODE_CWCCW, cmFALSE );
		}
	}
}
```

Visual Basic

```
Private Sub OnSetPlsrInMode ()

	Dim nInputMode As Long, nIsInverse As Long		' Pulsar 입력 모드 정보.

	' 설정된 Manual Pulsar 설정 값을 확인한 후, 입력 모드를 'CW/CCW' 모드로 설정합니다.
	If cmmPlsrGetInMode (cmX1, nInputMode, nIsInverse ) = cmERR_NONE Then
		If nInputMode <> cmIMODE_CWCCW Then
			Call cmmPlsrSetInMode (cmX1, cmIMODE_CWCCW, cmFALSE )
		End If
	End If

End Sub
```

Delphi

```
procedure OnSetPlsrInMode ();
var
	nInputMode, nIsInverse : LongInt		// Pulsar 입력 모드 정보.

begin
	// 설정된 Manual Pulsar 설정 값을 확인한 후, 입력 모드를 'CW/CCW' 모드로 설정합니다.
	if cmmPlsrGetInMode (cmX1, @nInputMode, @nIsInverse ) = cmERR_NONE then
	begin
		if nInputMode <> cmIMODE_CWCCW then
		begin
			cmmPlsrSetInMode (cmX1, cmIMODE_CWCCW, cmFALSE );
		end;
	end;
end;
```

NAME

cmmPlsrSetGain
cmmPlsrGetGain
- PA/PB 입력 대비, 출력 펄스 환경설정(環境設定)

INFORMATION

Manual Pulsar Motion

VC++/VB

BCB/Delphi/.NET

Level 5

다소 주의

함수의 내용이 약간의 난이도를 가지고 있습니다. 주의 깊게 읽어주시기 바랍니다.

SYNOPSIS

❒ VT_I4 cmmPlsrSetGain ([in] VT_I4 Axis, [in] VT_I4 GainFactor, [in] VT_I4 DivFactor)

❒ VT_I4 cmmPlsrGetGain ([in] VT_I4 Axis, [out] VT_PI4 GainFactor, [out] VT_PI4 DivFactor)

DESCRIPTION

cmmPlsrSetGain() 함수는 PA/PB 입력 펄스 대비 Command 출력 펄스 수의 비를 사용자가 임의로 조절할 수 있도록 하는 함수입니다.
cmmPlsrGetGain() 함수는 현재 설정된 Pulsar 출력 펄스 수 Gain 을 읽어들이는 함수입니다.
Manual Pulsar 출력은 PA/PB 로 입력되는 신호가 다음과 같이 3 단계의 회로를 거쳐서 출력되게 됩니다.

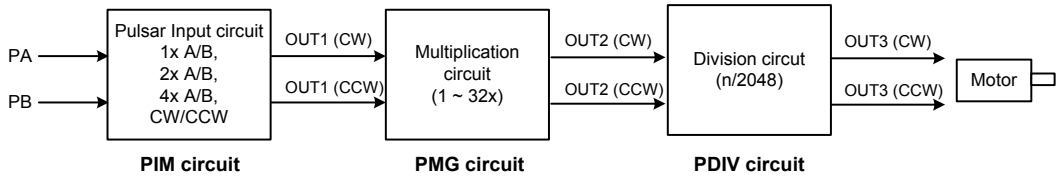


그림 10-1 PIM Circuit

그림에서 PIM 회로는 입력 신호를 분석하여 1 차 Command 출력 펄스 신호를 생성하는 회로입니다. PA/PB 입력 모드는 4 가지가 있을 수 있는데 이 것은 cmmPlsrSetInMode() 함수의 InputMode 매개 변수(媒介變數)에 의해서 결정됩니다. PIM 회로에서도 2 채배 또는 4 채배를 할 수 있으므로 2 배 또는 4 배의 수로 출력 펄스를 생성할 수 있습니다.
PMG 회로는 PIM 을 거쳐서 생성된 1 차 출력 펄스를 1~32 배수의 펄스로 재 생성하는 회로입니다.
PDIV 회로는 PMG 회로를 거쳐서 생성된 2 차 출력 펄스를 (n/2048) 을 곱하여 최종 출력 펄스를 생성하는 회로입니다. 여기서 n 은 cmmPlsrSetGain() 함수의 DivFactor 설정값을 의미하며, 1 ~ 2048 의 값을 설정할 수 있는데, 2048 보다 작은 수를 설정하면 결과적으로는 입력 펄스 대비 출력 펄스의 수를 줄이는 효과를 내므로 나누기 회로의 역할을 합니다.

PARAMETER

▶ Axis: 축번호. 축번호는 상수값으로 [cmX1] 부터 0 번째 축을 기준 축으로 임의의 축을 설정할 수 있습니다.

▶ GainFactor : cmmPlsrSetGain 함수의 인자, PMG 회로에 설정되는 사용자 정수로서 PIM 회로를 거쳐서 생성된 1 차 출력 펄스를 1~32 배수의 펄스로 재 생성하는 회로입니다. 이 값은 1 ~ 32 사이의 값이어야 합니다. 이 값의 초기 기본값은 1 입니다.

▶ GainFactor : cmmPlsrGetGain 함수의 인자이며, PMG 회로에 설정되는 사용자 정수를 반환합니다.

▶ DivFactor : cmmPlsrSetGain 함수의 인자, PDIV 회로에 설정되는 사용자 정수로서 PMG 회로를 거쳐서 생성된 2 차 출력 펄스에 (DivFactor/2048)가 곱해져서 최종 출력 펄스를 생성합니다. 이 값은 1 ~ 2048 의 값을 설정할 수 있는데

2048 을 제외한 나머지 값을 설정하면 결과적으로는 출력 펄스의 수를 줄이는 효과를 내므로 나누기 회로의 역할을 수행합니다. 이 값의 초기 기본값은 2048 입니다.

▶ DivFactor : cmmPlsrGetGain 함수의 인자이며, PDIV 회로에 설정되는 사용자 정수를 반환합니다.

REFERENCE

□ 다음의 표는 PA/PB 입력 펄스와 Command 출력 펄스 수의 관계를 규정하는 3 가지 항목 설정값에 따라서 입력 펄스와 출력 펄스 수의 비율이 어떻게 되는지를 몇 가지를 예시한 것입니다. 아래 표에서 Pulsar Input Mode 는 cmmPlsrSetInMode() 함수의 InputMode 매개 변수(媒介變數)를 의미합니다

Pulsar Input Mode	GainFactor	DivFactor	In:Out 펄스비
AB1X or CWCCW	1	2048	1 : 1
AB1X or CWCCW	10	2048	1 : 10
AB1X or CWCCW	32	2048	1 : 32
AB1X or CWCCW	1	1024	2 : 1
AB1X or CWCCW	1	512	4 : 1
AB1X or CWCCW	1	256	8 : 1
AB2X	1	2048	1 : 2
AB2X	10	2048	1 : 20
AB2X	32	2048	1 : 64
AB4X	1	2048	1 : 4
AB4X	10	2048	1 : 40
AB4X	32	2048	1 : 128

RETURN VALUE

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

EXAMPLE

C/C++

```
#include "Cmmsdk.h"
#include "CmmsdkDef.h"

void OnSetPlsrGain ()
{
	/* GainFactor 값과 DivFactor 값을 설정 하여 입력 펄스 대비 출력 펄스 수의 비를 설정합니다. */

	/* 출력 펄스를 입력 펄스의 2 배로 설정합니다.
	Pulsar 입력 모드 : CW/CCW, GainFactor : 2, DivFactor : 2048 로 설정하면 다음 수식에 의해
	출력 펄스 비 = 입력 펄스 모드 * GainFactor * ( DivFactor / 2048 ) = 1 * 2 * 20048/2048 = 2
	입력 펄스 : 출력 펄스 = 1 : 2 로 설정됩니다. */

	if (cmmPlsrSetInMode (cmX1, cmIMODE_CWCCW, cmFALSE ) == cmERR_NONE )
	{
		cmmPlsrSetGain (cmX1,		// 대상 축 설정
				2,		// GainFactor: PMG 회로에 설정되는 사용자 정수
				2048		// DivFactor: PDIV 회로에 설정되는 사용자 정수
				);
	}


	/* 출력 펄스를 입력 펄스의 0.25 배로 설정합니다.
	Pulsar 입력 모드 : AB1X, GainFactor : 1, DivFactor : 1024 로 설정하면 다음 수식에 의해
	출력 펄스 비 = 입력 펄스 모드 * GainFactor * ( DivFactor / 2048 ) = 1 * 1 * 1024/2048 = 0.5
	입력 펄스 : 출력 펄스 = 2 : 1 로 설정됩니다. */

	if (cmmPlsrSetInMode (cmX1, cmIMODE_CWCCW, cmFALSE ) == cmERR_NONE )
	{
		cmmPlsrSetGain (cmX1, 1, 1024 );
	}
```

```
}
```

Visual Basic

```
Private Sub OnSetPlsrGain ()

        ' GainFactor 값과 DivFactor 값을 설정 하여 입력 펄스 대비 출력 펄스 수의 비를 설정합니다.

        ' 출력 펄스를 입력 펄스의 4 배로 설정합니다.
        ' Pulsar 입력 모드 : AB2X, GainFactor : 2, DivFactor : 2048 로 설정하면 다음 수식에 의해
        ' 출력 펄스 비 = 입력 펄스 모드 * GainFactor * ( DivFactor / 2048 ) = 2 * 2 * 2048/2048 = 4
        ' 입력 펄스 : 출력 펄스 = 1 : 4 로 설정됩니다.

        If cmmPlsrSetInMode (cmX1, cmIMODE_AB2X, cmFALSE ) = cmERR_NONE Then
                Call cmmPlsrSetGain (cmX1, 2, 2048 )
        End If

        ' 출력 펄스를 입력 펄스의 0.25 배로 설정합니다.
        ' Pulsar 입력 모드 : AB2X, GainFactor : 1, DivFactor : 256 로 설정하면 다음 수식에 의해
        ' 출력 펄스 비 = 입력 펄스 모드 * GainFactor * ( DivFactor / 2048 ) = 2 * 1 * 256/2048 = 0.25
        ' 입력 펄스 : 출력 펄스 = 4 : 1 로 설정됩니다.

        If cmmPlsrSetInMode (cmX1, cmIMODE_AB2X, cmFALSE ) = cmERR_NONE Then
                Call cmmPlsrSetGain (cmX1, 1, 256 )
        End If

End Sub
```

Delphi

```
procedure OnSetPlsrGain ();
begin
        // GainFactor 값과 DivFactor 값을 설정 하여 입력 펄스 대비 출력 펄스 수의 비를 설정합니다.

        { 출력 펄스를 입력 펄스의 1 배로 설정합니다.
        Pulsar 입력 모드 : AB1X, GainFactor : 1, DivFactor : 2048 로 설정하면 다음 수식에 의해
                출력 펄스 비 = 입력 펄스 모드 * GainFactor * ( DivFactor / 2048 ) = 1 * 1 * 2048/2048 = 1
                입력 펄스 : 출력 펄스 = 1 : 1
        로 설정됩니다. }

        if cmmPlsrSetInMode (cmX1, cmIMODE_AB1X, cmFALSE ) = cmERR_NONE then
        begin
                cmmPlsrSetGain (cmX1, 1, 2048 );
        end


        { 출력 펄스를 입력 펄스의 10 배로 설정합니다.
        Pulsar 입력 모드 : AB2X, GainFactor : 5, DivFactor : 2048 로 설정하면 다음 수식에 의해
                출력 펄스 비 = 입력 펄스 모드 * GainFactor * ( DivFactor / 2048 ) = 2 * 5 * 2048/2048 = 10
                입력 펄스 : 출력 펄스 = 1 : 10
        로 설정됩니다. }

        if cmmPlsrSetInMode (cmX1, cmIMODE_AB2X, cmFALSE ) = cmERR_NONE then
        begin
                cmmPlsrSetGain (cmX1, 5, 2048 );
        end;

end;
```

NAME

cmmPlsrHomeMoveStart
- Pulsar Input 에 의한 원점 복귀(原點復歸) 이송

INFORMATION

Manual Pulsar Motion

VC++/VB

BCB/Delphi/.NET

Level 5

이송 함수

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

SYNOPSIS

❒ VT_I4 cmmPlsrHomeMoveStart ([in] VT_I4 Axis, [in] VT_I4 HomeType)

DESCRIPTION

이 함수는 Pulsar Input 에 의한 원점 복귀 작업을 수행합니다. 원점 복귀 모드(Home Type)에 따라 원점 복귀가 완료되거나 cmmSxStopEmg() 함수가 호출되면 모션을 종료합니다.

PARAMETER

▶ Axis: 축번호. 축번호는 상수값으로 [cmX1] 부터 0 번째 축을 기준 축으로 임의의 축을 설정할 수 있습니다.

▶ HomeType : Pulsar Input 에 의해 원점 복귀를 수행하는 모드를 설정합니다. 이 값은 다음 중 하나의 값이어야 합니다.

Value	Meaning
0	Command 카운터가 0 이 될때 원점복귀를 종료합니다.
1	ORG 신호가 ON 이 되면 원점복귀를 종료합니다.

REFERENCE

□ Manual Pulsar 모드 모션제어를 사용하고자 할 때에도 Manual Pulsar 모드 원점복귀 대신 일반적인 원점복귀를 수행할 수 있습니다.

RETURN VALUE

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

EXAMPLE

```
//* cmmPlsrMove / cmmPlsrMoveStart 예제를 참고하여 주시기 바랍니다.
```

NAME

cmmPlsrMove
cmmPlsrMoveStart
- Pulsar Input 에 의한
상대좌표이송(相對座標移送)

INFORMATION

Manual Pulsar Motion

VC++/VB

BCB/Delphi/.NET

Level 5

이송 함수

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

SYNOPSIS

❒ VT_I4 cmmPlsrMove ([in] VT_I4 Axis, [in] VT_R8 Distance, [in] VT_I4 IsBlocking)

❒ VT_I4 cmmPlsrMoveStart ([in] VT_I4 Axis, [in] VT_R8 Distance)

DESCRIPTION

이 함수는 Pulsar 신호 입력에 맞추어 지정한 거리(상대좌표)만큼 이동을 수행한 후에 Manual Pulsar 모드를 해제합니다. cmmPlsrMove() 함수는 모션이 완료되기 전까지 반환되지 않으며, cmmPlsrMoveStart() 함수는 모션을 시작시킨 후에 바로 반환됩니다.

PARAMETER

▶ Axis: 축번호. 축번호는 상수값으로 [cmX1] 부터 0 번째 축을 기준 축으로 임의의 축을 설정할 수 있습니다.

▶ Distance : 이동할 거리를 지정합니다. 지정한 거리만큼 이동한 후 Manual Pulsar 모드는 자동해제됩니다. 거리의 단위는 "Unit distance"에 의해 정의되는 논리적 거리입니다.

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

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

REFERENCE

☐ Manual Pulsar 모드 모션을 중지하기 위해서는 cmmSxStopEmg() 함수를 사용하십시오. cmmSxStopEmg() 함수가 호출되면 Manual Pulsar 모드는 해제됩니다.

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
cmERR_NONE	수행 성공

EXAMPLE

C/C++

```
#include "Cmmsdk.h"
#include "CmmsdkDef.h"

void OnSetPlsrConfig ()
{
	// Manual Pulsar 입력 모드 설정 및 입력 펄스와 출력 펄스 비를 1:2 로 설정합니다.
	cmmPlsrSetInMode (cmX1, cmIMODE_CWCCW, cmFALSE );
	cmmPlsrSetGain (cmX1, 2, 2048 );

	/* PA/PB 신호 입력 회로에 노이즈 필터 기능을 적용합니다.
	 노이즈 필터 기능이 활성화 되면 3.25MHz 이상의 주파수를 가지는 펄스는 노이즈로 처리됩니다.*/
	cmmCfgSetFilterAB (cmX1, cmAB_PULSAR, cmTRUE );
}


void OnPulsarMove ()
{
	/* 속도 환경을 설정합니다. 작업 속도는 출력 펄스의 폭을 결정하며,
	일반적으로 사용하는 서보 드라이버의 최대 속도로 설정하시면 됩니다. */
	cmmCfgSetSpeedPattern (cmX1,  // 대상 축을 설정합니다.
				cmSMODE_S,	// Pulsar 구동에서 속도 모드는 무시됩니다.
				655350,		// 작업 속도는 출력 펄스의 폭을 결정하며,
						// 일반적으로 서보드라이버의 최대 속도로 설정하시면 됩니다.
				10000,		// Pulsar 구동에서 가속도는 무시됩니다.
				10000		// Pulsar 구동에서 감속도는 무시됩니다.
				);


	/* Manual Pulsar 입력에 맞춰서 지정한 거리만큼 이송을 수행한 후
	Manual Pulsar 모드를 해제 합니다. Pulsar 이송 함수가 수행되면
	cmmStReadMotionState 함수의 모션 상태 반환 값이 8 ( cmMST_WAIT_PLSR ) 이 됩니다. */

	// 지정한 거리만큼 상대좌표 이송을 수행합니다. 모션이 완료되기 전까지 반환되지 않습니다.
	cmmPlsrMove (cmX1, 10000, cmFALSE );

	// cmmPlsrMoveStart()함수는 모션을 시작시킨 후에 바로 반환됩니다.
}
```

Visual Basic

```
'// * 이 함수는 설정된 거리만큼 이동하는 함수입니다.
Private Sub DoMotion()

	'// Manual Pulsar 입력 모드를 채배 방식과 Pulsar 입력 신호에 의해
	'// 결정되는 방향(Direction)을 설정합니다.
	Call cmmPlsrSetInMode(cmX1, cmIMODE_AB4X, cmFALSE)

	'// 실제 Manual Pulsar 입력에 맞추어 지정한 거리(상대좌표) 만큼 이동을
	'// 수행한후 Manual Pulsar 모드를 해제합니다.
	'// cmmPlsrMove() 함수는 모션이 완료되기 전까지 반환되지 않으며,
	'// cmmPlsrMoveStart() 함수는 모션을 시작시킨 후에 바로 반환됩니다.
	Call cmmPlsrMove(cmX1,  10000, cmTRUE)
End Sub
```

Delphi

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

procedure OnCreate();
var
        g_nAxis : LongInt;
begin

        // Load CMMSDK(DLL) Library
        if ( cmmGnDeviceLoad(cmTRUE,@g_nAxis) <> cmERR_NONE ) then
        begin
                // 마지막에 발생한 에러를 화면에 표시합니다.
                // 함수 인자로는 Form 의 Handle 이 전달됩니다.
                cmmErrShowLast(Form1.Handle);
                exit;
        end
end;

// * 기본 속도를 설정하는 함수 입니다.
procedure btnSetSpeedClick();
var
   fAccelSpeed : Double;
   fDecelSpeed : Double;
   fWorkSpeed : Double;
   nSMODE : LongInt;

begin

      fAccelSpeed :=50000;
       fDecelSpeed :=50000;
       fWorkSpeed := 30000;
       nSMODE := cmSMODE_S;

       // 설정된 기준 속도를 실제 SDK 함수에 전달합니다.

       cmmCfgSetSpeedPattern(
       cmX1,           // 활성화 된 축을 선택합니다.
       nSMODE,         // 가감속이 없는 모드와 선형 가감속, S-CURVE 가감속을 모드 중 하나를 설정합니다.
       fWorkSpeed,     // 작업 속도를 설정합니다.
       fAccelSpeed,    // 가속도를 설정합니다.
       fDecelSpeed);   // 감속도를 설정합니다.

end;

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

Procedure DoMotion();
begin

        // 설정된 기준속도에 비율로서 실제 모션 동작속도를 결정합니다.
        cmmSxSetSpeedRatio(cmX1, cmSMODE_KEEP,100.0, 100.0, 100.0);

        // Manual Pulsar 입력 모드를 채배 방식과 Pulsar 입력 신호에 의해
        // 결정되는 방향(Direction)을 설정합니다.
        cmmPlsrSetInMode(cmX1, cmIMODE_AB4X, cmFALSE);

        // 실제 Manual Pulsar 입력에 맞추어 지정한 거리(상대좌표) 만큼 이동을
        // 수행한후 Manual Pulsar 모드를 해제합니다.
        // cmmPlsrMove() 함수는 모션이 완료되기 전까지 반환되지 않으며,
        // cmmPlsrMoveStart() 함수는 모션을 시작시킨 후에 바로 반환됩니다.

        cmmPlsrMove
        (
        cmX1,
        10000,   // '-' 연산자에 주의
```

```
        cmFALSE
        );
end;

procedure btnStopClick();
begin

  // 함수의 인자에 대해서 설명 드리겠습니다.
  // 정지(停止) 함수의 원형은 cmmSxStop([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.
  // cmFALSE 시에는 윈도우 메시지를 함수 내부에서
  //  처리해주게 됩니다.
  // 보다 자세한 설명은 매뉴얼을 참고해주시기 바랍니다.

        cmmSxStop(cmX1, cmTRUE, cmFALSE);
end;
```

NAME

cmmPlsrMoveTo
cmmPlsrMoveToStart
- Pulsar Input 에 의한
절대좌표이송(絶對座標移送)

INFORMATION

Manual Pulsar Motion

VC++/VB

BCB/Delphi/.NET

Level 5

이송 함수

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

SYNOPSIS

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

❒ VT_I4 cmmPlsrMoveToStart ([in] VT_I4 Axis, [in] VT_R8 Position)

DESCRIPTION

이 함수는 Pulsar 신호 입력에 맞추어 지정한 위치(절대좌표)로 이동을 수행합니다. 지정한 위치로 이동한 후에는 Manual Pulsar 모드가 자동으로 해제됩니다. cmmPlsrMoveTo() 함수는 모션이 완료되기 전까지 반환되지 않으며, cmmPlsrMoveToStart() 함수는 모션을 시작시킨 후에 바로 반환됩니다.

PARAMETER

▶ Axis: 축번호. 축번호는 상수값으로 [cmX1] 부터 0 번째 축을 기준 축으로 임의의 축을 설정할 수 있습니다.

▶ Position : 이동할 목표 위치(절대좌표값)를 지정합니다. 지정한 위치로 이동한 후에는 Manual Pulsar 모드가 자동으로 해제됩니다. 좌표의 단위는 "Unit distance"에 의해 정의되는 논리적 거리입니다.

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

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

REFERENCE

☐ Manual Pulsar 모드 모션을 중지하기 위해서는 cmmSxStopEmg() 함수를 사용하십시오. cmmSxStopEmg() 함수가 호출되면 Manual Pulsar 모드는 해제됩니다.

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
cmERR_NONE	수행 성공

EXAMPLE

```
//* cmmPlsrMove / cmmPlsrMoveStart 예제를 참고하여 주시기 바랍니다.
```

NAME

cmmPlsrVMoveStart
- Pulsar Input 에 의한
연속속도이송(連續速度移送)

INFORMATION

Manual Pulsar Motion

VC++/VB

BCB/Delphi/.NET

Level 5

이송 함수

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

SYNOPSIS

❒ VT_I4 cmmPlsrVMoveStart ([in] VT_I4 Axis)

DESCRIPTION

이 함수는 Stop 함수가 호출될 때까지 Pulsar 신호 입력에 맞추어 계속적인 모션을 수행합니다. 모션의 속도는 Pulsar 신호의 주파수에 따라 결정됩니다.

PARAMETER

▶ Axis : 축 번호. 축번호는 상수값으로 [cmX1] 부터 0 번째 축을 기준 축으로 임의의 축을 설정할 수 있습니다.

RETURN VALUE

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

EXAMPLE

```
//* cmmPlsrMove / cmmPlsrMoveStart 예제를 참고하여 주시기 바랍니다.
```

NAME

cmmPlsrIsActive
- Pulsar 입력 상태(狀態) 확인(確認)

INFORMATION

Manual Pulsar Motion

VC++/VB

BCB/Delphi/.NET

Level 5

위험 요소 없음

SYNOPSIS

❒ VT_I4 cmmPlsrIsActive ([in] VT_I4 Axis, [out] VT_PI4 IsActive)

DESCRIPTION

이 함수는 Pulsar 신호의 현재 입력 상태를 확인(確認)합니다.

PARAMETER

▶ Axis : 축 번호. 축번호는 상수값으로 [cmX1] 부터 0 번째 축을 기준 축으로 임의의 축을 설정할 수 있습니다.

▶ IsActive : 현재 Pulsar 신호 입력 상태를 반환합니다

Value	Meaning
cmFALSE	Pulsar 신호 입력(Active) 상태가 아닙니다.
cmTRUE	Pulsar 신호 입력(Active) 상태입니다.

RETURN VALUE

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

10.2 외부스위치(External switch)에 의한 모션제어

이 단원에서는 External Switch Operation 모드에 관련된 함수들을 소개합니다. 여기서 External Switch는 +DR과 −DR 신호를 의미합니다. External Switch Operation 모드는 +DR 또는 −DR 신호가 ON상태인 경우에만 모션을 구동하고 OFF인 상태에서는 모션을 정지(停止)하는 기능을 말합니다. 이 때의 속도 패턴cmmCfgSetSpeedPattern() 함수에 의해 설정된 속도 패턴을 사용합니다. +DR신호는 정방향으로 모션을 구동시키고 −DR 신호는 역방향으로 모션을 구동시킵니다.

+DR 또는 −DR 신호와 모션 구동의 관계는 다음 그림과 같습니다.

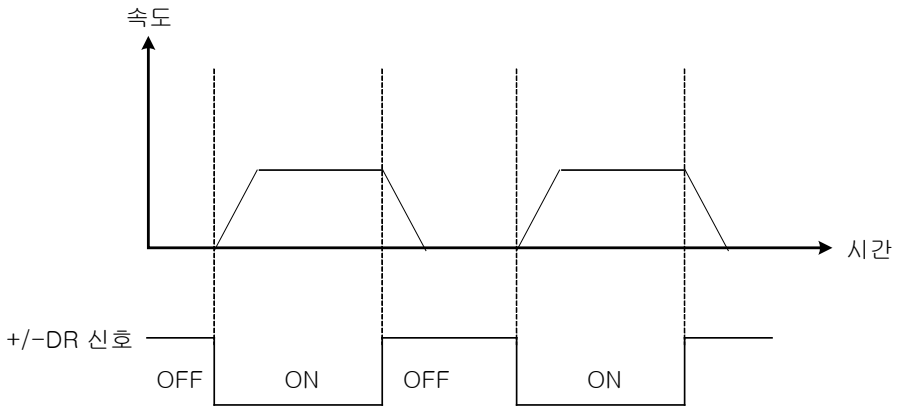


그림 10-2 External Switch Operation 의 모션 동작 관계

10.2.1 함수 요약

External Switch Operation 모드 모션에 관련된 함수들은 다음과 같습니다.

Summary of Functions
❒ VT_I4 cmmExVMoveStart ([in] VT_I4 Axis) 대상(對象) 모션 채널에 대해서, External switch operation 모드하에 Velocity-Move(速度移送)를 시작합니다.
❒ VT_I4 cmmExMoveStart ([in] VT_I4 Axis, [in] VT_R8 Distance) 대상(對象) 모션 채널에 대해서, External switch operation 모드하에 상대좌표(相對座標) 위치결정(位置決定) 모션을 시작합니다.
❒ VT_I4 cmmExMoveToStart ([in] VT_I4 Axis, [in] VT_R8 Position) 대상(對象) 모션 채널에 대해서, External switch operation 모드하에 절대좌표(絶對座標) 위치결정(位置決定) 모션을 시작합니다.

10.2.2 함수 설명

NAME

cmmExVMoveStart
– 외부(外部) 스위치에 의한 속도 이송(速度移送)

INFORMATION

External Switch
VC++/VB
BCB/Delphi/.NET
Level 5
😐 이송 함수
실제 이송이 진행되는 함수이므로, 사전에 반드시 안전을 확인(確認)합니다.

SYNOPSIS

❒ VT_I4 cmmExVMoveStart ([in] VT_I4 Axis)

DESCRIPTION

+/-DR 신호가 ON 상태가 되면 Velocity Move 모션을 수행합니다. +/-DR 신호가 OFF 가 되면 모션을 정지(停止)합니다. 이 때의 구동 방향은 +DR 신호와 −DR 신호에 의해 결정됩니다. +DR 신호가 ON 되면 (+)방향으로 이동하고, -DR 신호가 ON 되면 (-)방향으로 이동합니다. 단, +DR 과 −DR 신호가 동시에 ON 이 되면 오동작할 수 있습니다.

PARAMETER

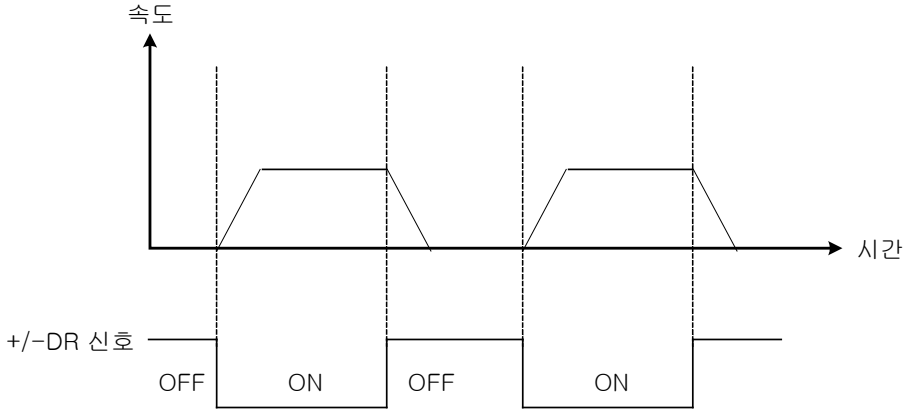
▶ Axis : 축번호. 축 번호는 0 부터 시작합니다.

RETURN VALUE

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

REFERENCE

□ 속도 모드를 Constant Speed Mode 로 지정한 경우에는 가속 구간이 없이 작업속도로 모션을 시작합니다.
□ +DR 또는 −DR 신호와 모션 구동의 관계는 다음 그림과 같습니다.



EXAMPLE

C/C++

```
#include "Cmmsdk.h"
#include "CmmsdkDef.h"

/************************************************************
* OnProgramInitial : 이 함수는 가상의 함수로서 프로그램 초기화 루틴이
* 적용되는 부분을 의미합니다.

************************************************************/
void OnProgramInitial()
{

	long m_nNumAxes;

	cmmLoadDll();
	if(cmmGnDeviceLoad(cmTRUE, &m_nNumAxes) != cmERR_NONE)
	{
		//Handle 은 사용자가 생성한 폼의 핸들 값입니다.
		cmmErrShowLast(Handle);
		return;
	}
	//X1 축의 속도패턴과 기본속도를  설정합니다.
	cmmCfgSetSpeedPattern(cmX1, cmSMODE_S, 10000, 100000, 100000);
}

void OnExVMove()
{
	//X1 축의 작업속도를 기본속도의 100% 가속도를 50% 감속도를
	//50%로 설정합니다.
	cmmSxSetSpeedRatio(cmX1, cmSMODE_S, 100, 50, 50);

	////////////////////////////////////////////////////////
	// Start V-Move by External Switch (-/+ DR) : Stop 함수가 호출될
	// 때까지 -/+DR 신호의 ON/OFF 상태에 의해 이동과 정지(停止)를 수행함
	if(cmmExVMoveStart (cmX1) != cmERR_NONE){
		cmmErrShowLast(Handle);//에러처리
	}
}

/************************************************************
* OnStop() : "Stop"명령시에 호출되는 가상의 함수
************************************************************/
void OnStop()
{
	cmmSxStopEmg(cmX1);
}
```

Visual Basic

```
Private Sub OnExVMove()
'//X1 축의 작업속도를 기본속도의 100% 가속도를 50% 감속도를
'//50%로 설정합니다.
Call cmmSxSetSpeedRatio(cmX1, cmSMODE_S, 100, 50, 50)

'////////////////////////////////////////////////////////
'// Start V-Move by External Switch (-/+ DR) : Stop 함수가 호출될
'// 때까지 -/+DR 신호의 ON/OFF 상태에 의해 이동과 정지(停止)를 수행함

If cmmExVMoveStart(cmX1) <> cmERR_NONE Then
   Call cmmErrShowLast(thisform.Hwnd) '에러처리
End If
End Sub

Private Sub OnStop()

	Call cmmSxStopEmg(cmX1)

End Sub
```

Delphi

```
//*************************************************************
// OnProgramInitial : 이 함수는 가상의 함수로서 프로그램 초기화 루틴이
// 적용되는 부분을 의미합니다.

//*************************************************************/

Const Handle = 1;

procedure OnProgramInitial();

var
        m_nNumAxes : LongInt;

begin

        if(cmmGnDeviceLoad(cmTRUE, @m_nNumAxes) <> cmERR_NONE) then
        begin
                //Handle 은 사용자가 생성한 폼의 핸들 값입니다.
                cmmErrShowLast(Handle);
                exit;
        end;
        //X1 축의 속도패턴과 기본속도를  설정합니다.
        cmmCfgSetSpeedPattern(cmX1, cmSMODE_S, 10000, 100000, 100000);
end;

procedure OnExVMove();

begin
        //X1 축의 작업속도를 기본속도의 100% 가속도를 50% 감속도를
        //50%로 설정합니다.
        cmmSxSetSpeedRatio(cmX1, cmSMODE_S, 100, 50, 50);

        ////////////////////////////////////////////////////////////
        // Start V-Move by External Switch (-/+ DR) : Stop 함수가 호출될
        // 때까지 -/+DR 신호의 ON/OFF 상태에 의해 이동과 정지(停止)를 수행함
        if(cmmExVMoveStart (cmX1) <> cmERR_NONE) then
        begin
                cmmErrShowLast(Handle);//에러처리
        end;
end;

procedure OnStop();
begin

        cmmSxStopEmg(cmX1);

end;
```

NAME

cmmExMoveStart
외부(外部) 스위치에 의한
상대좌표이송(相對座標移送)

INFORMATION

External Switch

VC++/VB

BCB/Delphi/.NET

Level 5

이송 함수

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

SYNOPSIS

❒ VT_I4 cmmExMoveStart ([in] VT_I4 Axis, [in] VT_R8 Distance)

DESCRIPTION

External switch operation 모드하에 Relative In-Position(상대좌표 위치결정) 모션을 시작합니다. +/-DR 신호가 ON 상태일 때에만 모터를 구동합니다. +DR 신호가 ON 이 되면 정방향으로 회전하고 −DR 신호가 ON 이 되면 역방향으로 회전합니다.
지정한 상대좌표로의 이동이 완료되면 +/-DR 신호와 관계없이 모션을 종료합니다.

PARAMETER

▶ Axis: 축번호. 축번호는 상수값으로 [cmX1] 부터 0 번째 축을 기준 축으로 임의의 축을 설정할 수 있습니다.

▶ 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.

RETURN VALUE

Value	Meaning
음수	수행 실패. 자세한 내용은 ‘에러처리’ 편을 참고합니다
cmERR_NONE	수행 성공

REFERENCE

□ 논리적 거리 단위는 cmmCfgSetUnitDist() 함수에 의해 결정됩니다.

□ cmmSxIsDone() 함수나 cmmSxWaitDone() 함수를 사용하여 모션의 완료를 확인(確認)할 수 있습니다.

EXAMPLE

C/C++

```
#include "Cmmsdk.h"
#include "CmmsdkDef.h"

/************************************************************
* OnProgramInitial : 이 함수는 가상의 함수로서 프로그램 초기화 루틴이
* 적용되는 부분을 의미합니다.
************************************************************/
void OnProgramInitial()
{
```

```
	long m_nNumAxes;

	cmmLoadDll();
	if(cmmGnDeviceLoad(cmTRUE, &m_nNumAxes) != cmERR_NONE)
	{
		//Handle 은 사용자가 생성한 폼의 핸들 값입니다.
		cmmErrShowLast(Handle);
		return;
	}
	//X1 축의 속도패턴과 기본속도를  설정합니다.
	cmmCfgSetSpeedPattern(cmX1, cmSMODE_S, 10000, 100000, 100000);
}

void OnExVMove()
{
	//X1 축의 작업속도를 기본속도의 100% 가속도를 50% 감속도를
	//50%로 설정합니다.
	cmmSxSetSpeedRatio(cmX1, cmSMODE_S, 100, 50, 50);

	////////////////////////////////////////////////////////////
	// Start Move by External Switch (-/+ DR)
	if(cmmExMoveStart (cmX1, 10000) != cmERR_NONE){
		cmmErrShowLast(Handle);// 에러 처리
	}
	// 종료될때까지 기다린다. 지정한 거리만큼 이동이 완료되면 자동종료 //
	if(cmmSxWaitDone(cmX1, cmFALSE) != cmFALSE){
		cmmErrShowLast(Handle);// 에러 처리
	}
}

/************************************************************
* OnStop() : "Stop"명령시에 호출되는 가상의 함수
************************************************************/
void OnStop()
{
	cmmSxStopEmg(cmX1);
}
```

Visual Basic

```
Private Sub OnExVMove()

	'//X1 축의 작업속도를 기본속도의 100% 가속도를 50% 감속도를
	'//50%로 설정합니다.
	Call cmmSxSetSpeedRatio(cmX1, cmSMODE_S, 100, 50, 50)

	'////////////////////////////////////////////////////////////
	'// Start Move by External Switch (-/+ DR)
	If cmmExMoveStart(cmX1, 10000) <> cmERR_NONE Then
	Call cmmErrShowLast(thisform.Hwnd) '에러처리
	End If

	'// 종료될때까지 기다린다. 지정한 거리만큼 이동이 완료되면 자동종료 //
	If cmmSxWaitdone(cmX1, cmFALSE) <> cmFALSE Then
	   Call cmmErrShowLast(thisform.Hwnd) '에러처리
	End If

End Sub

Private Sub OnStop()

	Call cmmSxStopEmg(cmX1)

End Sub
```

Delphi

```
//************************************************************
// OnProgramInitial : 이 함수는 가상의 함수로서 프로그램 초기화 루틴이
// 적용되는 부분을 의미합니다.

//************************************************************/

Const Handle = 1;

procedure OnProgramInitial();

var
	m_nNumAxes : LongInt;

begin

	if(cmmGnDeviceLoad(cmTRUE, @m_nNumAxes) <> cmERR_NONE) then
	begin
		//Handle 은 사용자가 생성한 폼의 핸들 값입니다.
		cmmErrShowLast(Handle);
		exit;
	end;
	//X1 축의 속도패턴과 기본속도를  설정합니다.
	cmmCfgSetSpeedPattern(cmX1, cmSMODE_S, 10000, 100000, 100000);
end;

procedure OnExVMove();

begin
	//X1 축의 작업속도를 기본속도의 100% 가속도를 50% 감속도를
	//50%로 설정합니다.
	cmmSxSetSpeedRatio(cmX1, cmSMODE_S, 100, 50, 50);

	////////////////////////////////////////////////////////
	// Start Move by External Switch (-/+ DR)
	if(cmmExMoveStart (cmX1, 10000) <> cmERR_NONE) then
	begin
		cmmErrShowLast(Handle);// 에러 처리
	end;
	// 종료될때까지 기다린다. 지정한 거리만큼 이동이 완료되면 자동종료 //
	if(cmmSxWaitDone(cmX1, cmFALSE) <> cmFALSE) then
	begin
		cmmErrShowLast(Handle);// 에러 처리
	end;

end;

procedure OnStop();
begin

	cmmSxStopEmg(cmX1);

end;
```

NAME

cmmExMoveToStart
외부(外部) 스위치에 의한
절대좌표이송(絶對座標移送)

INFORMATION

External Switch

VC++/VB

BCB/Delphi/.NET

Level 5

이송 함수

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

SYNOPSIS

❒ VT_I4 cmmExMoveToStart ([in] VT_I4 Axis, [in] VT_R8 Position)

DESCRIPTION

External switch operation 모드하에 절대좌표 위치결정 모션을 시작합니다. +/-DR 신호가 ON 상태일 때에만 모터를 구동합니다. +DR 신호가 ON 이 되면 정방향으로 회전하고 −DR 신호가 ON 이 되면 역방향으로 회전합니다.
지정한 절대좌표로의 이동이 완료되면 +/-DR 신호와 관계없이 모션을 종료합니다.

PARAMETER

▶ Axis: 축번호. 축번호는 상수값으로 [cmX1] 부터 0 번째 축을 기준 축으로 임의의 축을 설정할 수 있습니다.

▶ Position : 이동할 목표 위치(절대좌표값)를 지정합니다. 이 값은 현재의 위치에 대한 상대 좌표이며, 거리의 단위는 논리적 거리(Logic distance) 단위를 사용합니다.

RETURN VALUE

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

REFERENCE

□ 논리적 거리 단위는 cmmCfgSetUnitDist() 함수에 의해 결정됩니다.

□ cmmSxIsDone() 함수나 cmmSxWaitDone() 함수를 사용하여 모션의 완료를 확인(確認)할 수 있습니다.

EXAMPLE

본 예제는 X1 축을 절대좌표 10000 위치로 이동한 후 다시 절대좌표 0 위치로 이동합니다. 이때 이동의 시작은 -/+ DR 신호에 의하여 이루어집니다.

C/C++

```
#include "Cmmsdk.h"
#include "CmmsdkDef.h"

/*************************************************************
* OnProgramInitial : 이 함수는 가상의 함수로서 프로그램 초기화 루틴이
* 적용되는 부분을 의미합니다.

*************************************************************/
void OnProgramInitial()
```

```
{
	long m_nNumAxes;

	cmmLoadDll();
	if(cmmGnDeviceLoad(cmTRUE, &m_nNumAxes) != cmERR_NONE)
	{
		//Handle 은 사용자가 생성한 폼의 핸들 값입니다.
		cmmErrShowLast(Handle);
		return;
	}
	//X1 축의 속도패턴과 기본속도를  설정합니다.
	cmmCfgSetSpeedPattern(cmX1, cmSMODE_S, 10000, 100000, 100000);
}

void OnExVMove()
{
	//X1 축의 작업속도를 기본속도의 100% 가속도를 50% 감속도를
	//50%로 설정합니다.
	cmmSxSetSpeedRatio(cmX1, cmSMODE_S, 100, 50, 50);

	////////////////////////////////////////////////////////
	// Move to 10000 position (단, 이동은 +DR 신호가 ON 상태일때만 이루
	// 어짐)
	if(cmmExMoveToStart (cmX1, 10000) != cmERR_NONE){
		cmmErrShowLast(Handle);// 에러 처리
	}

	// 종료될때까지 기다린다. 지정한 위치로의 이동이 완료되면 자동종료 //
	if(cmmSxWaitDone(cmX1, cmFALSE) != cmERR_NONE){
		cmmErrShowLast(Handle);// 에러 처리
	}

	////////////////////////////////////////////////////////
	// Move to 10000 position (단, 이동은 -DR 신호가 ON 상태일때만 이루
	// 어짐)
	if(cmmExMoveToStart (cmX1, 0) != cmERR_NONE){
		cmmErrShowLast(Handle);// 에러 처리
	}

	// 종료될때까지 기다린다. 지정한 위치로의 이동이 완료되면 자동종료 //
	if(cmmSxWaitDone(cmX1, FALSE) != cmERR_NONE){
		cmmErrShowLast(Handle);// 에러 처리
	}
}

/************************************************************
* OnStop() : "Stop"명령시에 호출되는 가상의 함수
************************************************************/
void OnStop()
{
	cmmSxStopEmg(cmX1);
}
```

Visual Basic

```
' ************************************************************
' OnProgramInitial : 이 함수는 가상의 함수로서 프로그램 초기화 루틴이
' 적용되는 부분을 의미합니다.

' ************************************************************/

Private Sub OnProgramInitial (void)

Dim m_nNumAxes As Long
Dim IRetVal As Long

IRetVal = cmmGnDeviceLoad(cmTRUE, m_nNumAxes)
```

```
If lRetVal <> cmERR_NONE Then
        Call cmmErrShowLast(Handle)
End If

Call cmmCfgSetSpeedPattern(cmX1, cmSMODE_S, 10000, 100000, 100000)

End Sub

Private Sub OnExVMove (void)

        Dim lRetVal As Long

        ' X1 축의 작업속도를 기본속도의 100% 가속도를 50% 감속도를
        ' 50%로 설정합니다.
        Call cmmSxSetSpeedRatio(cmX1, cmSMODE_S, 100, 50, 50)

        ''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
        ' Move to 10000 position (단, 이동은 +DR 신호가 ON 상태일때만 이루어짐)
        lRetVal = cmmExMoveToStart (cmX1, 10000)

        If lRetVal <> cmERR_NONE Then
                Call cmmErrShowLast(Handle) ' 에러 처리
        End If

        ' 종료될때까지 기다린다. 지정한 위치로의 이동이 완료되면 자동종료
        lRetVal = cmmSxWaitDone(cmX1, cmFALSE)

        If lRetVal <> cmERR_NONE Then
                Call cmmErrShowLast(Handle) ' 에러 처리
        End If

        ''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
        ' Move to 10000 position (단, 이동은 -DR 신호가 ON 상태일때만 이루어짐)
        lRetVal = cmmExMoveToStart (cmX1, 0)

        If lRetVal <> cmERR_NONE Then
                Call cmmErrShowLast(Handle) ' 에러 처리
        End If

        ' 종료될때까지 기다린다. 지정한 위치로의 이동이 완료되면 자동종료
        lRetVal = cmmSxWaitDone(cmX1, cmFALSE)

        If lRetVal <> cmERR_NONE Then
                Call cmmErrShowLast(Handle) ' 에러 처리
        End If

End Sub

Private Sub OnStop(void)

        Call cmmSxStopEmg(cmX1)

End Sub
```

```
Delphi

//***************************************************************
// OnProgramInitial : 이 함수는 가상의 함수로서 프로그램 초기화 루틴이
// 적용되는 부분을 의미합니다.

//***************************************************************/

Const Handle = 1;

procedure OnProgramInitial();

var
        m_nNumAxes : LongInt;

begin
```

```
	if(cmmGnDeviceLoad(cmTRUE, @m_nNumAxes) <> cmERR_NONE) then
	begin
		//Handle 은 사용자가 생성한 폼의 핸들 값입니다.
		cmmErrShowLast(Handle);
		exit;
	end;
	//X1 축의 속도패턴과 기본속도를  설정합니다.
	cmmCfgSetSpeedPattern(cmX1, cmSMODE_S, 10000, 100000, 100000);
end;

procedure OnExVMove();

begin
	//X1 축의 작업속도를 기본속도의 100% 가속도를 50% 감속도를
	//50%로 설정합니다.
	cmmSxSetSpeedRatio(cmX1, cmSMODE_S, 100, 50, 50);

	////////////////////////////////////////////////////////
	// Move to 10000 position (단, 이동은 +DR 신호가 ON 상태일때만 이루어짐)
	if(cmmExMoveToStart (cmX1, 10000) <> cmERR_NONE) then
	begin
		cmmErrShowLast(Handle);// 에러 처리
	end;

	// 종료될때까지 기다린다. 지정한 위치로의 이동이 완료되면 자동종료 //
	if(cmmSxWaitDone(cmX1, cmFALSE) <> cmERR_NONE) then
	 begin
		cmmErrShowLast(Handle);// 에러 처리
	end;

	////////////////////////////////////////////////////////
	// Move to 10000 position (단, 이동은 -DR 신호가 ON 상태일때만 이루어짐)
	if(cmmExMoveToStart (cmX1, 0) <> cmERR_NONE) then
	 begin
		cmmErrShowLast(Handle);// 에러 처리
	end;

	// 종료될때까지 기다린다. 지정한 위치로의 이동이 완료되면 자동종료 //
	if(cmmSxWaitDone(cmX1, cmFALSE) <> cmERR_NONE) then
	 begin
		cmmErrShowLast(Handle);// 에러 처리
	end;

end;

procedure OnStop();
begin

	cmmSxStopEmg(cmX1);

end;
```

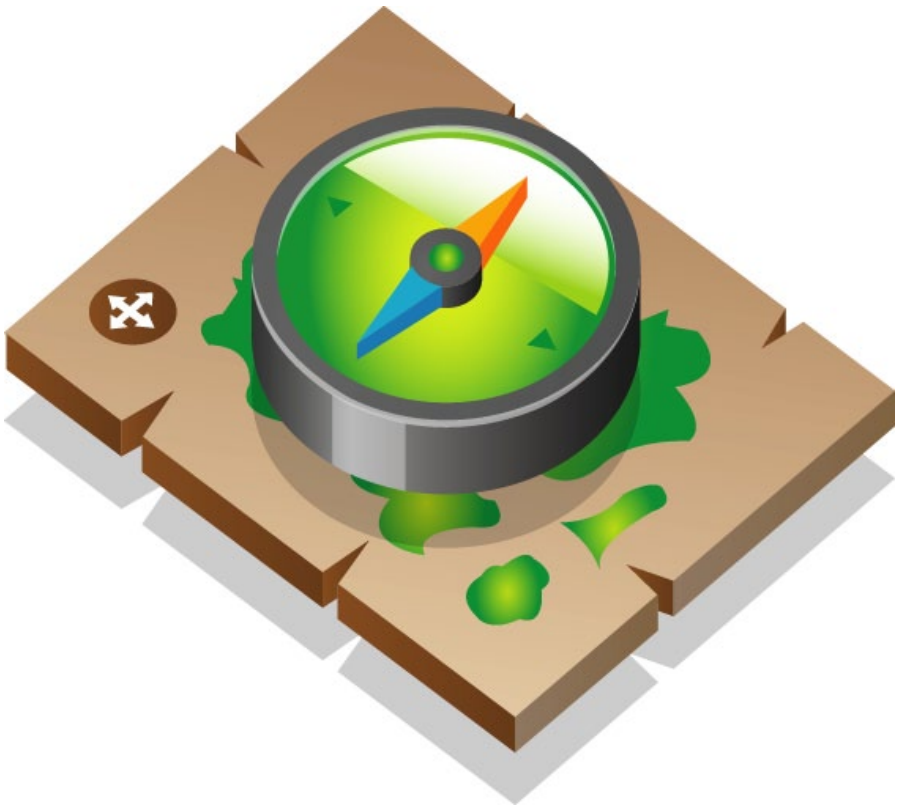
Chapter

11

Monitoring Motion Status

모션제어의 상태를 확인(確認)하는 역할은 고객(顧客)여러분들께서 개발하시는 응용프로그램의 필수 요건 중에 하나입니다. CMMSDK 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 감시 함수들은 모션의 상태를 감시하는데 필요한 함수들을 그룹화한 것입니다. 상태(狀態) 감시에는 모션의 감시에는 모션의 속도, 위치 등을 확인(確認)하는 것을 포함하며 이외에도 현재 모션이 진행되고 있는지를 진행되고 있는지를 확인(確認)하고, 현재 진행되고 있는 모션의 단계 및 각 I/O 상태들을 점검하는 기능도 점검하는 기능도 포함합니다.



11 상태감시 편

11.1 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.

11.1.1 함수 요약

상태 감시 및 제어 함수에 관련된 함수들은 다음과 같습니다.

Summary of Functions
❒ VT_I4 cmmStSetCount ([in] VT_I4 Axis, [in] VT_I4 Target, [in] VT_I4 Count) 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 cmmStGetCount ([in] VT_I4 Axis, [in] VT_I4 Source, [out] VT_PI4 Count) 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 cmmStSetPosition ([in] VT_I4 Axis, [in] VT_I4 Target, [in] VT_R8 Position) 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 cmmStGetPosition ([in] VT_I4 Axis, [in] VT_I4 Source, [out] VT_PR8 Position) 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 cmmStGetSpeed ([in] VT_I4 Axis, [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 cmmStReadMotionState ([in] VT_I4 Axis, [out] VT_PI4 MotStates) 대상(對象) 모션 채널에 대해서, 현재의 모션 동작 상태를 반환(返還)합니다.
❒ VT_I4 cmmStReadMioStatuses ([in] VT_I4 Axis, [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 cmmStGetMstString ([in] VT_I4 MstCode, [out] VT_STR Buffer, [in] VT_I4 BufferLen) 대상(對象) 모션 채널에 대해서, 현재 모션 동작 상태와 관련된 문자열을 반환합니다. 이 문자열은 라이브러리에서 생성된 문자열이므로, 사용자의 의도에 의해 결정된 문자열은 아닙니다.

11.1.2 함수 설명

NAME

cmmStSetCount
– 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 cmmStSetCount ([in] VT_I4 Axis, [in] VT_I4 Target, [in] VT_I4 Count)

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 cmmStSetPosition() 함수와 동일합니다.

PARAMETER

▶ Axis: 축번호. 축번호는 상수값으로 [cmX1] 부터 0 번째 축을 기준 축으로 임의의 축을 설정할 수 있습니다.

▶ Target : 설정할 카운터 번호. 이 값은 다음의 4 가지 값중의 하나이어야 합니다.

Value	Meaning
0 또는 cmCNT_COMM	Command Counter
1 또는 cmCNT_FEED	Feedback Counter
2 또는 cmCNT_DEV	Deviation Counter : Command 와 Feedback counter 의 편차 카운터
3 또는 cmCNT_GEN	General Counter : 사용자의 정의에 따라 여러가지 용도로 사용될 수 있는 카운터

▶ Count : 지정한 값으로 대상 카운터의 값을 설정합니다. 이 값은 이 값은 논리단위가 아닌 실제 펄스 카운트값입니다.

SEE ALSO

cmmStGetCount

RETURN VALUE

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

EXAMPLE

C/C++

```
#include "Cmmsdk.h"
#include "CmmsdkDef.h"
```

```
void OnSetCount ()
{
	long nCommCount;	// Command Count
	long nFeedCount;		// Feedback Count

	/* Command, Feedback 카운트 값을 확인한 후, '0' 으로 초기화 합니다. */

	if (cmmStGetCount (cmX1, cmCNT_COMM, &nCommCount ) == cmERR_NONE )
	{
		if ( nCommCount != 0 )
		{
			// Command 카운트를 0 으로 설정합니다.
			cmmStSetCount (cmX1, cmCNT_COMM, 0 );
		}
	}

	if (cmmStGetCount (cmX1, cmCNT_FEED, &nFeedCount ) == cmERR_NONE )
	{
		if ( nFeedCount != 0 )
		{
			// Feedback 카운트를 0 으로 설정합니다.
			cmmStSetCount (cmX1, cmCNT_FEED, 0 );
		}
	}
}
```

```
Visual Basic

Private Sub OnSetCount ()

	Dim nCommCount As Long	' Command Count
	Dim nFeedCount As Long		' Feedback Count

	' Command, Feedback 카운트 값을 확인한 후, '0' 으로 초기화 합니다.

	If cmmStGetCount (cmX1, cmCNT_COMM, nCommCount ) = cmERR_NONE Then
		If nCommCount <> 0 Then
			' Command 카운트를 0 으로 설정합니다.
			Call cmmStSetCount (cmX1, cmCNT_COMM, 0 )
		End If
	End If

	If cmmStGetCount (cmX1, cmCNT_FEED, nFeedCount ) = cmERR_NONE Then

		If nFeedCount <> 0 Then
			' Feedback 카운트를 0 으로 설정합니다.
			Call cmmStSetCount (cmX1, cmCNT_FEED, 0 )
		End If
	End If

End Sub
```

```
Delphi

procedure OnSetCount ();
var
	nCommCount : LongInt;		// Command Count
	nFeedCount : LongInt;		// Feedback Count

begin
	// Command, Feedback 카운트 값을 확인한 후, '0' 으로 초기화 합니다.

	if cmmStGetCount (cmX1, cmCNT_COMM, @nCommCount ) = cmERR_NONE then
	begin
		if nCommCount <> 0 then
		begin
			// Command 카운트를 0 으로 설정합니다.
```

```
                cmmStSetCount (cmX1, cmCNT_COMM, 0 );
            end;
        end;

        if cmmStGetCount (cmX1, cmCNT_FEED, @nFeedCount ) = cmERR_NONE then
        begin
            if nFeedCount <> 0 then
            begin
                // Feedback 카운트를 0 으로 설정합니다.
                cmmStSetCount (cmX1, cmCNT_FEED, 0 );
            end;
        end;

end;
```

NAME

cmmStGetCount
– 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 cmmStGetCount ([in] VT_I4 Axis, [in] VT_I4 Source, [out] VT_PI4 Count)

DESCRIPTION

지정한 채널의 지정한 카운터의 값을 읽어서 반환합니다. 단, 이때 반환되는 값의 단위는 펄스수입니다.

PARAMETER

▶ Axis: 축번호. 축번호는 상수값으로 [cmX1] 부터 0 번째 축을 기준 축으로 임의의 축을 설정할 수 있습니다.

▶ Source : 값을 읽을 카운터 번호. 이 값은 다음의 4 가지 값중의 하나이어야 합니다.

Value	Meaning
0 또는 cmCNT_COMM	Command Counter
1 또는 cmCNT_FEED	Feedback Counter
2 또는 cmCNT_DEV	Deviation Counter : Command 와 Feedback counter 의 편차 카운터
3 또는 cmCNT_GEN	General Counter : 사용자의 정의에 따라 여러가지 용도로 사용될 수 있는 카운터

▶ Count : 대상 카운터의 값을 반환합니다. 이 값은 이 값은 논리단위가 아닌 실제 펄스 카운트값입니다.

SEE ALSO

cmmStSetCount

RETURN VALUE

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

EXAMPLE

```
//* cmmStSetCount 예제를 참고하여 주시기 바랍니다.
```

NAME

cmmStSetPosition
– 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 cmmStSetPosition ([in] VT_I4 Axis, [in] VT_I4 Target, [in] VT_R8 Position)

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 cmmStSetCount() 함수와 동일합니다.

PARAMETER

▶ Axis: 축번호. 축번호는 상수값으로 [cmX1] 부터 0 번째 축을 기준 축으로 임의의 축을 설정할 수 있습니다.

▶ Target : 설정할 카운터 번호. 이 값은 다음의 4 가지 값중의 하나이어야 합니다.

Value	Meaning
0 또는 cmCNT_COMM	Command Counter
1 또는 cmCNT_FEED	Feedback Counter
2 또는 cmCNT_DEV	Deviation Counter : Command 와 Feedback counter 의 편차 카운터
3 또는 cmCNT_GEN	General Counter : 사용자의 정의에 따라 여러가지 용도로 사용될 수 있는 카운터

▶ Position : 대상 카운터에 설정될 값. 단, 이 값은 논리적 거리 단위로 설정하여야 합니다.

RETURN VALUE

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

SEE ALSO

cmmStGetPosition

REFERENCE

□ 논리적 거리 단위는 cmmCfgSetUnitDist() 함수에 의해 결정됩니다

EXAMPLE

```
//* cmmStSetCount 예제를 참고하여 주시기 바랍니다.
```

NAME

cmmStGetPosition
– 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 cmmStGetPosition ([in] VT_I4 Axis, [in] VT_I4 Source, [out] VT_PR8 Position)

DESCRIPTION

지정한 채널의 지정한 카운터의 값을 읽어서 반환합니다. 단, 이때 반환되는 값의 단위는 논리적 거리입니다.

PARAMETER

▶ Axis: 축번호. 축번호는 상수값으로 [cmX1] 부터 0 번째 축을 기준 축으로 임의의 축을 설정할 수 있습니다.

▶ Source : 대상 카운터 번호. 이 값은 다음의 4 가지 값중의 하나이어야 합니다.

Value	Meaning
0 또는 cmCNT_COMM	Command Counter
1 또는 cmCNT_FEED	Feedback Counter
2 또는 cmCNT_DEV	Deviation Counter : Command 와 Feedback counter 의 편차 카운터
3 또는 cmCNT_GEN	General Counter : 사용자의 정의에 따라 여러가지 용도로 사용될 수 있는 카운터

▶ Position : 전달된 변수를 통해 대상 카운터의 값을 읽어서 논리적 거리 단위로 반환합니다.

RETURN VALUE

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

SEE ALSO

cmmStSetPosition

REFERENCE

□ 논리적 거리 단위는 cmmCfgSetUnitDist() 함수에 의해 결정됩니다.

EXAMPLE

```
//* cmmStSetCount 예제를 참고하여 주시기 바랍니다.
```

NAME

cmmStGetSpeed
– 논리적(論理的) 속도 반환

INFORMATION

Motion Status
VC++/VB
BCB/Delphi/.NET
Level 7
위험 요소 없음

SYNOPSIS

❒ VT_I4 cmmStGetSpeed ([in] VT_I4 Axis, [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.

PARAMETER

▶ Axis: 축번호. 축번호는 상수값으로 [cmX1] 부터 0 번째 축을 기준 축으로 임의의 축을 설정할 수 있습니다.

▶ Source : 속도 반환대상이 되는 카운터 번호. 이 값은 다음의 2 가지 값중의 하나이어야 합니다.

Value	Meaning
0 또는 cmCNT_COMM	Command Counter
1 또는 cmCNT_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
cmERR_NONE	수행 성공

SEE ALSO

□ 논리적 거리 단위는 cmmCfgSetUnitDist() 함수에 의해 결정됩니다.

NAME

cmmStReadMotionState
– 모션 동작상태반환(動作狀態返還)

INFORMATION

- Motion Status
- VC++/VB
- BCB/Delphi/.NET
- Level 7
- 위험 요소 없음

SYNOPSIS

❒ VT_I4 cmmStReadMotionState ([in] VT_I4 Axis, [out] VT_PI4 MotStates)

DESCRIPTION

이 함수는 현재 모션의 동작 상태를 반환합니다.

PARAMETER

▶ Axis: 축번호. 축번호는 상수값으로 [cmX1] 부터 0 번째 축을 기준 축으로 임의의 축을 설정할 수 있습니다.

▶ MotStates : 모션 상태.

Value	Meaning
음수	모션작업 도중 에러 발생 => 에러코드 참조.
0 또는 cmMST_STOP	Stop
1 또는 cmMST_WAIT_DR	Waiting for DR input signal
2 또는 cmMST_WAIT_STA	Waiting for STA input signal
3 또는 cmMST_WAIT_INSYNC	Waiting for internal sync. signal
4 또는 cmMST_WAIT_OTHER	Waiting other axis
5 또는 cmMST_WAIT_ERC	Waiting for ERC output finished
6 또는 cmMST_WAIT_DIR	Waiting for DIR change (DIR 은 외부입력 신호가 아닌 내부적인 신호임)
7 또는 cmMST_RESERVED1	Reserved
8 또는 cmMST_WAIT_PLSR	Waiting for PA/PB input signal
9 또는 cmMST_IN_RVSSPD	In home special speed (reverse speed)
10 또는 cmMST_IN_INISPD	In start velocity motion
11 또는 cmMST_IN_ACC	In acceleration
12 또는 cmMST_IN_WORKSPD	In working velocity
13 또는 cmMST_IN_DEC	In deceleration
14 또는 cmMST_WAIT_INP	Waiting for INP input signal
15 또는 cmMST_SPARE0	Reserved

RETURN VALUE

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

SEE ALSO

cmmStGetMstString

EXAMPLE

C/C++

```
char szMstList[16][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",
        ""};

        long nMST;

        cmmStReadMotionState(cmX1, &nMST);

        if(nMST < 0){ // Status 가 0 보다 작으면 마지막 모션에서 에러 발생한 상태
                cmmErrShowLast(Handle);
        }

        ////////////////////////////////////////////////////////////
        // DisplayStatus() 함수는 MessageBox 와 같이 화면에 상태를 표시하는 가상의 함수입니다.
        CString sMsg = "Current Motion State : " + szMstList[nMST];
        DisplayStatus(sMsg)
}
```

Visual Basic

```
Private Sub Timer1_Timer()

Dim state As Long

   If (cmmStReadMotionState(cmX1, state) <> cmERR_NONE) Then
      Call cmmErrShowLast(Handle)
   End If

   Select Case state

      Case cmMST_STOP: lblState.Caption = "Stop"
      Case cmMST_WAIT_DR: lblState.Caption = "Waiting for DR input signal"
      Case cmMST_WAIT_STA: lblState.Caption = "Waiting for STA input signal"
      Case cmMST_WAIT_INSYNC: lblState.Caption = "Waiting for internal sync, signal"
      Case cmMST_WAIT_OTHER: lblState.Caption = "Waiting other axis"
      Case cmMST_WAIT_ERC: lblState.Caption = "Waiting for ERC output finished"
      Case cmMST_WAIT_DIR: lblState.Caption = "Waiting for DIR change"

      ' 아래 cmMST_RESERVED1 은 내부적으로 예약되었기 때문에 사용하지 않습니다.
      'Case cmMST_RESERVED1 : lblState.Caption = "RESERVED1"
      Case cmMST_WAIT_PLSR: lblState.Caption = "Waiting for PA/PB input signal"
      Case cmMST_IN_RVSSPD: lblState.Caption = "In home special speed (reverse speed)"
      Case cmMST_IN_INISPD: lblState.Caption = "In start velocity motion"
      Case cmMST_IN_ACC: lblState.Caption = "In Acceleration"
      Case cmMST_IN_WORKSPD: lblState.Caption = "In Working Velocity"
      Case cmMST_IN_DEC: lblState.Caption = "In Deceleration"
      Case cmMST_WAIT_INP: lblState.Caption = "Waiting for INP input signal"
```

```
    ' 아래 cmMST_SPARE0 는 내부적으로 예약되었습니다. 사용하지 않습니다.
    ' Case cmMST_SPARE0 : lblState.Caption = "RESERVED2"
  End Select

End Sub
```

Delphi

```
function GetMitonStateString( i : Integer ) : String;
begin
  Case i of

    cmMST_STOP : Result := 'Stop';
    cmMST_WAIT_DR : Result := 'Waiting for DR input signal';
    cmMST_WAIT_STA : Result := 'Waiting for STA input signal';
    cmMST_WAIT_INSYNC : Result := 'Waiting for internal sync, signal';
    cmMST_WAIT_OTHER : Result := 'Waiting other axis';
    cmMST_WAIT_ERC : Result := 'Waiting for ERC output finished';
    cmMST_WAIT_DIR : Result := 'Waiting for DIR change';

    // cmMST_RESERVED1 은 내부적으로 예약되었습니다. 사용하지 않습니다.
    //cmMST_RESERVED1 : Result := 'RESERVED1';
    cmMST_WAIT_PLSR : Result := 'Waiting for PA/PB input signal';
    cmMST_IN_RVSSPD : Result := 'In home special speed (reverse speed)';
    cmMST_IN_INISPD : Result := 'In start velocity motion';
    cmMST_IN_ACC : Result := 'In Acceleration';
    cmMST_IN_WORKSPD : Result := 'In Working Velocity';
    cmMST_IN_DEC : Result := 'In Deceleration';
    cmMST_WAIT_INP : Result := 'Waiting for INP input  signal';

    // cmMST_SPARE0 는 내부적으로 예약되었습니다. 사용하지 않습니다.
    //cmMST_SPARE0 : Result := 'RESERVED2';
  end;
end;

procedure OnEvent();
var
  stateus : LongInt;
  i : Integer;
begin

  // 에러 발생하면, 에러 메시지를 화면에 나타냅니다.
  if ( cmmStReadMotionState(cmX1, @stateus) <> cmERR_NONE ) then
  begin
    ShowMessage('cmmStReadMotionState has been failed');
    // 또는 cmmErrShowLast(self.Handle);
    exit;
  end;

  if ( stateus < 0 ) then
  begin

    ShowMessage('모션 동작 상태에 에러가 발생하였습니다.');
    // 또는 cmmErrShowLast(self.Handle);
    // OnEvent procedure 를 종료합니다.
    exit;
  end;
end;
```

NAME

cmmStReadMioStatuses
- 모션 관련 (MIO) 상태 반환(狀態返還)

INFORMATION

Motion Status

VC++/VB

BCB/Delphi/.NET

Level 7

위험 요소 없음

SYNOPSIS

❒ VT_I4 cmmStReadMioStatuses ([in] VT_I4 Axis, [out] VT_PI4 MioStates)

DESCRIPTION

이 함수는 현재 모션과 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.

PARAMETER

▶ Axis: 축번호. 축번호는 상수값으로 [cmX1] 부터 0 번째 축을 기준 축으로 임의의 축을 설정할 수 있습니다.

▶ MioStates : Machine I/O 상태

Bit No.	Name	Meaning
0 (cmIOST_RDY)	RDY	Servo ready signal input status(1=ON)
1 (cmIOST_ALM)	ALM	Alarm signal status(1=ON)
2 (cmIOST_ELP)	+EL	Positive limit switch status(1=ON)
3 (cmIOST_ELN)	-EL	Negative limit switch status(1=ON)
4 (cmIOST_ORG)	ORG	Origin switch status(1=ON)
5 (cmIOST_DIR)	DIR	Operating direction status(1=ON)
6 (cmIOST_RSV1)	Reserved	
7 (cmIOST_PCS)	PCS	PCS signal input status(1=ON)
8 (cmIOST_ERC)	ERC	ERC pin output status(1=ON)
9 (cmIOST_EZ)	EZ	Index signal status(1=ON)
10 (cmIOST_CLR)	CLR	Clear input status(1=ON)
11 (cmIOST_LTC)	Latch	Latch signal input status(1=ON)
12 (cmIOST_SD)	SD	Slow Down signal input status(1=ON)
13 (cmIOST_INP)	INP	In-Position signal input status(1=ON)
14 (cmIOST_DRP)	DRP	+DR input signal status(1=ON)
15 (cmIOST_DRN)	DRN	-DR input signal status(1=ON)
16 (cmIOST_STA)	STA	STA input signal status(1=ON)
17 (cmIOST_STP)	STP	STP input signal status(1=ON)
18 ~31	Reserved	

RETURN VALUE

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

EXAMPLE

C/C++

```
long dwMioState = 0;
cmmStReadMioStatuses(cmX1, &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;
……
……
```

Visual Basic

```
Dim dwMioState As Long
Dim RDY_State, ALM_State, ELP_State, ELN_State As Boolean

' dwMioState 의 값을 오른쪽으로 쉬프트 연산(Shift Operation) 하여, 해당 상태 값을 얻습니다. CMMSDK 는
' 비쥬얼 베이직 사용자를 위해서 편리한 비트 연산 함수를 제공합니다. 아래의 예제에서는 cmmGnBitShift
'함수 를 이용해 비트연산을 용이하게 수행하고 있습니다.

Call cmmStReadMioStatuses(cmX1, dwMioState)
Call cmmGnBitShift(dwMioState,-cmIOST_RDY, RDY_State) And 1
Call cmmGnBitShift(dwMioState,-cmIOST_ALM, ALM_State) And 1
Call cmmGnBitShift(dwMioState,-cmIOST_ELP, ELP_State) And 1
Call cmmGnBitShift(dwMioState,-cmIOST_ELN, ELN_State) And 1

RDY_State = ((RDY_State) And &H1)
ALM_State = ((ALM_State) And &H1)
ELP_State = ((ELP_State) And &H1)
ELN_State = ((ELN_State) And &H1)
……
……
```

Delphi

```
Var
    dwMioState : LongInt;
    RDY_State : Boolean;
    ALM_State : Boolean;
    ELP_State : Boolean;
    ELN_State : Boolean;

begin
    cmmStReadMioStatuses(cmX1,@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

cmmStGetMstString
– 모션 동작상태(動作狀態)와 관련된 문자열(文字列) 반환

INFORMATION

Motion Status

VC++/VB

BCB/Delphi/.NET

Level 7

위험 요소 없음

SYNOPSIS

❒ VT_I4 cmmStGetMstString ([in] VT_I4 MstCode, [out] VT_STR Buffer, [in] VT_I4 BufferLen)

DESCRIPTION

이 함수는 현재 모션 동작 상태(動作狀態)와 관련된 문자열을 반환합니다. CMMSDK 의 모션 동작 상태를 확인(確認)하는 cmmStReadMotionState() 함수를 통해 반환된 모션 동작 상태 값을 매개 변수로 전달하게 되면, 해당 상태에 대한 이미 내정된 문자열을 지정된 문자열 주소에 복사합니다. 이 문자열은 라이브러리 내부에서 이미 결정된 문자열 이므로, 사용자 정의 에러 메시지와는 무관합니다.

PARAMETER

▶ MstCode : 모션 동작 상태와 관련된 값입니다. cmmStReadMotionState() 함수를 통해 지정된 상태 값을 알아볼 수 있으며, 해당 상태 값을 이 함수의 매개변수로 활용할 수 있습니다.

▶ Buffer : 모션 동작 상태와 관련된 문자열을 복사해올 문자열 버퍼의 주소입니다.

▶ BufferLen : 모션 동작 상태를 확인(確認)하기 위해 전달된 문자열 버퍼의 길이입니다. 이 길이가 실제 복사될 모션 동작 상태보다 작으면, 실제 전달된 문자열 버퍼의 길이만큼 상태에 대한 문자열을 복사합니다.

RETURN VALUE

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

SEE ALSO

cmmStReadMioStatuses

EXAMPLE

C/C++

```
#include "Cmmsdk.h"
#include "CmmsdkDef.h"

void OnGetMstString ()
{
char szBuffer[32] = {0, };

// 현재 Motion State 를 Buffer 에 문자열로 저장합니다.
if ( cmmStGetMstString (cmX1, szBuffer, 32 ) == cmERR_NONE )
        {
                //DisplayMstString() 함수는 화면에 문자를 표시하는 가상의 함수입니다.
                DisplayMstString( szBuffer );
        }
}
```

11.2 인터럽트 이벤트

이 단원에서는 인터럽트에 관련된 함수들을 소개합니다. 인터럽트는 특정 상황이 발생되었을 때 사용자(또는 프로그래머)에게 이것을 알려주기 위한 것입니다. 인터럽트는 폴링(Polling) 방식과 달리 CPU에 부하를 주지 않는 것이 장점입니다. 윈도우 환경에서는 일반 Application 레벨에서 인터럽트를 처리할 수 없으므로 이벤트를 통하여 Application에게 인터럽트가 발생하였음을 알려줍니다.

㈜커미조아 모션컨트롤러는 InterruptEvent 라는 이벤트를 제공합니다. 사용자는 이 이벤트 핸들러를 등록하여 손쉽게 이벤트를 통지받을 수 있습니다. 이벤트 핸들러를 등록하는 방법은 일반적인 윈도우 리소스의 이벤트를 등록하는 것과 동일합니다.

사용자 프로그램에서 인터럽트이벤트 핸들러를 등록하였으면 인터럽트가 발생할 때마다 자동적으로 해당 이벤트 핸들러가 호출됩니다. 사용자는 이벤트 핸들러에서 아래표와 같이 인터럽트 처리 관련 함수들을 사용하여 인터럽트의 발생원인과 그에 대한 사후 처리 루틴을 작성하여야 합니다. ㈜커미조아 CMMSDK에서는 '윈도우 메시지' 방식, 'Call Back 함수 방식', '이벤트 객체' 방식의 3 가지의 이벤트 처리 방식을 사용할 수 있습니다.

인터럽트의 종류는 크게 "에러 인터럽트"와 "이벤트 인터럽트"로 나뉘어집니다. 이중에서 "이벤트 인터럽트"에 포함되는 인터럽트 상황들은 여러가지가 있으며, 각각의 상황들을 인터럽트로 처리할지를 마스크할 수 있습니다. "이벤트 인터럽트"에 속한 인터럽트들은 마스크되어야만 인터럽트로 처리됩니다.

"에러 인터럽트"는 마스크되지 않으며 이벤트 핸들러를 등록하면 "에러 인터럽트"에 해당하는 상황이 발생하면 무조건 통지됩니다.

11.2.1 함수 요약

인터럽트 처리에 관련된 이벤트 및 함수들은 다음과 같습니다.

Summary of Functions
❒ VT_I4 cmmIntSetMask ([in] VT_I4 Axis, [in] VT_I4 Mask) 지정한 모션 대상(對象) 채널의 이벤트 인터럽트(Event Interrupt) 조건을 설정합니다.
❒ VT_I4 cmmIntGetMask ([in] VT_I4 Axis, [out] VT_PI4 Mask) 지정한 모션 대상(對象) 채널의 이벤트 인터럽트(Event Interrupt) 조건을 반환(返還)합니다.
❒ VT_I4 cmmIntHandlerSetup ([in] VT_I4 HandlerType, [in] VT_HANDLE Handler, [in] VT_UI4 nMessage, [in] VT_HANDLE IParam) CMMSDK 가 지원하는 3 가지 유형의 인터럽트 처리 방식을 배경으로, 인터럽트 발생시 호출(呼出)될 핸들러를 비롯한 환경 설정을 구성합니다.
❒ VT_I4 cmmIntHandlerEnable ([in] VT_I4 IsEnable) 실제 인터럽트 동작에 있어, 인터럽트 핸들러를 활성화(活性化) 혹은 비활성화(非活性化) 합니다.
❒ VT_I4 cmmIntReadFlag ([out] VT_PI4 IntFlag1, [out] VT_PI4 IntFlag2) 각 축에 대한 인터럽트 발생여부를 알려주는 플래그를 반환합니다. 반환(返還)된 값의 각 비트값이 1 이면 해당 축에 인터럽트가 발생된 것임을 의미하게 됩니다. 32Bit 데이터형의 2 개의 전달 인자를 통해 총 64 채널의 인터럽트 플래그를 반환(返還)받을 수 있습니다.
❒ VT_I4 cmmIntReadErrorStatus ([in] VT_I4 Axis, [out] VT_PI4 ErrState) 지정한 모션 대상(對象) 채널의, 에러 인터럽트(Error Interrupt)에 대한 상태를 반환(返還)합니다.
❒ VT_I4 cmmIntReadEventStatus ([in] VT_I4 Axis, [out] VT_PI4 EventState) 지정한 모션 대상(對象) 채널의, 이벤트 인터럽트(Event Interrupt)에 대한 상태를 반환(返還)합니다.

11.2.2 함수 설명

NAME

cmmIntSetMask
– 이벤트 인터럽트 발생 조건(條件) 설정

INFORMATION

- Interrupt Event
- VC++/VB
- BCB/Delphi/.NET
- Level 7
- 위험 요소 없음

SYNOPSIS

❒ VT_I4 cmmIntSetMask ([in] VT_I4 Axis, [in] VT_I4 Mask)

DESCRIPTION

이 함수는 어떠한 조건의 인터럽트를 받아들일 것인지를 설정합니다. 인터럽트를 발생 시킬 조건을 Mask 매개 변수(媒介變數)를 통하여 설정하십시오. 단, 이 것은 "이벤트 인터럽트"에만 적용되는 것이며, "에러 인터럽트"는 마스크될 수 없습니다.

PARAMETER

▶ Axis: 축번호. 축번호는 상수값으로 [cmX1] 부터 0 번째 축을 기준 축으로 임의의 축을 설정할 수 있습니다.

▶ Mask : 인터럽트를 발생시킬 조건을 설정합니다. 이 값의 각 비트는 다음의 표와 같이 인터럽트 발생 조건을 설정합니다. 각 비트값이 1 이면 해당조건이 인터럽트를 발생시키며, 0 이면 발생시키지 않습니다.

BIT No.	Meaning
BIT0	Normal Stop
BIT1	Succesive start of the next operation
BIT2	Reserved
BIT3	Reserved
BIT4	Start of acceleration
BIT5	End of acceleration
BIT6	Start of deceleration
BIT7	End of deceleration
BIT8	Reserved
BIT9	Reserved
BIT10	Position error tolerance exceed (*cmmCmpErrSetConfig()* 함수 참조)
BIT11	General Comparator (*cmmCmpGenSetConfig()* 함수 참조)
BIT12	CMP output triggered (*cmmCmpTrgSetConfig()* 함수 참조)
BIT13	CLR signal input resetting counter value.
BIT14	LTC input making counter value latched
BIT15	ORG input signal ON
BIT16	SD input signal ON
BIT17	±DR input signal state changed
BIT18	STA input signal state turned ON

RETURN VALUE

Value	Meaning

음수	수행 실패. 자세한 내용은 '에러처리' 편을 참고합니다
cmERR_NONE	수행 성공

EXAMPLE

C/C++

```
//************************************************************
// OnbtnApplyMask(): "Apply Mask Change" 이벤트 핸들러.
// EventInterrupt 의 마스크를 설정한다. 마스크는 인터럽트를 받고자하는
// 종류를 선택하는 것입니다.
// 참고: ErrorInterrupt 는 마스크에 관계없이 항상 인터럽트를 받을
// 수 있도록 되어 있습니다.
//************************************************************

#define NUM_CHECKBOX 15
void CInterruptEventDlg::OnbtnApplyMask()
{
	int nAxis = cmX1;

	// IntMask 에서 몇 개의 비트가 Reserved 로 되어 있어서 화면상의
	// CheckBox 순서가
	// 비트 순서와 일치하지 않아서 아래와 같이 각 CheckBox 별로 비트 순서를
	// 알려주는 배열을 작성하였습니다.

	int nBitIdx[NUM_CHECKBOX] = {0, 1, 4, 5, 6, 7, 10, 11, 12, 13,  14, 15, 16, 17, 18};
	long dwIntMask = 0x0;
	for(int i=0; i<NUM_CHECKBOX; i++){
		if(IsDlgButtonChecked(IDC_chkMask0+i)){
			cmmSetBit(dwIntMask, nBitIdx[i], 1);
		} else{
			cmmSetBit(dwIntMask, nBitIdx[i], 0);
		}
	}
	cmmIntSetMask(nAxis, dwIntMask);
}
```

Visual Basic

```
'//************************************************************
'// OnbtnApplyMask(): "Apply Mask Change" 이벤트 핸들러.
'// EventInterrupt 의 마스크를 설정한다. 마스크는 인터럽트를 받고자하는
'// 종류를 선택하는 것입니다.
'// 참고: ErrorInterrupt 는 마스크에 관계없이 항상 인터럽트를 받을
'// 수 있도록 되어 있습니다.
'//************************************************************

Private Sub OnApplyMask()

	Dim nAxis As Long
	'// IntMask 에서 몇 개의 비트가 Reserved 로 되어 있어서 화면상의
	'// CheckBox 순서가
	'// 비트 순서와 일치하지 않아서 아래와 같이 각 CheckBox 별로 비트 순서를
	'// 알려주는 배열을 작성하였습니다.

	Dim nBitIdx(4) As Long
	nBitIdx(0) = 0
	nBitIdx(1) = 1
	nBitIdx(2) = 4
	nBitIdx(3) = 5


	Dim dwIntMask As Long

	dwIntMask = &H0

	If Check1.Value Then
```

```
    Call cmmSetBit(dwIntMask, nBitIdx(0), 1)
  Else
    Call cmmSetBit(dwIntMask, nBitIdx(0), 0)
  End If

  If Check2.Value Then
    Call cmmSetBit(dwIntMask, nBitIdx(1), 1)
  Else
    Call cmmSetBit(dwIntMask, nBitIdx(1), 0)
  End If

  If Check3.Value Then
    Call cmmSetBit(dwIntMask, nBitIdx(2), 1)
  Else
    Call cmmSetBit(dwIntMask, nBitIdx(2), 0)
  End If

  If Check4.Value Then
    Call cmmSetBit(dwIntMask, nBitIdx(3), 1)
  Else
    Call cmmSetBit(dwIntMask, nBitIdx(3), 0)
  End If

  Call cmmIntSetMask(nAxis, dwIntMask)

End Sub
```

Delphi

```
//  Description : 인터럽트 마스크를 설정합니다.
//                                  상수 배열 nBitIndex 의 요소가 0,1,2,3,4 순으로
//                                  1 씩 증가하지 않는 이유는 2 번이나 3 번등의 인터럽트 마스크
//                                  는 내부적으로 예약되어 있기 때문입니다. 따라서 해당
//                                  인터럽트 마스크 비트는 사용하지 말아야 합니다.

Procedure btnSetInterruptMaskClick();
const
  nBitIndex : Array[0..14] of LongInt  = (0, 1, 4, 5, 6, 7, 10, 11, 12, 13, 14, 15, 16, 17, 18);
var
  i : Integer;
  nIntMask : LongInt;
begin

  // CheckBox 컴포넌트를 검색하여, 어떤 CheckBox 가 Checked 되어 있는지
  // 확인(確認)합니다. 확인(確認)된 상태를 통해 Interrupt Mask 를 구성합니다.
  for i:=0 to 14 do begin

  if TCheckBox(FindComponent('CheckBox' + IntToStr(i))).Checked then
  begin
     nIntMask := (nIntMask) OR (1 shl nBitIndex[i]);
  end
  else
    nIntMask := nIntMask And Not(1 shl nBitIndex[i]);
  end;

  // 인터럽트 마스크 대상축은 cmX1 (0 번째 축) 을 기준으로 합니다.
  // 실제 응용프로그램에서는 인터럽트 이벤트 발생시에
  // 다축의 인터럽트 마스크일 경우 어떤 축의 이벤트가 발생했는지를
  // 확인(確認)해 볼 필요가 있습니다.
  cmmIntSetMask(cmX1,nIntMask);

  MessageBox('Event mask has been updated');

end;
```

NAME

cmmIntGetMask
– 이벤트 인터럽트 설정 조건(條件) 확인(確認)

INFORMATION

Interrupt Event
VC++/VB
BCB/Delphi/.NET
Level 7
위험 요소 없음

SYNOPSIS

❒ VT_I4 cmmIntGetMask ([in] VT_I4 Axis, [out] VT_PI4 Mask)

DESCRIPTION

설정된 인터럽트 마스크를 가져옵니다. 단, 이 것은 "이벤트 인터럽트" 조건에만 해당됩니다.

PARAMETER

▶ Axis: 축번호. 축번호는 상수값으로 [cmX1] 부터 0 번째 축을 기준 축으로 임의의 축을 설정할 수 있습니다.

▶ Mask : 설정된 인터럽트를 발생시킬 조건 값

RETURN VALUE

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

NAME

cmmIntHandlerSetup
– 인터럽트 이벤트 핸들러 등록(登錄)

INFORMATION

Interrupt Event
VC++/VB
BCB/Delphi/.NET
Level 7
위험 요소 없음

SYNOPSIS

❒ VT_I4 cmmIntHandlerSetup
([in] VT_I4 HandlerType, [in] VT_HANDLE Handler, [in] VT_UI4 nMessage, [in] VT_HANDLE IParam)

DESCRIPTION

인터럽트 이벤트 핸들러를 등록합니다. CMMSDK 에서는 윈도우 메시지 방식, 이벤트 객체 방식, 콜백 함수 방식등의 인터럽트 처리를 수행할 수 있으며, 해당 인터럽트가 발생되었을 때, 지정된 HandlerType 에 따른 처리가 이루어지게 됩니다.
이벤트 처리에 대한 보다 자세한 정보는 ㈜ 커미조아 CMMSDK 에서 제공하는 예제 프로그램의 실제 코드를 통해서 확인(確認)하시는 것도 좋은 방법입니다.

PARAMETER

▶ HandlerType : 인터럽트 핸들러의 종류를 지정합니다. 이 값의 종류는 아래와 같이 3 가지로 지정할 수 있습니다.

Value	Meaning
0 (cmIHT_MESSAGE)	윈도우 메시지 전달방식을 통하여 인터럽트를 통지합니다. • 장점: GUI 관련 작업을 이벤트핸들러에서 직접 수행할 수 있다. • 단점: 윈도우가 메시지루프 상황에 따라서 상당한 지연시간을 가질 수 있다.
1 (cmIHT_EVENT)	이벤트 객체를 통하여 인터럽트를 통지합니다. • 장점: 메시지 전달 방식보다 비교적 적은 지연시간을 가질 수 있다. • 단점: 사용하기가 복잡하다.
2 (cmIHT_CALLBACK)	콜백 함수를 통하여 인터럽트를 통지합니다. 이 방식이 가장 권장되는 방식입니다. • 장점: 메시지 전달의 지연시간이 3 가지 방식 중에서 가장 적다. • 단점: GUI 관련 작업을 수행할 수 없다.

▶ Handler : 이 매개변수의 의미는 HandlerType 의 설정에 따라서 다음과 같이 달라집니다.

HandlerType 설정값	Handler 매개변수의 의미
0 (메시지 방식) 일때	Handler 매개변수는 윈도우 핸들을 의미합니다.
1 (이벤트 방식) 일때	Handler 매개변수는 이벤트 객체를 의미합니다.
2 (콜백 방식) 일때	Handler 매개변수는 콜백 함수를 의미합니다.

▶ nMessage : 이 매개변수는 HandlerType 이 cmIHT_MESSAGE 로 설정되었을 때만 유효한 것으로서 윈도우 메시지 번호를 설정합니다.

▶ IParam : 인터럽트가 처리되는 함수에 전달될 핸들값을 설정합니다.

RETURN VALUE

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

NAME

cmmIntHandlerEnable
- 인터럽트 이벤트 활성화(活性化)

INFORMATION

Interrupt Event
VC++/VB
BCB/Delphi/.NET
Level 7
위험 요소 없음

SYNOPSIS

❒ VT_I4 cmmIntHandlerEnable ([in] VT_I4 Enable)

DESCRIPTION

인터럽트 핸들러를 사용할 수 있도록 설정합니다.

PARAMETER

▶ Enable : 인터럽트 이벤트 핸들러 사용 가능 여부.

RETURN VALUE

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

NAME

cmmIntReadFlag
– 각 축의 인터럽트 발생(發生) 여부 확인(確認)

INFORMATION

- Interrupt Event
- VC++/VB
- BCB/Delphi/.NET
- Level 7
- 위험 요소 없음

SYNOPSIS

❒ VT_I4 cmmIntReadFlag

([out] VT_PI4 IntFlag1, [out] VT_PI4 IntFlag2)

DESCRIPTION

각 축에 대한 인터럽트 발생여부를 알려주는 플래그를 반환합니다. 반환된 값의 각 비트값이 1 이면 해당 축에 인터럽트가 발생된 것임을 의미합니다.
실제 구현 코드에서는 개별적으로 cmmIntReadErrorStatus() 와 cmmIntReadEventStatus() 를 사용해서도 인터럽트가 발생되었는지를 확인(確認) 할 수 있습니다. 다만 여러축의 인터럽트 상태를 확인(確認)할 때에는 cmmIntReadFlag() 함수를 사용하는 것이 처리속도 측면에서 상당히 유리합니다.

PARAMETER

▶ IntFlag1 : 하위 32 축에 대한 비트 값 논리합

▶ IntFlag2 : 상위 32 축에 대한 비트 값 논리합.

BIT No.	Meaning
BIT0	0 번 축의 인터럽트 발생 여부. 0=>발생안함, 1=>인터럽트 발생
BIT1	1 번 축의 인터럽트 발생 여부. 0=>발생안함, 1=>인터럽트 발생
BIT2	2 번 축의 인터럽트 발생 여부. 0=>발생안함, 1=>인터럽트 발생
BIT3	3 번 축의 인터럽트 발생 여부. 0=>발생안함, 1=>인터럽트 발생
BIT4	4 번 축의 인터럽트 발생 여부. 0=>발생안함, 1=>인터럽트 발생
BIT5	5 번 축의 인터럽트 발생 여부. 0=>발생안함, 1=>인터럽트 발생
BIT6	6 번 축의 인터럽트 발생 여부. 0=>발생안함, 1=>인터럽트 발생
BIT7	7 번 축의 인터럽트 발생 여부. 0=>발생안함, 1=>인터럽트 발생

RETURN VALUE

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

REFERENCE

□ Visual Basic 에서는 비트마스킹이 용이하지 않습니다. CMMSDK 에서 제공하는 cmmGnBitShift() 함수를 사용하면 Visual Basic 에서도 비트마스킹을 수행할 수 있습니다.

NAME

cmmIntReadErrorStatus
– 해당 축의 에러 인터럽트 상태 확인(確認)

INFORMATION

Interrupt Event
VC++/VB
BCB/Delphi/.NET
Level 7
위험 요소 없음

SYNOPSIS

❒ VT_I4 cmmIntReadErrorStatus
([in] VT_I4 Axis, [out] VT_PI4 ErrState)

DESCRIPTION

지정한 축의 "에러 인터럽트"에 대한 상태를 반환합니다.

PARAMETER

▶ Axis: 축번호. 축번호는 상수값으로 [cmX1] 부터 0 번째 축을 기준 축으로 임의의 축을 설정할 수 있습니다.

▶ ErrState : 발생된 인터럽트를 통해 확인(確認)되는 에러 상태.

지정한 축의 "에러 인터럽트"에 대한 상태를 반환합니다. 에러가 없으면 이값은 0 이며, 에러가 발생한 경우에는 아래와 같이 에러코드값을 반환합니다.

Value	Meaning
0 (cmERR_NONE)	에러 없음
-5002 (cmERR_STOP_BY_SLP)	Abnormally stopped by positive soft limit
-5003 (cmERR_STOP_BY_SLN)	Abnormally stopped by negative soft limit
-5004 (cmERR_STOP_BY_CMP3)	Abnormally stopped by comparator3
-5005 (cmERR_STOP_BY_CMP4)	Abnormally stopped by comparator4
-5006 (cmERR_STOP_BY_CMP5)	Abnormally stopped by comparator5
-5007 (cmERR_STOP_BY_ELP)	Abnormally stopped by (+) external limit
-5008 (cmERR_STOP_BY_ELN)	Abnormally stopped by (-) external limit
-5009 (cmERR_STOP_BY_ALM)	Abnormally stopped by alarm input signal
-5010 (cmERR_STOP_BY_CSTP)	Abnormally stopped by CSTP input signal
-5011 (cmERR_STOP_BY_CEMG)	Abnormally stopped by CEMG input signal
-5012 (cmERR_STOP_BY_SD)	Abnormally stopped by SD input signal
-5013 (cmERR_STOP_BY_DERROR)	Abnormally stopped by operation data error
-5014 (cmERR_STOP_BY_IP)	Abnormally stopped by other axis error during interpolation
-5015 (cmERR_STOP_BY_PO)	An overflow occurred in the PA/PB input buffer
-5016 (cmERR_STOP_BY_AO)	Out of range position counter during interpolation
-5017 (cmERR_STOP_BY_EE)	An EA/EB input error occurred (does not stop)
-5018 (cmERR_STOP_BY_PE)	An PA/PB input error occurred (does not stop)
-5019 (cmERR_STOP_BY_SLVERR)	Abnormally stopped because slave axis has been stopped (Only in Master/Slave mode)

다양한 에러 상태는 CMMSDK 의 버전에 따라서 수시로 업데이트 되므로, 최신 에러 코드는 각 개발 환경 별 인터페이스 파일(Header File) 의 에러코드 정의 부를 참조해주시기 바랍니다.

RETURN VALUE

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

NAME

cmmIntReadEventStatus
– 해당 축의 이벤트 인터럽트 확인(確認)

INFORMATION

Interrupt Event
VC++/VB
BCB/Delphi/.NET
Level 7
위험 요소 없음

SYNOPSIS

❒ VT_I4 cmmIntReadEventStatus
([in] VT_I4 Axis, [out] VT_PI4 EventState)

DESCRIPTION

지정한 축의 "이벤트 인터럽트"에 대한 상태를 반환합니다.

PARAMETER

▶ Axis: 축번호. 축번호는 상수값으로 [cmX1] 부터 0 번째 축을 기준 축으로 임의의 축을 설정할 수 있습니다.

▶ EventState : 지정한 축의 "이벤트 인터럽트"에 대한 상태를 반환합니다. 이 값은 비트별로 각 이벤트에 대한 상태를 나타내므로 사용자는 비트마스크를 통하여 각 이벤트의 상태를 확인(確認)하여야 합니다.

BIT No.	Meaning
BIT0	Normal Stop
BIT1	Succesive start of the next operation
BIT2	Reserved
BIT3	Reserved
BIT4	Start of acceleration
BIT5	End of acceleration
BIT6	Start of deceleration
BIT7	End of deceleration
BIT8	Reserved
BIT9	Reserved
BIT10	Position error tolerance exceed (*CmpErrSetConfig()* 함수 참조)
BIT11	General Comparator (*CmpGenSetConfig()* 함수 참조)
BIT12	CMP output triggered (*CmpTrgSetConfig()* 함수 참조)
BIT13	CLR signal input resetting counter value.
BIT14	LTC input making counter value latched
BIT15	ORG input signal ON
BIT16	SD input signal ON
BIT17	+DR input signal state changed
BIT18	-DR input signal state changed
BIT19	STA input signal state turned ON

RETURN VALUE

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

REFERENCE

☐ Visual Basic 에서는 비트마스킹이 용이하지 않습니다. CMMSDK 에서 제공하는 cmmGnBitShift() 함수를 사용하면 Visual Basic 에서도 비트마스킹을 수행할 수 있습니다.

11.3 위치값 래치(Position Latch)

Position Latch는 특정 순간에 Motion의 위치 관련 카운터값을 래치(Latch)하여 읽을 수 있도록하는 기능입니다. Position Latch는 LTC입력핀에 LATCH신호가 입력되면 그 순간의 각 카운트값을 래치(Latch)합니다. 이 기능은 LTC 핀에 입력되는 하드웨어 신호에 동기되어서 각 위치값이 래치되므로 시간 지연이 거의 발생하지 않습니다.

사용자는 cmmIntReadEventStatus() 함수를 이용하여 래치 상태(래치가 발생했는지)를 확인(確認)할 수 있으며, 래치가 되었으면 cmmLtcReadLatch() 함수를 이용하여 래치된 위치 카운트값을 읽을 수 있습니다.

11.3.1 함수 요약

Position Latch에 관련된 함수들은 다음과 같습니다.

Summary of Functions
❒ VT_I4 cmmIntReadEventStatus ([in] VT_I4 Channel, [out] VT_PI4 EventState) 지정한 모션 대상(對象) 축의 이벤트 인터럽트에 대한 상태를 반환합니다. 이 반환값의 BIT14 값을 참조하면 래치 상태(래치가 발생했는지)를 알수 있습니다.
❒ VT_I4 cmmLtcIsLatched ([in] VT_I4 Axis, [out]VT_PI4 IsLatched) 지정한 모션 대상(對象) 축의 래치 카운터(Latch Counter) 가 활성화 되었음을 확인하고, 결과를 전달인자를 통해 반환(返還)합니다.
❒ VT_I4 cmmLtcReadLatch ([in] VT_I4 Axis, [in] VT_I4 Counter, [out] VT_PR8 LatchedPos) 지정한 모션 대상(對象) 축의 래치 카운터(Latch Counter) 에 저장된 카운터(Counter) 값을 전달 인자를 통해 반환(返還)합니다.
❒ VT_I4 cmmLtcQue_SetCfg ([in] VT_I4 Axis, [in] VT_I4 QueSize, [in] VT_I4 LtcTargCntr)
❒ VT_I4 cmmLtcQue_GetCfg ([in] VT_I4 Axis, [out] VT_PI4 pQueSize, [out] VT_PI4 pLtcTargCntr)
❒ VT_I4 cmmLtcQue_SetEnable ([in] VT_I4 Axis, [in] VT_I4 IsEnabled)
❒ VT_I4 cmmLtcQue_GetEnable ([in] VT_I4 Axis, [out] VT_PI4 pIsEnabled)
❒ VT_I4 cmmLtcQue_GetItemCount([in] VT_I4 Axis, [out] VT_PI4 pLtcItemCount)
❒ VT_I4 cmmLtcQue_ResetItemCount([in] VT_I4 Axis)
❒ VT_I4 cmmLtcQue_Deque ([in] VT_I4 Axis, [out] VT_PR8 pLtcData)
❒ VT_I4 cmmLtcQue_PeekAt ([in] VT_I4 Aixs,[in] VT_I4 Index, [out] VT_PR8 pLtcData)

□ LATCH(LTC) 입력 신호의 로직 설정

래치 기능을 사용하기 위해서는 가장 먼저 LATCH 신호(LTC 입력핀에 전달되는 신호)의 입력 로직이 Normal Open(A접점) 방식인지 Normal Close(B접점) 방식인지를 설정해주어야 합니다. 본 설정에 대한 부분은 CME Builder 를 통해, 보다 쉽고 빠르게 GUI 환경 하에서 설정가능하며, 생성되는 CMMSDK 환경 파일인, CME2 파일을 통해서 지정 하실 수 있습니다. CME2 파일은 COMIZOA Motion Environment 2 파일로서, 커미조아 모션 시스템의 환경설정을 편리하게 할 수 있는 도구입니다. 해당 응용프로그램의 사용방법을 자세히 알기 원하실 경우에는 부록 CME Builder 를 이용한 환경설정 편을 읽어주시기 바랍니다. 개별적인 함수를 사용할 경우에는 cmmCfgSetMioProperty 함수를 통해 입력 로직을 설정할 수 있습니다.

□ LATCH(LTC) 인터럽트 이벤트 마스크

래치 상태(래치가 발생했는지)를 확인(確認)하기 위해서는 cmmIntReadEventStatus() 함수를 사용합니다. 따라서 LATCH(LTC) 인터럽트 이벤트를 마스크해주어야 합니다. LATCH(LTC) 인터럽트 이벤트를 마스크하는 방법은 CME Builder 를 사용하실 수 있으며, 개별적인 함수를 사용할 경우에는 cmmIntSetMask 함수를 사용해서 인터럽트 마스크(Interrupt Mask) 를 설정할 수 있습니다. 자세한 내용은 인터럽트 이벤트 단원을 참조해주시기 바랍니다.

11.3.2 함수 설명

NAME

cmmLtcIsLatched
– 해당축의 래치(Latch)카운터 활성화(活性化) 여부 확인(確認)

INFORMATION

Latch
VC++/VB
BCB/Delphi/.NET
Level 7
위험 요소 없음

SYNOPSIS

❒ VT_I4 cmmLtcIsLatched
([in] VT_I4 Axis, [out]VT_PI4 IsLatched)

DESCRIPTION

해당축의 래치 카운터가 활성화 되었음을 확인(確認)하고, 결과를 반환합니다.

PARAMETER

▶ Axis: 축번호. 축번호는 상수값으로 [cmX1] 부터 0 번째 축을 기준 축으로 임의의 축을 설정할 수 있습니다.

▶ IsLatched : 해당축의 래치 카운터의 활성화 여부를 반환합니다.

RETURN VALUE

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

EXAMPLE

C/C++

```
#include "Cmmsdk.h"
#include "CmmsdkDef.h"

// 0 번 축의 래치 카운터 활성화 여부를 반환합니다.
long nIsLatched = cmFALSE;
cmmLtcIsLatched (cmX1, &nIsLatched );
```

Visual Basic

```
' 0 번 축의 래치 카운터 활성화 여부를 반환합니다.
Dim nIsLatched As Long
nIsLatched = cmFALSE

Call cmmLtcIsLatched (cmX1, nIsLatched)
```

Delphi

```
// 0 번 축의 래치 카운터 활성화 여부를 반환합니다.
var
        nIsLatched : LongInt;

begin
        nIsLatched := cmFALSE;

        cmmLtcIsLatched (cmX1, @nIsLatched );

end;
```

NAME

cmmLtcReadLatch
– 해당축의 래치(Latch)카운터값 확인(確認)

INFORMATION

Latch
VC++/VB
BCB/Delphi/.NET
Level 7
위험 요소 없음

SYNOPSIS

❒ VT_I4 cmmLtcReadLatch
([in] VT_I4 Axis, [in] VT_I4 Counter, [out] VT_PR8 LatchedPos)

DESCRIPTION

지정한 축의 현재 래치된 카운트값을 반환합니다. 이때 반환되는 위치 값의 단위는 논리적 거리 단위가 적용됩니다.

PARAMETER

▶ Axis: 축번호. 축번호는 상수값으로 [cmX1] 부터 0 번째 축을 기준 축으로 임의의 축을 설정할 수 있습니다.

▶ Counter : 읽을 래치 카운터를 지정합니다. 이 값은 다음과 같습니다.

Value	Meaning
0	명령위치 카운터(Command position counter)
1	실제위치 카운터(Feedback position counter)
2	Deviation 또는 펄스 출력 속도.
3	General Counter

▶ LatchedPos : 지정한 축의 래치된 카운트값을 반환합니다. 이때 반환되는 위치 값의 단위는 논리적 거리 단위가 적용됩니다

RETURN VALUE

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

EXAMPLE

C/C++

```
#include "Cmmsdk.h"
#include "CmmsdkDef.h"

// 0 번 축의 래치 카운트를 Feedback position counter 로 설정하고 래치된 카운트 값을 반환합니다.
long nLtcCounter = cmCNT_FEED;
double fLatchedPos = 0.0f;
cmmLtcReadLatched(cmX1, nLtcCounter, &nLatchedPos );
```

Visual Basic

```
'0 번 축의 래치 카운트를 Feedback position counter 로 설정하고 래치된 카운트 값을 반환합니다.
Dim nLtcCounter As Long
```

```
Dim fLatchedPos As Double

nLtcCounter = cmCNT_FEED
fLatchedPos = 0#

Call cmmLtcReadLatched(cmX1, nLtcCounter, nLatchedPos )
```

```
Delphi

// 0 번 축의 래치 카운트를 Feedback position counter 로 설정하고 래치된 카운트 값을 반환합니다.
var
        nLtcCounter : LongInt;
        fLatchedPos : Double;

begin

        nLtcCounter := cmCNT_FEED;
  fLatchedPos := 0;
        cmmLtcReadLatch(cmX1, nLtcCounter, @fLatchedPos );

end;
```

NAME

cmmLtcQue_SetCfg
– 래치 카운터 소스 및 래치큐(Latch Queue) 크기 설정

INFORMATION

- Latch
- VC++/VB
- BCB/Delphi/.NET
- Level 7
- ☺ 위험 요소 없음

SYNOPSIS

❒VT_I4 cmmLtcQue_SetCfg

([in] VT_I4 Axis, [in] VT_I4 QueSize, [in] VT_I4 LtcTargCntr)

DESCRIPTION

cmmLtcQue_SetCfg() 함수는 래치큐(Latch Queue) 의 크기 및 래치(Latch) 대상 카운터(Counter) 를 선택 합니다.

PARAMETER

▶ Axis: 축번호. 축번호는 상수값으로 [cmX1] 부터 0 번째 축을 기준 축으로 임의의 축을 설정할 수 있습니다.

▶ QueSize : 래치큐(Latch Queue) 의 크기

▶ LtcTargCntr : 대상 래치 카운터

Value	Meaning
cmCNT_COMM (또는 0)	Command Pulse Counter
cmCNT_FEED (또는 1)	Feedback Pulse Counter

RETURN VALUE

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

EXAMPLE

C/C++

```
#include "Cmmsdk.h"
#include "CmmsdkDef.h"

// 0 번 축의 래치 카운트를 Feedback position counter 로 설정하고 래치된 카운트 값을 반환합니다.
long nLtcCounter = cmCNT_FEED;
double fLatchedPos = 0.0f;
cmmLtcReadLatched(cmX1, nLtcCounter, &nLatchedPos );
```

NAME

cmmLtcQue_GetCfg
- 설정된 래치 카운터 소스 및 래치큐(Latch Queue) 크기 반환

INFORMATION

- Latch
- VC++/VB
- BCB/Delphi/.NET
- Level 7
- ☺ 위험 요소 없음

SYNOPSIS

❒VT_I4 cmmLtcQue_GetCfg

([in] VT_I4 Axis, [out] VT_PI4 pQueSize, [out] VT_PI4 pLtcTargCntr)

DESCRIPTION

cmmLtcQue_GetCfg() 함수는 설정된 래치큐(Latch Queue) 의 크기 및 래치(Latch) 대상 카운터(Counter) 종류를 반환합니다.

PARAMETER

▶ Axis: 축번호. 축번호는 상수값으로 [cmX1] 부터 0 번째 축을 기준 축으로 임의의 축을 설정할 수 있습니다.

▶ QueSize : 래치큐(Latch Queue) 의 크기

▶ pLtcTargCntr : 대상 래치 카운터

Value	Meaning
cmCNT_COMM (또는 0)	Command Pulse Counter
cmCNT_FEED (또는 1)	Feedback Pulse Counter

RETURN VALUE

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

NAME

cmmLtcQue_SetEnable
- 래치큐(Latch Queue) 기능 사용 활성화

INFORMATION

- Latch
- VC++/VB
- BCB/Delphi/.NET
- Level 7
- ☺ 위험 요소 없음

SYNOPSIS

❒VT_I4 cmmLtcQue_SetEnable

([in] VT_I4 Axis, [in] VT_I4 IsEnabled)

DESCRIPTION

해당 축의 래치 카운터를 큐(Queue) 형태로 사용할 수 있도록 활성화 또는 비활성화 시킵니다.

PARAMETER

▶ Axis: 축번호. 축번호는 상수값으로 [cmX1] 부터 0 번째 축을 기준 축으로 임의의 축을 설정할 수 있습니다.

▶ IsEnabled : 해당축의 래치큐 활성화 여부를 설정합니다.

SEE ALSO

cmmLtcQue_GetEnable

RETURN VALUE

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

NAME

cmmLtcQue_GetEnable
- 래치큐(Latch Queue) 기능 활성여부 반환

INFORMATION

Latch

VC++/VB

BCB/Delphi/.NET

Level 7

위험 요소 없음

SYNOPSIS

❒VT_I4 cmmLtcQue_GetEnable

([in] VT_I4 Axis, [out] VT_PI4 pIsEnabled)

DESCRIPTION

해당 축에 대한 래치 카운터 큐(Queue) 설정 활성화 여부 반환합니다.

PARAMETER

▶ Axis: 축번호. 축번호는 상수값으로 [cmX1] 부터 0 번째 축을 기준 축으로 임의의 축을 설정할 수 있습니다.

▶ pIsEnabled : 해당축의 래치큐 활성화 여부를 반환합니다.

SEE ALSO

cmmLtcQue_SetEnable

RETURN VALUE

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

NAME

cmmLtcQue_GetItemCount
- 래치큐(Latch Queue) 상의 래치(Latch) 된 데이터 개수 반환

INFORMATION

- Latch
- VC++/VB
- BCB/Delphi/.NET
- Level 7
- ☺ 위험 요소 없음

SYNOPSIS

❒ VT_I4 cmmLtcQue_GetItemCount([in] VT_I4 Axis, [out] VT_PI4 pLtcItemCount)

DESCRIPTION

해당 축에 대한 래치(Latch) 카운터 큐(Queue) 에 들어있는 래치(Latch) 된 Data 의 개수를 반환 합니다.

PARAMETER

▶ Axis: 축번호. 축번호는 상수값으로 [cmX1] 부터 0 번째 축을 기준 축으로 임의의 축을 설정할 수 있습니다.

▶ pLtcItemCount : 해당축의 래치 카운터 큐상의 Latch 된 데이터 개수

RETURN VALUE

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

NAME

cmmLtcQue_ResetItemCount
– 래치큐(Latch Queue) 상의 큐 인덱스(Queue Index) 초기화

INFORMATION

- Latch
- VC++/VB
- BCB/Delphi/.NET
- Level 7
- 위험 요소 없음

SYNOPSIS

❒ VT_I4 cmmLtcQue_ResetItemCount([in] VT_I4 Axis)

DESCRIPTION

cmmLtcQue_ResetItemCount() 함수는 래치큐(Latch Queue) 상의 큐 인덱스(Queue Index) 를 초기화 합니다. 결과적으로 래치큐(Latch Queue) 상에 들어 있는 래치(Latch) 된 데이터는 모두 초기화 됩니다.

PARAMETER

▶ Axis: 축번호. 축번호는 상수값으로 [cmX1] 부터 0 번째 축을 기준 축으로 임의의 축을 설정할 수 있습니다.

▶ Index : 래치큐(Latch Queue) 에 들어 있는 데이터 순번. 먼저 래치(Latch) 된 순서로 저장되어 있습니다.

▶ pLtcData : 확인된 Data 를 저장할 변수

RETURN VALUE

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

NAME

cmmLtcQue_Deque
– 래치큐(Latch Queue) 상의 가장 먼저 래치(Latch) 된 데이터를 큐에서 꺼냄

INFORMATION

- Latch
- VC++/VB
- BCB/Delphi/.NET
- Level 7
- 위험 요소 없음

SYNOPSIS

❒ VT_I4 cmmLtcQue_Deque([in] VT_I4 Axis, [out] VT_PR8 pLtcData)

DESCRIPTION

cmmLtcQue_Deque() 함수는 FIFO(First In First Out) 형태인 Queue 구조 상 먼저 래치(Latch) 된 데이터 순으로 큐(Queue) 로부터 꺼낼 때 사용 됩니다.

PARAMETER

▶ Axis: 축번호. 축번호는 상수값으로 [cmX1] 부터 0 번째 축을 기준 축으로 임의의 축을 설정할 수 있습니다.

▶ pLtcItemCount : 해당축의 래치 카운터 큐상의 Latch 된 데이터 개수

RETURN VALUE

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

NAME

cmmLtcQue_PeekAt
– 래치큐(Latch Queue) 상에서 임의의 데이터 확인

INFORMATION

- Latch
- VC++/VB
- BCB/Delphi/.NET
- Level 7
- 위험 요소 없음

SYNOPSIS

❒ VT_I4 cmmLtcQue_PeekAt([in] VT_I4 Aixs,[in] VT_I4 Index, [out] VT_PR8 pLtcData)

DESCRIPTION

cmmLtcQue_PeekAt() 함수는 래치큐(Latch Queue) 상에서 Index 로 지정된 임의의 데이터를 확인할 때 사용 됩니다. 단, 데이터 확인만 할 뿐 실제 데이터를 큐(Queue) 에서 제거하지 않습니다.

PARAMETER

▶ Axis: 축번호. 축번호는 상수값으로 [cmX1] 부터 0 번째 축을 기준 축으로 임의의 축을 설정할 수 있습니다.

▶ Index : 래치큐(Latch Queue) 에 들어 있는 데이터 순번. 먼저 래치(Latch) 된 순서로 저장되어 있습니다.

▶ pLtcData : 확인된 Data 를 저장할 변수

RETURN VALUE

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

Chapter

12

Compare Method

㈜커미조아의 강력한 모션 기능은 비교기기능과 연계되어 머신 비젼의 모션 응용 도구로서 활용됩니다. 고성능 모션 시스템에 계획된 머신 비젼을 위한 준비로서 커미조아의 모션 제품은 탁월한 선택이 아닐 수 없습니다. 국내 최고 사양의 머신 비젼 트리거 출력기능의 제품을 보유하고 있는 커미조아의 기술력(技術力)을 이제 마음껏 활용하십시요.

범용 비교기 기능과 위치 비교 출력 기능은 커미조아 모션 기능 (機能)의 꽃이라고 할 수 있습니다. 특히 있습니다. 특히 LX504a 제품에서 지원되는 최대 40Khz 사양의 머신 비젼 시스템(Machine Vision System) 을 Vision System) 을 위한 비젼 트리거 출력(Vision Trigger Output) 기능은 고속의 라인 CCD 촬영이 요구되는 촬영이 요구되는 현장의 다양한 시스템에서 그 기능 (機能)을 활용(活用)할 수 있습니다.

12 비교기 편

12.1 범용비교기

㈜커미조아 모션컨트롤러는 각 축마다 위치비교출력 기능을 위한 비교기 외에 각 두개의 비교기를 추가적으로 제공합니다.

하나는 에러비교기(Error comparator)로서, Command 카운터와 Feedback 카운터의 편차를 비교하는 비교기입니다. 사용자는 편차의 한계를 지정한 후에 지정한 값보다 큰 편차가 발생하면 인터럽트를 발생하도록 할 수 있습니다.

다른 하나는 범용비교기(General comparator)입니다. 이 비교기는 비교대상 카운터와 비교조건을 만족시켰을 때의 동작모드를 사용자가 임의로 설정할 수 있습니다.

12.1.1 함수 요약

범용비교기 제어와 관련된 함수들은 다음과 같습니다.

Summary of Functions
❒ VT_I4 cmmCmpErrSetConfig ([in] VT_I4 Axis, [in] VT_R8 Tolerance, [in] VT_I4 IsEnable) 대상(對象) 모션 채널에 대해서, 에러비교기(Error Comparator)의 환경을 설정(設定)합니다.
❒ VT_I4 cmmCmpErrGetConfig ([in] VT_I4 Axis, [out] VT_PR8 Tolerance, [out] VT_PI4 IsEnabled) 대상(對象) 모션 채널에 대해서, 에러 비교기(Error Comparator)의 설정된 환경을 반환(返還)합니다.
❒ VT_I4 cmmCmpGenSetConfig ([in] VT_I4 Axis, [in] VT_I4 CmpSrc, [in] VT_I4 CmpMethod, [in] VT_I4 CmpAction, [in] VT_R8 CmpData) 대상(對象) 모션 채널에 대해서, 범용 비교기(General Comparator)의 환경을 설정(設定)합니다.
❒ VT_I4 cmmCmpGenGetConfig ([in] VT_I4 Axis, [out] VT_PI4 CmpSrc, [out] VT_PI4 CmpMethod, [out] VT_PI4 CmpAction, [out] VT_PR8CmpData) 대상(對象) 모션 채널에 대해서, 범용 비교기(General Comparator)의 설정된 환경(環境)을 읽어들입니다.

12.1.2 함수 설명

NAME

cmmCmpErrSetConfig
cmmCmpErrGetConfig
– 편차(偏差) 카운터(에러비교기) 환경설정(環境設定) 및 확인(確認)

INFORMATION

- Compare Method
- VC++/VB
- BCB/Delphi/.NET
- Level 8
- ☺ 위험 요소 없음

SYNOPSIS

❒ VT_I4 cmmCmpErrSetConfig
([in] VT_I4 Axis, [in] VT_R8 Tolerance, [in] VT_I4 IsEnable)
❒ VT_I4 cmmCmpErrGetConfig
([in] VT_I4 Axis, [out] VT_PR8 Tolerance, [out] VT_PI4 IsEnabled)

DESCRIPTION

cmmCmpErrSetConfig() 함수는 에러비교기(Error comparator)의 환경을 설정합니다. 에러비교기는 Command 카운터와 Feedback 카운터의 편차를 비교하는 비교기입니다. 사용자는 편차의 한계를 지정한 후에 지정한 값보다 큰 편차가 발생하면 인터럽트를 발생하도록 할 수 있습니다.
cmmCmpErrGetConfig() 함수는 에러비교기의 설정된 환경을 읽어들입니다.

PARAMETER

▶ Axis: 축번호. 축번호는 상수값으로 [cmX1] 부터 0 번째 축을 기준 축으로 임의의 축을 설정할 수 있습니다.

▶ Tolerance : cmmCmpErrSetConfig 함수의 인자이며, 편차카운터의 한계값입니다. 편차카운트의 값이 이 값보다 크면 인터럽트가 발생됩니다.

▶ Tolerance : cmmCmpErrGetConfig 함수의 인자이며, 편차카운터의 한계값을 반환합니다.

▶ IsEnabled : cmmCmpErrSetConfig 함수의 인자이며, 에러비교기의 활성화 여부를 설정합니다.

Value	Meaning
0	에러비교기를 비활성화 합니다.
1	에러비교기를 활성화합니다.

▶ IsEnabled : cmmCmpErrGetConfig 함수의 인자이며, 에러비교기의 활성화 여부를 반환합니다.

Value	Meaning
0	에러비교기가 비활성화 상태입니다.
1	에러비교기를 활성화상태입니다.

RETURN VALUE

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

SEE ALSO

□ 사용자가 에러비교기에서 발생하는 인터럽트를 받기위해서는 인터럽트 마스크의 BIT10 을 1 로 하여야합니다.
인터럽트 마스크를 설정하는 방법은 cmmIntSetMask() 함수 설명편을 참조하시기 바랍니다.

NAME

cmmCmpGenSetConfig
cmmCmpGenGetConfig
– 범용(汎用) 비교기(比較器) 환경 설정 및 확인(確認)

INFORMATION

Compare Method
VC++/VB
BCB/Delphi/.NET
Level 8
위험 요소 없음

SYNOPSIS

❒ VT_I4 cmmCmpGenSetConfig
([in] VT_I4 Axis, [in] VT_I4 CmpSrc, [in] VT_I4 CmpMethod, [in] VT_I4 CmpAction, [in] VT_R8 CmpData)

❒ VT_I4 cmmCmpGenGetConfig
([in] VT_I4 Axis, [out] VT_PI4 CmpSrc, [out] VT_PI4 CmpMethod, [out] VT_PI4 CmpAction, [out] VT_PR8CmpData)

DESCRIPTION

cmmCmpGenSetConfig() 함수는 범용비교기(General comparator)의 환경을 설정합니다. 범용비교기는 비교대상 카운터를 사용자가 임의로 설정할 수 있도록 한 비교기입니다. 범용비교기에서는 비교조건, 조건만족시 동작 등의 환경도 사용자가 정의할 수 있습니다.
cmmCmpGenGetConfig() 함수는 범용비교기의 설정된 환경값을 읽어들입니다.

PARAMETER

▶ Axis: 축번호. 축번호는 상수값으로 [cmX1] 부터 0 번째 축을 기준 축으로 임의의 축을 설정할 수 있습니다.

▶ CmpSrc : cmmCmpGenSetConfig 함수의 인자이며, 비교대상 카운터를 설정합니다.

Value	Meaning
0 (cmCNT_COMM)	Command Counter
1 (cmCNT_FEED)	Feedback Counter
2 (cmCNT_DEV)	Deviation Counter (편차카운터)
3 (cmCNT_GEN)	General Counter

▶ CmpSrc : cmmCmpGenGetConfig 함수의 인자이며, 비교대상 카운터를 반환합니다.

Value	Meaning
0 (cmCNT_COMM)	Command Counter
1 (cmCNT_FEED)	Feedback Counter
2 (cmCNT_DEV)	Deviation Counter (편차카운터)
3 (cmCNT_GEN)	General Counter

▶ CmpMethod : cmmCmpGenSetConfig 함수의 인자이며, 비교조건을 설정합니다.

Value	Meaning
0 (cmDISABLE)	Disable comparator
1 (cmEQ_BIDIR)	CmpData = CmpSrc_Value (regardless of counting direction)
2 (cmEQ_PDIR)	CmpData = CmpSrc_Value (while counting up)
3 (cmEQ_NDIR)	CmpData = CmpSrc_Value (while counting down)

4 (cmLESS)	CmpData > CmpSrc_Value
5 (cmGREATER)	CmpData < CmpSrc_Value

▶ CmpMethod : cmmCmpGenGetConfig 함수의 인자이며, 비교조건을 반환합니다.

Value	Meaning
0 (cmDISABLE)	Disable comparator
1 (cmEQ_BIDIR)	CmpData = CmpSrc_Value (regardless of counting direction)
2 (cmEQ_PDIR)	CmpData = CmpSrc_Value (while counting up)
3 (cmEQ_NDIR)	CmpData = CmpSrc_Value (while counting down)
4 (cmLESS)	CmpData > CmpSrc_Value
5 (cmGREATER)	CmpData < CmpSrc_Value

▶ CmpAction : cmmCmpGenSetConfig 함수의 인자이며, 비교 조건이 성립되었을 때 취할 Action 을 설정합니다.

Value	Meaning
0 (cmEVNT_ONLY)	인터럽트만 발생
1 (cmEVNT_IS)	즉시 정지(停止) 및 인터럽트 발생
2 (cmEVNT_DS)	감속 후 정지(停止) 및 인터럽트 발생
3 (cmEVNT_SPDCHG)	속도 변경 및 인터럽트 발생

▶ CmpAction : cmmCmpGenGetConfig 함수의 인자이며, 비교 조건이 성립되었을 때 취할 Action 의 설정값을 반환합니다.

Value	Meaning
0 (cmEVNT_ONLY)	인터럽트만 발생
1 (cmEVNT_IS)	즉시 정지(停止) 및 인터럽트 발생
2 (cmEVNT_DS)	감속 후 정지(停止) 및 인터럽트 발생
3 (cmEVNT_SPDCHG)	속도 변경 및 인터럽트 발생

▶ CmpData : cmmCmpGenSetConfig 함수의 인자이며, 비교 기준 값입니다. 비교기는 이 값과 CmpSrc 로 지정된 카운터의 값을 비교합니다. CmpSrc 가 General coutner 인 경우는 이 값의 단위는 펄스수가 되며, 나머지에 대해서 CmpData 는 “Unit distance”에 의하여 정의되는 논리적 거리 단위입니다.

▶ CmpData : cmmCmpGenGetConfig 함수의 인자이며, 비교 기준 값을 반환합니다.

RETURN VALUE

Value	Meaning
음수	수행 실패. 자세한 내용은 ‘에러처리’ 편을 참고합니다
cmERR_NONE	수행 성공

12.2 위치비교출력(CMP)

㈜커미조아 모션컨트롤러는 각 축마다 위치비교출력 기능을 제공합니다. 위치비교출력 기능은 Command counter 또는 Position counter의 카운트값이 사용자가 지정한 조건에 만족되면 CMP출력핀을 통하여 트리거 펄스를 출력해주는 기능입니다.

이 기능을 사용하면 모션을 구동하면서 원하는 위치에서 외부기기에 하드웨어적인 트리거 신호를 제공할 수 있습니다. 특히 Machine Vision 시스템에서 유용하게 사용될 수 있습니다.

cmmCmpTrgSetOneData() 함수를 사용하면 하나의 비교데이터만 설정할 수 있습니다. 이 함수를 사용하여 여러 포인트에서 위치비교출력을 하려면 사용자가 Command나 Feedback 카운트값을 모니터링하거나 인터럽트 이벤트를 활용하여 다음 데이터 포인트를 설정할 시점을 결정하여야 합니다.

연속적인 포인트에서 위치비교출력 기능을 사용하려면 cmmCmpTrgSetOneData() 함수를 사용하는 것보다는 cmmCmpTrgContRegTable() 함수나 cmmCmpTrgContBuildTable() 함수와 같이 연속적인 위치비교출력 기능을 사용하는 것이 좋습니다.

12.2.1 함수 요약

위치비교 출력 기능과 관련된 함수들은 다음과 같습니다.

Summary of Functions
❒ VT_I4 cmmCmpTrgSetConfig ([in] VT_I4 Axis, [in] VT_I4 CmpSrc, [in] VT_I4 CmpMethod) 대상(對象) 모션 채널에 대해서, 위치 비교 출력 기능을 담당하는 비교기의 비교조건의 환경설정(環境設定)을 구성합니다.
❒ VT_I4 cmmCmpTrgGetConfig ([in] VT_I4 Axis, [out] VT_PI4 CmpSrc, [out] VT_PI4 CmpMethod) 대상(對象) 모션 채널에 대해서, 위치 비교 출력 기능을 담당하는 비교기의 비교조건에 대한 환경설정(環境設定)을 반환(返還)합니다.
❒ VT_I4 cmmCmpTrgSetOneData ([in] VT_I4 Axis, [in] VT_R8 Data) 대상(對象) 모션 채널에 대해서, 위치 비교 출력기에 1 회에 해당하는 비교 데이터 조건을 설정(設定)합니다.
❒ VT_I4 cmmCmpTrgGetCurData ([in] VT_I4 Axis, [out] VT_PR8 Data) 대상(對象) 모션 채널에 대해서, 위치 비교 출력기에 설정된 비교 데이터를 반환(返還)합니다.
❒ VT_I4 cmmCmpTrgContRegTable ([in] VT_I4 Axis, [in] VT_PR8 Buffer, [in] VT_I4 NumData) 대상(對象) 모션 채널에 대해서, 위치 비교 출력기에 임의의 연속적인 다중 위치 비교 데이터를 등록(登錄)합니다.
❒ VT_I4 cmmCmpTrgContBuildTable ([in] VT_I4 Axis, [in] VT_R8 StartData, [in] VT_R8 Interval, [in] VT_I4 NumData) 대상(對象) 모션 채널에 대해서, 위치 비교 출력기에 일정 간격을 가지는 연속적인 다중 위치 비교 데이터를 주어진 조건을 통하여 자동 생성하여, 등록(登錄)합니다.
❒ VT_I4 cmmCmpTrgContStart ([in] VT_I4 Axis) 대상(對象) 모션 채널에 대해서, 위치 비교 출력 기능을 활성화(活性化) 합니다.
❒ VT_I4 cmmCmpTrgContStop ([in] VT_I4 Axis) 대상(對象) 모션 채널에 대해서, 위치 비교 출력 기능을 비활성화(非活性化) 합니다.
❒ VT_I4 cmmCmpTrgContIsActive ([in] VT_I4 Axis, [out] VT_PI4 IsActive) 대상(對象) 모션 채널에 대해서 현재의 위치 비교 출력 상태를 반환(返還)합니다.

12.2.2 함수 설명

NAME

cmmCmpTrgSetConfig
cmmCmpTrgGetConfig
– 위치비교기(位置比較器) 조건 설정 (設定) 및 확인(確認)

INFORMATION

- Compare Method
- VC++/VB
- BCB/Delphi/.NET
- Level 8
- ☺ 위험 요소 없음

SYNOPSIS

❒ VT_I4 cmmCmpTrgSetConfig
([in] VT_I4 Axis, [in] VT_I4 CmpSrc, [in] VT_I4 CmpMethod)
❒ VT_I4 cmmCmpTrgGetConfig
([in] VT_I4 Axis, [out] VT_PI4 CmpSrc, [out] VT_PI4 CmpMethod)

DESCRIPTION

cmmCmpTrgSetConfig() 함수는 위치비교출력 기능을 담당하는 비교기의 비교조건을 설정합니다. 단, 비교데이터는 cmmCmpTrgSetOneData() 또는 cmmCmpTrgContRegTable(), CmpTrgContBuildTable() 함수를 사용하여 설정합니다.
cmmCmpTrgGetConfig() 함수는 위치비교출력 기능을 담당하는 비교기에 설정된 비교조건을 읽어들입니다.

PARAMETER

▶ Axis: 축번호. 축번호는 상수값으로 [cmX1] 부터 0 번째 축을 기준 축으로 임의의 축을 설정할 수 있습니다.

▶ CmpSrc : cmmCmpTrgSetConfig 함수의 인자이며, 비교 대상 카운터를 설정합니다.

Value	Meaning
0 또는 cmCNT_COMM	Command Counter
1 또는 cmCNT_FEED	Feedback Counter
2 또는 cmCNT_DEV	Deviation Counter : Command 와 Feedback counter 의 편차 카운터
3 또는 cmCNT_GEN	General Counter : 사용자의 정의에 따라 여러가지 용도로 사용될 수 있는 카운터

▶ CmpSrc : cmmCmpTrgGetConfig 함수의 인자이며, 비교 대상 카운터를 반환합니다.

Value	Meaning
0 또는 cmCNT_COMM	Command Counter
1 또는 cmCNT_FEED	Feedback Counter
2 또는 cmCNT_DEV	Deviation Counter : Command 와 Feedback counter 의 편차 카운터
3 또는 cmCNT_GEN	General Counter : 사용자의 정의에 따라 여러가지 용도로 사용될 수 있는 카운터

▶ CmpMethod : cmmCmpTrgSetConfig 함수의 인자이며, CMP 신호 출력 조건을 설정합니다. 즉, 비교 대상 카운트값과 사용자가 지정한 데이터 값을 어떻게 비교하여 CMP 신호 출력을 발생할지에 대한 조건을 결정합니다.

Value	Meaning
0 (cmDISABLE)	Disable comparator
1 (cmEQ_BIDIR)	CmpData = CmpSrc_Value (regardless of counting direction)
2 (cmEQ_PDIR)	CmpData = CmpSrc_Value (while counting up)

3 (cmEQ_NDIR)	CmpData = CmpSrc_Value (while counting down)
4 (cmLESS)	CmpData > CmpSrc_Value
5 (cmGREATER)	CmpData < CmpSrc_Value

▶ CmpMethod : cmmCmpTrgGetConfig 함수의 인자이며, CMP 신호 출력 조건을 반환합니다.

Value	Meaning
0 (cmDISABLE)	Disable comparator
1 (cmEQ_BIDIR)	CmpData = CmpSrc_Value (regardless of counting direction)
2 (cmEQ_PDIR)	CmpData = CmpSrc_Value (while counting up)
3 (cmEQ_NDIR)	CmpData = CmpSrc_Value (while counting down)
4 (cmLESS)	CmpData > CmpSrc_Value
5 (cmGREATER)	CmpData < CmpSrc_Value

RETURN VALUE

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

NAME

cmmCmpTrgSetOneData
– 위치비교기(位置比較器) 에 1 회의 비교데이터 설정

INFORMATION

Compare Method

VC++/VB

BCB/Delphi/.NET

Level 8

위험 요소 없음

SYNOPSIS

❒ VT_I4 cmmCmpTrgSetOneData ([in] VT_I4 Axis, [in] VT_R8 Data)

DESCRIPTION

위치비교출력기에 1 회의 비교데이터를 설정합니다. 모션컨트롤러는 카운터 의 값을 cmmCmpTrgSetOneData() 함수의 Data 값과 비교하여 비교조건 을 충족하면 CMP 단자로 출력 을 내보냅니다.

PARAMETER

▶ Axis: 축번호. 축번호는 상수값으로 [cmX1] 부터 0 번째 축을 기준 축으로 임의의 축을 설정할 수 있습니다.

▶ Data : 비교 대상 레퍼런스(Reference)값. 비교기는 지정한 카운트값을 이 값과 비교하여 CMP 출력을 내보냅니다. "Unit distance"에 의해 정의되는 논리적 거리 단위를 사용하며, 절대좌표값으로 설정하여야 합니다.

REFERENCE

□ 논리적 거리 단위는 cmmCfgSetUnitDist() 함수에 의해 결정됩니다.

□ 이 함수를 사용하면 하나의 비교데이터만 설정할 수 있습니다. 이 함수를 사용하면서 여러 포인트에서 위치비교출력을 하려면 사용자가 Command 나 Feedback 카운트값을 모니터링하거나 인터럽트 이벤트를 활용하여 다음 데이터 포인트를 설정할 시점을 결정하여야 합니다. 따라서 여러 개의 포인트에서 위치비교출력을 연속적으로 사용하려면 cmmCmpTrgContRegTable() 이나 cmmCmpTrgContBuildTable() 함수를 사용하여 데이터를 등록하고 cmmCmpTrgContStart() 함수에 의해 시작되는 "연속적인 위치비교출력" 기능을 사용하는 것이 바람직합니다.

□ 이 함수는 cmmCmpTrgContStart() 함수나 cmmCmpTrgContStop() 함수와 함께 사용되는 함수가 아닙니다.

RETURN VALUE

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

EXAMPLE

□ 아래의 예제에서는 X 축의 Command(cmCNT_COMM) 위치값이 10000 이 되면 CMP 트리거 펄스를 출력하도록 하는 것입니다. 이때 (+)방향쪽으로 이동시에만 트리거펄스가 출력되고 (-)방향으로 이동할 때는 출력하지 않도록(cmEQ_PDIR) 합니다.

C/C++

```
CmpTrgSetConfig(cmX1, cmCNT_COMM, cmEQ_PDIR);
CmpTrgSetOneData (cmX1, 10000);
```

Visual Basic

```
Call CmpTrgSetConfig(cmX1, cmCNT_COMM, cmEQ_PDIR)
Call CmpTrgSetOneData (cmX1, 10000)
```

Delphi

```
CmpTrgSetConfig(cmX1, cmCNT_COMM, cmEQ_PDIR);
CmpTrgSetOneData (cmX1, 10000);
```

NAME

cmmCmpTrgGetCurData
– 위치비교기 (位置比較器) 에 설정된 데이터 확인(確認)

INFORMATION

Compare Method
VC++/VB
BCB/Delphi/.NET
Level 8
위험 요소 없음

SYNOPSIS

❒ VT_I4 cmmCmpTrgGetCurData ([in] VT_I4 Axis, [out] VT_PR8 Data)

DESCRIPTION

위치비교출력기에 현재 설정된 비교데이터를 반환합니다. 이 함수는 cmmCmpTrgSetOneData()에 의해 설정된 비교데이터를 확인(確認)할 때는 물론이고, "연속적인 위치비교출력" 기능을 사용할 때는 현재 위치비교출력기에 설정된 비교데이터를 확인(確認)할 수 있습니다.

PARAMETER

▶ Axis: 축번호. 축번호는 상수값으로 [cmX1] 부터 0 번째 축을 기준 축으로 임의의 축을 설정할 수 있습니다.

▶ Data : 위치비교출력기에 현재 설정된 비교데이터. 이 값은 "Unit distance"에 의해 정의되는 논리적 거리 단위를 사용하는 절대좌표값입니다.

RETURN VALUE

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

NAME

cmmCmpTrgContRegTable
- 연속적인 비교 위치(比較) 데이터 등록 (登錄)

INFORMATION

- Compare Method
- VC++/VB
- BCB/Delphi/.NET
- Level 8
- 위험 요소 없음

SYNOPSIS

❒ VT_I4 cmmCmpTrgContRegTable ([in] VT_I4 Axis, [in] VT_PR8 Buffer, [in] VT_I4 NumData)

DESCRIPTION

연속적인 위치 비교 출력 기능을 사용하기 위해서 임의의 연속적인 위치 데이터를 등록 (登錄) 합니다.

PARAMETER

▶ Axis: 축번호. 축번호는 상수값으로 [cmX1] 부터 0 번째 축을 기준 축으로 임의의 축을 설정할 수 있습니다.

▶ Buffer : 연속적인 위치 데이터를 담고 있는 버퍼(배열). 이 버퍼에 설정되는 위치데이터는 "Unit distance"에 의해 정의되는 논리적 거리 단위를 사용하며, 절대좌표값으로 설정하여야 합니다.

▶ NumData : 버퍼에 담겨있는 위치 데이터의 수

REFERENCE

□ 논리적 거리 단위는 cmmCfgSetUnitDist() 함수에 의해 결정됩니다.

□ 이 함수는 위치데이터만 등록하는 것이며 CmpTrgContStart() 함수를 호출해야만 등록한 데이터들이 적용되는 연속적인 위치비교출력 기능이 시작됩니다.

□ 아래의 예제에서는 X 축을 0 에서 50000 좌표로 이동시키면서 Command counter 값이 1000, 5000, 10000, 20000, 30000 이 될 때 각각 트리거 펄스가 출력되도록 하는 예입니다. 따라서 연속적으로 총 5 회의 트리거 펄스가 출력되게 됩니다.

EXAMPLE

C/C++

```
double fDataBuf [5]={1000, 5000, 10000, 20000, 30000};
...
cmmCmpTrgSetConfig(cmX1, cmCNT_COMM, cmEQ_PDIR);

// 연속적인 위치데이터 등록 //
cmmCmpTrgContRegTable(cmX1, fDataBuf, 5);

// 연속적인 위치 비교 기능 시작 //
cmmCmpTrgContStart(cmX1);

...
/* X 축을 50000 포인트로 이동한다. 이동하면서 지정한 위치마다 CMP 트리거 펄스가 출력되게 된다. */
cmmSxMoveTo(cmX1, 50000, cmFALSE);

cmmCmpTrgContStop(cmX1) ; // 연속위치비교 기능 종료
```

```
Visual Basic

Dim fDataBuf(5) As Double
fDataBuf(0) =1000
fDataBuf(1) = 5000
fDataBuf(2) = 10000
fDataBuf(3) = 20000
fDataBuf(4) = 30000
…

Call cmmCmpTrgSetConfig(cmX1, cmCNT_COMM, cmEQ_PDIR)

‘ 연속적인 위치데이터 등록
Call cmmCmpTrgContRegTable(cmX1, fDataBuf, 5)

‘ 연속적인 위치 비교 기능 시작
Call cmmCmpTrgContStart(cmX1)

…
‘ X 축을 50000 포인트로 이동한다. 이동하면서 지정한 위치마다 CMP 트리거 펄스가 출력되게 된다.
Call cmmSxMoveTo(cmX1, 50000, cmFALSE)
Call cmmCmpTrgContStop(cmX1) ‘ 연속위치비교 기능 종료
```

```
Delphi

Const

        fDataBuf : Array[0..4] of Double = (1000,5000,10000,20000,30000);

begin

cmmCmpTrgSetConfig(cmX1, cmCNT_COMM, cmEQ_PDIR);

// 연속적인 위치데이터 등록 //
cmmCmpTrgContRegTable(cmX1,@fDataBuf, 5);

// 연속적인 위치 비교 기능 시작 //
cmmCmpTrgContStart(cmX1);

…

// X 축을 50000 포인트로 이동한다. 이동하면서 지정한 위치마다 CMP 트리거 펄스가 출력되게 된다.
cmmSxMoveTo(cmX1, 50000, cmFALSE);

cmmCmpTrgContStop(cmX1) ; // 연속위치비교 기능 종료

end;
```

RETURN VALUE

Value	Meaning
음수	수행 실패. 자세한 내용은 ‘에러처리’ 편을 참고합니다
cmERR_NONE	수행 성공

NAME

cmmCmpTrgContBuildTable
– 일정 간격 (一定間隔) 의 비교위치(比較位置) 데이터 등록

INFORMATION

- Compare Method
- VC++/VB
- BCB/Delphi/.NET
- Level 8
- ☺ 위험 요소 없음

SYNOPSIS

❒ VT_I4 cmmCmpTrgContBuildTable

([in] VT_I4 Axis, [in] VT_R8 StartData, [in] VT_R8 Interval, [in] VT_I4 NumData)

DESCRIPTION

연속적인 위치 비교 출력 기능을 사용하기 위해서 일정한 위치 간격을 가지는 연속적인 위치 데이터를 자동으로 생성하여 등록 (登錄) 하도록 합니다.
이 함수는 일정한 위치 간격으로 CMP 출력을 내보낼 때 cmmCmpTrgContRegTable() 함수 대신에 사용할 수 있습니다.

PARAMETER

▶ Axis: 축번호. 축번호는 상수값으로 [cmX1] 부터 0 번째 축을 기준 축으로 임의의 축을 설정할 수 있습니다.

▶ StartData : 시작 위치값. 이 값은 "Unit distance"에 의해 정의되는 논리적 거리 단위를 사용하는 절대좌표값으로 설정합니다.

▶ Interval : 위치 간격. 이 값은 "Unit distance"에 의해 정의되는 논리적 거리 단위로 설정합니다.

▶ NumData : 자동생성되는 총 데이터 수

RETURN VALUE

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

NAME

cmmCmpTrgContStart
– 연속 위치 비교 출력 (連續位置比較出力) 기능 시작(始作)

INFORMATION

Compare Method

VC++/VB

BCB/Delphi/.NET

Level 8

위험 요소 없음

SYNOPSIS

❒ VT_I4 cmmCmpTrgContStart ([in] VT_I4 Axis)

DESCRIPTION

연속적인 위치 비교 출력 기능을 시작합니다. cmmCmpTrgContRegTable() 또는 cmmCmpTrgContBuildTable() 함수를 이용하여 등록된 연속적인 위치데이터를 비교기에 순차적으로 자동로딩하면서 연속적인 위치비교 출력기능을 수행합니다.
연속적인 위치비교 출력 기능은 현재 비교기에 로드된 비교조건이 만족되어 CMP 출력이 나가게 되면 인터럽트가 발생되어 드라이버 프로그램에서 사용자가 등록한 다음 데이터를 비교기에 자동으로 로드하도록 하는 기능입니다.

PARAMETER

▶ Axis: 축번호. 축번호는 상수값으로 [cmX1] 부터 0 번째 축을 기준 축으로 임의의 축을 설정할 수 있습니다.

REFERENCE

□ 연속적인 위치비교 출력 기능이 시작된 이후에 cmmIntSetMask() 함수를 이용하여 BIT12 를 언마스크(0 으로 설정)하면 인터럽트를 받을 수 없어서 연속적인 위치비교출력 기능이 진행되지 않습니다.

참고하십시요 !

사용자가 등록한 모든 위치 데이터에 대하여 CMP 출력이 완료되었으면 cmmCmpTrgContStop() 함수를 호출하여 중지하지 않는한 처음 데이터부터 다시 비교기에 로드되어 연속적인 비교출력이 자동으로 재개됩니다.
그러나 현재 로드된 비교조건이 만족되기 전까지는 다음 위치데이터가 로드되지 않는다는 점에 유의하시기 바랍니다. 즉, 1000, 5000, 10000, 15000 의 연속적인 위치데이터를 등록한 후 모션이 10000 위치까지 이동 후 다시 0 위치로 복귀하고 다시 10000 위치로 이동했을 때 처음 10000 위치로 이동시에는 1000, 5000, 10000 의 위치에서 CMP 출력이 나가지만 두번째 10000 위치로 이동시에는 CMP 출력이 발생하지 않습니다. 이유는 현재 비교기에 로드된 위치데이터가 15000 인데 이 조건이 만족되지 않았으므로 계속해서 비교기에 이 데이터가 남아있기 때문입니다.

RETURN VALUE

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

NAME

cmmCmpTrgContStop
– 연속 위치 비교 출력(連續位置比較出力) 기능 종료(終了)

INFORMATION

- Compare Method
- VC++/VB
- BCB/Delphi/.NET
- Level 8
- 위험 요소 없음

SYNOPSIS

❒ VT_I4 cmmCmpTrgContStop ([in] VT_I4 Axis)

DESCRIPTION

연속적인 위치 비교 출력 기능을 종료합니다.

PARAMETER

▶ Axis: 축번호. 축번호는 상수값으로 [cmX1] 부터 0 번째 축을 기준 축으로 임의의 축을 설정할 수 있습니다.

RETURN VALUE

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

NAME

cmmCmpTrgContIsActive
– 해당축의 위치 비교 출력(位置比較出力) 상태 확인(確認)

INFORMATION

Compare Method
VC++/VB
BCB/Delphi/.NET
Level 8
위험 요소 없음

SYNOPSIS

❒ VT_I4 cmmCmpTrgContIsActive ([in] VT_I4 Axis, [out] VT_PI4 IsActive)

DESCRIPTION

지정된 축의 위치 비교 출력 상태를 반환합니다.

PARAMETER

▶ Axis: 축번호. 축번호는 상수값으로 [cmX1] 부터 0 번째 축을 기준 축으로 임의의 축을 설정할 수 있습니다.

▶ IsActive : 대상 축(채널)의 위치 비교 출력 상태를 반환합니다.

Value	Meaning
cmFALSE	위치 비교 출력 상태가 비활성화 되었습니다.
cmTRUE	위치 비교 출력 상태가 활성화 되었습니다.

RETURN VALUE

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

12.3 예약 위치비교출력(CMP QUE)

㈜커미조아 모션컨트롤러는 각 축마다 위치비교출력 기능을 제공합니다. 위치비교출력 기능은 Command counter 또는 Position counter의 카운트값이 사용자가 지정한 조건에 만족되면 CMP출력핀을 통하여 트리거 펄스를 출력해주는 기능입니다.

여기에 ㈜커미조아 에서는 원래의 비교데이터 설정 이후 모션 구동을 통하여 위치비교출력을 수행하던 방식에서 발전하여 구동이 진행중인 상황에 추가로 위치비교지점을 설정 가능하도록 CMP QUE 기능을 추가로 구현하였습니다.

12.3.1 함수 요약

예약 위치비교 출력 기능과 관련된 함수들은 다음과 같습니다.

Summary of Functions
❒ VT_I4 cmmCmpQue_SetEnable ([in] VT_I4 Axis, [in] VT_I4 IsEnable) 대상(對象) 모션 채널에 대해서, 예약 위치 비교 출력 기능을 담당하는 예약 큐의 사용 여부를 설정(設定)합니다.
❒ VT_I4 cmmCmpQue_GetEnable ([in] VT_I4 Axis, [out] VT_PI4 IsEnabled) 대상(對象) 모션 채널에 대해서, 예약 위치 비교 출력 기능을 담당하는 예약 큐의 사용 여부 에 대한 설정(設定)값을 반환(返還)합니다.
❒ VT_I4 cmmCmpQue_SetQueSize ([in] VT_I4 Axis, [in] VT_I4 QueSize) 대상(對象) 모션 채널에 대해서, 예약 큐의 크기를 설정(設定)합니다.
❒ VT_I4 cmmCmpQue_GetQueSize ([in] VT_I4 Axis [out] VT_PI4 QueSize) 대상(對象) 모션 채널에 대해서, 예약 큐의 크기를 반환(返還)합니다.
❒ VT_I4 cmmCmpQue_Enque ([in] VT_I4 Axis, [in] VT_I4 CmpSrc, [in] VT_I4 CmpMethod, [in]VT_I4 CmpData) 대상(對象) 모션 채널에 대해서, 예약 큐에 비교할 위치 데이터를 등록(登錄)합니다.
❒ VT_I4 cmmCmpQue_GetEnqueCnt ([in] VT_I4 Axis, [out] VT_PI4 EnqueCnt) 대상(對象) 모션 채널에 대해서, 예약 큐에 등록되었던 모든 데이터에 대한 개수를 반환(返還)합니다.
❒ VT_I4 cmmCmpQue_GetOutCnt ([in] VT_I4 Axis, [out] VT_PI4 OutCnt) 대상(對象) 모션 채널에 대해서, 예약 큐에 등록되어 위치 비교 후 출력된 데이터에 대한 개수를 반환(返還)합니다.
❒ VT_I4 cmmCmpQue_SetOutCnt ([in] VT_I4 Axis, [in] VT_I4 OutCnt) 대상(對象) 모션 채널에 대해서, 예약 큐에 등록되어 위치 비교 후 출력된 데이터에 대한 개수를 임의로 설정(設定)합니다.
❒ VT_I4 cmmCmpQue_SetLtcLinkMode ([in] VT_I4 Axis, [in] VT_I4 Enable, [in] VT_I4 SrcLtcCnt, [in] VT_I4 CmpSrc, [in] VT_I4 CmpMethod, [in] VT_I4 Offset) 대상(對象) 모션 채널에 대해서 래치(LTC)신호와 위치 비교 출력 신호를 서로 Link 하여 예약 큐에 해당 위치 데이터를 등록(登錄) 할 수 있도록 설정(設定)합니다.
❒ VT_I4 cmmCmpQue_GetLtcLinkMode ([in] VT_I4 Axis, [out] VT_PI4 Enable, [out] VT_PI4 SrcLtcCnt, [out] VT_PI4 CmpSrc, [out] VT_PI4 CmpMethod, [out] VT_PI4 Offset) 대상(對象) 모션 채널에 대해서 래치(LTC)신호와 Link 된 위치 비교 출력 신호 설정 상태를 반환(返還) 합니다.

12.3.2 함수 설명

NAME

cmmCmpQue_SetEnable
cmmCmpQue_GetEnable
– 예약 큐 사용 여부 설정 (設定) 및 확인(確認)

INFORMATION

Compare Method
VC++/VB
BCB/Delphi/.NET
Level 8
위험 요소 없음

SYNOPSIS

❒ VT_I4 cmmCmpQue_SetEnable
([in] VT_I4 Axis, [in] VT_I4 IsEnable)
❒ VT_I4 cmmCmpQue_GetEnabled
([in] VT_I4 Axis, [out] VT_PI4 IsEnabled)

DESCRIPTION

cmmCmpQue_SetEnable() 함수는 위치비교출력 시 예약 큐의 기능을 활성화/비활성화 합니다.
이 함수를 사용하여 예약 큐가 활성화 되면 모션 구동 중에도 위치비교기에 순차적으로 비교지점을 설정하실 수 있습니다. 단, 이 함수를 사용하기 전에 미리 큐의 크기를 설정하여 주는 것이 안전합니다.
cmmCmpQue_GetEnable() 함수는 위치비교출력 시 예약 큐의 설정 상태를 확인합니다.

PARAMETER

▶ Axis: 축(채널) 번호. 축번호는 상수값으로 [cmX1] 부터 0 번째 채널을 기준 채널로 임의의 채널을 설정할 수 있습니다.

▶ IsEnable: cmmCmpQue_SetEnable 함수의 인자이며, 예약 큐의 사용여부를 설정합니다.

Value	Meaning
0 또는 cmFALSE	예약 큐 사용 안함
1 또는 cmTRUE	예약 큐 사용

▶ IsEnabled : cmmCmpQue_GetEnable 함수의 인자이며, 예약 큐의 사용여부를 반환합니다.

Value	Meaning
0 또는 cmFALSE	예약 큐 사용 안함
1 또는 cmTRUE	예약 큐 사용

RETURN VALUE

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

EXAMPLE

□ cmmCmpQue_SetQueSize 예제 참조

NAME

cmmCmpQue_SetQueSize
cmmCmpQue_GetQueSize
– 예약 큐 크기 설정 (設定) 및 확인(確認)

INFORMATION

Compare Method
VC++/VB
BCB/Delphi/.NET
Level 8
위험 요소 없음

SYNOPSIS

❒ VT_I4 cmmCmpQue_SetQueSize
([in] VT_I4 Axis, [in] VT_I4 QueSize)
❒ VT_I4 cmmCmpQue_GetQueSize
([in] VT_I4 Axis, [out] VT_PI4 QueSize)

DESCRIPTION

cmmCmpQue_SetQueSize () 함수는 예약 위치비교출력 기능에 사용할 예약 큐의 크기를 설정합니다.
cmmCmpQue_GetQueSize() 함수는 예약 위치비교출력 기능에 사용되는 예약 큐의 크기를 확인합니다.

PARAMETER

▶ Axis: 축(채널) 번호. 축번호는 상수값으로 [cmX1] 부터 0 번째 채널을 기준 채널로 임의의 채널을 설정할 수 있습니다.

▶ QueSize: cmmCmpQue_SetQueSize, cmmCmpQue_GetQueSize 함수의 인자이며, 예약 큐의 크기와 일치합니다. 이 값이 0 인 경우 예약 큐가 없는 것으로 간주되어 예약 위치비교출력 기능은 동작하지 않습니다.

RETURN VALUE

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

EXAMPLE

□ 아래의 예제에서는 X 축에 대하여 CMP Que 의 크기를 지정한 후 기능을 사용하도록 설정하는 것입니다. 이 함수를 사용하면 입력된 Que 의 개수 및 출력된 위치비교 개수가 모두 0 이 되어 큐가 초기화되는 효과가 적용됩니다

```
C/C++
#define DEFAULT_QUE_SIZE                100

…

cmmCmpQue_SetQueSize(cmX1,DEFAULT_QUE_SIZE);
cmmCmpQue_SetEnable(cmX1,cmTRUE);
```

NAME

cmmCmpQue_Enque
– 예약 큐에 위치비교(位置比較) 조건 설정(設定)

INFORMATION

Compare Method

VC++/VB

BCB/Delphi/.NET

Level 8

위험 요소 없음

SYNOPSIS

❒ VT_I4 cmmCmpQue_Enque

([in] VT_I4 Axis, [in] VT_I4 CmpSrc, [in] VT_I4 CmpMethod, [in] VT_I4 CmpData)

DESCRIPTION

cmmCmpQue_Enque() 함수는 예약 큐에 위치비교 조건 및 비교데이터를 등록합니다. 여기에 등록된 위치비교 데이터는 최초의 데이터부터 순차적으로 비교됩니다.

PARAMETER

▶ Axis: 축(채널) 번호. 축번호는 상수값으로 [cmX1] 부터 0 번째 채널을 기준 채널로 임의의 채널을 설정할 수 있습니다.

▶ CmpSrc : cmmCmpQue_Enque 함수의 인자이며, 비교 대상 카운터를 설정합니다.

Value	Meaning
0 또는 cmCNT_COMM	Command Counter
1 또는 cmCNT_FEED	Feedback Counter
2 또는 cmCNT_DEV	Deviation Counter : Command 와 Feedback counter 의 편차 카운터
3 또는 cmCNT_GEN	General Counter : 사용자의 정의에 따라 여러가지 용도로 사용될 수 있는 카운터

▶ CmpMethod : cmmCmpQue_Enque 함수의 인자이며, CMP 신호 출력 조건을 설정합니다. 즉, 비교 대상 카운트값과 사용자가 지정한 데이터 값을 어떻게 비교하여 CMP 신호 출력을 발생할지에 대한 조건을 결정합니다.

Value	Meaning
0 (cmDISABLE)	Disable comparator
1 (cmEQ_BIDIR)	CmpData = CmpSrc_Value (regardless of counting direction)
2 (cmEQ_PDIR)	CmpData = CmpSrc_Value (while counting up)
3 (cmEQ_NDIR)	CmpData = CmpSrc_Value (while counting down)
4 (cmLESS)	CmpData > CmpSrc_Value
5 (cmGREATER)	CmpData < CmpSrc_Value

▶ CmpData: cmmCmpQue_Enque 함수의 인자이며, CMP 신호 출력 위치를 설정합니다.

RETURN VALUE

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

EXAMPLE

□ 아래의 예제에서는 X 축에 대하여 CMP 큐의 크기 만큼 반복하면서 매 1000pulse 위치마다 비교지점을 설정하는 것입니다.
이 때 비교되는 카운터는 Command Counter 가 되며, 증가하는 방향(cmEQ_PDIR)으로 정확히 일치하는 경우에만 CMP 가 출력되도록 지정하였습니다.

```
C/C++
int i = 0;

for(i = 0; i < DEFAULT_QUE_SIZE; i++) {
        cmmCmpQue_Enque( cmX1, cmCNT_COMM, cmEQ_PDIR, 1000 * i );
}
```

NAME

cmmCmpQue_GetEnqueCnt
- 예약 큐에 등록되었던 비교(比較)카운트 개수 확인(確認)

INFORMATION

- Compare Method
- VC++/VB
- BCB/Delphi/.NET
- Level 8
- 위험 요소 없음

SYNOPSIS

❒ VT_I4 cmmCmpQue_GetEnqueCnt
([in] VT_I4 Axis, [out] VT_PI4 EnqueCnt)

DESCRIPTION

cmmCmpQue_GetEnqueCnt() 함수는 예약 큐에 등록된 전체 위치비교 카운트의 개수를 확인합니다. 여기에서 확인되는 카운트 수는 큐에 등록된 후 출력이 된 비교 카운트 및 아직 출력이 되지 않고 큐에 남아있는 비교 카운트 모두를 포함합니다.

PARAMETER

▶ Axis: 축(채널) 번호. 축번호는 상수값으로 [cmX1] 부터 0 번째 채널을 기준 채널로 임의의 채널을 설정할 수 있습니다.

▶ EnqueCnt: cmmCmpQue_GetEnqueCnt 함수의 인자이며, 큐에 등록되었던 비교 카운터 개수를 반환합니다.

RETURN VALUE

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

EXAMPLE

□ 아래의 예제에서는 X 축에 대하여 CMP Que 에 입력되었던 위치비교지점의 개수를 확인하고 있습니다.

```
C/C++
long nEnqueCnt;

…

cmmCmpQue_GetEnqueCnt(cmX1, &nEnqueCnt);
```

NAME

cmmCmpQue_GetOutCnt
– 예약 큐에 등록 후 출력된 비교(比較) 카운트 개수 확인(確認)

INFORMATION

- Compare Method
- VC++/VB
- BCB/Delphi/.NET
- Level 8
- 위험 요소 없음

SYNOPSIS

❒ VT_I4 cmmCmpQue_GetOutCnt
([in] VT_I4 Axis, [out] VT_PI4 OutCnt)

DESCRIPTION

cmmCmpQue_GetOutCnt() 함수는 예약 큐에 등록 후 출력된 위치비교 카운트의 개수를 확인합니다. 여기에서 확인되는 카운트 수는 큐에 등록된 후 출력이 된 비교 카운트만을 포함합니다.

PARAMETER

▶ Axis: 축(채널) 번호. 축번호는 상수값으로 [cmX1] 부터 0 번째 채널을 기준 채널로 임의의 채널을 설정할 수 있습니다.

▶ OutCnt: cmmCmpQue_GetOutCnt 함수의 인자이며, 큐에서 출력된 비교 카운트 개수를 반환합니다.

RETURN VALUE

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

EXAMPLE

□ 아래의 예제에서는 X 축에 대하여 CMP Que 에 입력된 후 비교되어 출력된 위치비교지점의 개수를 확인하고 있습니다.

```
C/C++
long nOutCnt;

…

cmmCmpQue_GetOutCnt(cmX1, &nOutCnt);
```

NAME

cmmCmpQue_SetOutCnt
– 예약 큐에 등록 후 출력된 비교(比較) 카운트 개수 설정(設定)

INFORMATION

Compare Method

VC++/VB

BCB/Delphi/.NET

Level 8

위험 요소 없음

SYNOPSIS

❒ VT_I4 cmmCmpQue_SetOutCnt
([in] VT_I4 Axis, [in] VT_I4 OutCnt)

DESCRIPTION

cmmCmpQue_SetOutCnt() 함수는 예약 큐에 등록 후 출력된 위치비교 카운트의 개수를 설정합니다. 여기에서 설정되는 카운트 수는 큐에 등록된 후 출력이 된 비교 카운트만을 포함합니다.

PARAMETER

▶ Axis: 축(채널) 번호. 축번호는 상수값으로 [cmX1] 부터 0 번째 채널을 기준 채널로 임의의 채널을 설정할 수 있습니다.

▶ OutCnt: cmmCmpQue_SetOutCnt 함수의 인자이며, 큐에서 출력된 비교 카운트 개수를 지정합니다.

RETURN VALUE

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

EXAMPLE

□ 아래의 예제에서는 X 축에 대하여 CMP Que 에 입력된 후 비교되어 출력된 위치비교지점의 개수를 0 으로 설정하고 있습니다. 이는 입력된 Que 의 개수는 그대로 두되 출력된 위치비교개수가 0 이 되어 다시 처음부터 비교되는 효과를 얻을 수 있습니다. cmmCmpQue_SetEnable() 함수를 호출하면 입력된 Que 의 개수 및 출력된 위치비교 개수가 모두 0 이 되어 큐가 초기화되는 효과가 적용됩니다.

C/C++

```
cmmCmpQue_SetOutCnt(cmX1, 0);
```

NAME

cmmCmpQue_SetLtcLinkMode
cmmCmpQue_GetLtcLinkMode
– 래치(LTC) 입력을 통한 위치비교(位置比較) 연계 조건 설정 (設定) 및 확인(確認)

INFORMATION

Compare Method
VC++/VB
BCB/Delphi/.NET
Level 8
위험 요소 없음

SYNOPSIS

❒ VT_I4 cmmCmpQue_SetLtcLinkMode
([in] VT_I4 Axis, [in] VT_I4 Enable, [in] VT_I4 SrcLtcCnt, [in] VT_I4 CmpSrc, [in] VT_I4 CmpMethod, [in] VT_I4 Offset)

❒ VT_I4 cmmCmpQue_GetLtcLinkMode
([in] VT_I4 Axis, [out] VT_PI4 Enable, [out] VT_PI4 SrcLtcCnt, [out] VT_PI4 CmpSrc, [out] VT_PI4 CmpMethod, [out] VT_PI4 Offset)

DESCRIPTION

cmmCmpQue_SetLtcLinkMode() 함수는 래치(Latch : LTC)입력으로부터 위치비교기의 예약 큐에 대한 위치비교 조건 및 비교데이터를 연계 등록하는 방법에 대하여 설정합니다.
이 조건이 설정되면 대상 축에 대한 LTC 입력이 들어오면 입력지점으로부터 정해진 오프셋(Offset)만큼의 이후에 자동으로 위치비교출력이 수행됩니다.
cmmCmpQue_GetLtcLinkMode() 함수는 이에 대한 설정이 정상적으로 등록되었는지 각 항에 대하여 확인하는 함수입니다.

PARAMETER

▶ Axis: 축(채널) 번호. 축번호는 상수값으로 [cmX1] 부터 0 번째 채널을 기준 채널로 임의의 채널을 설정할 수 있습니다.

▶ Enable: LTC 입력으로부터 위치비교출력을 활성화할 것인지 설정합니다.

Value	Meaning
0 또는 cmFALSE	LTC LINK 사용 안함
1 또는 cmTRUE	LTC LINK 사용

▶ SrcLtcCnt : LTC 입력 대상 카운터를 설정합니다.

Value	Meaning
0 또는 cmCNT_COMM	Command Counter
1 또는 cmCNT_FEED	Feedback Counter
2 또는 cmCNT_DEV	Deviation Counter : Command 와 Feedback counter 의 편차 카운터
3 또는 cmCNT_GEN	General Counter : 사용자의 정의에 따라 여러가지 용도로 사용될 수 있는 카운터

▶ CmpSrc : 비교 대상 카운터를 설정합니다.

Value	Meaning
0 또는 cmCNT_COMM	Command Counter
1 또는 cmCNT_FEED	Feedback Counter
2 또는 cmCNT_DEV	Deviation Counter : Command 와 Feedback counter 의 편차 카운터
3 또는 cmCNT_GEN	General Counter : 사용자의 정의에 따라 여러가지 용도로 사용될 수 있는 카운터

▶ CmpMethod : cmmCmpQue_Enque 함수의 인자이며, CMP 신호 출력 조건을 설정합니다. 즉, 비교 대상 카운트값과 사용자가 지정한 데이터 값을 어떻게 비교하여 CMP 신호 출력을 발생할지에 대한 조건을 결정합니다.

Value	Meaning
0 (cmDISABLE)	Disable comparator
1 (cmEQ_BIDIR)	CmpData = CmpSrc_Value (regardless of counting direction)
2 (cmEQ_PDIR)	CmpData = CmpSrc_Value (while counting up)
3 (cmEQ_NDIR)	CmpData = CmpSrc_Value (while counting down)
4 (cmLESS)	CmpData > CmpSrc_Value
5 (cmGREATER)	CmpData < CmpSrc_Value

▶ Offset: LTC 신호 입력 후 지연 출력 될 위치 비교 신호의 오프셋을 지정합니다. LTC 신호가 입력된 시점으로부터 이 인자에 지정된 Offset 만큼의 펄스 출력 이후에 CMP 신호를 출력합니다.

RETURN VALUE

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

EXAMPLE

□ 아래의 예제에서는 X 축에 대하여 Command Counter 값을 LTC 로 입력받아 입력 받은 지점에서 50000 pulse 위치 이후에서 위치비교(CMP)를 출력하도록 설정하였습니다.
만약 LTC 입력위치가 10,000 인 경우, 해당 축에서 실제 출력되는 CMP 신호는 60,000 의 위치에서 발생하게 됩니다.

```
C/C++
long nOffset;

nOffset = 50000;

cmmCmpQue_SetLtcLinkMode(cmX1, TRUE, cmCNT_COMM, cmCNT_COMM, cmEQ_BIDIR, nOffset);
```

12.4 고속 위치비교출력(High CMP)

머신 비젼(Machine Vision) 시스템에서 탁월한 성능을 발휘 할 수 있는 COMI-LX504a 제품에 대한 기능을 설명하고 있습니다. COMI-LX504a 제품의 고속 위치비교 출력(高速位置出力) 기능은 기본적으로 제공되는 위치 비교출력(CMP) 기능과 별도로 다른 형태의 함수를 사용해야만 기능을 사용할 수 있습니다.

기본 연속 위치비교출력 기능(cmmCmpTrgContStart() 등의 함수를 사용하는 경우)을 사용할 때는 연속적인 위치 비교 출력의 시간 간격이 500 us 미만(주파수로 보면 2KH 이상)이 되면 정상적으로 동작하지 않을 수 있습니다. 그 이유는 위치 비교기에 비교 데이터를 업데이트하는 작업을 PC의 인터럽트 서비스 루틴에서 수행하기 때문입니다. 이러한 단점을 보완하기 위하여 COMI-LX504a 제품에서는 보드 내에 있는 FIFO 메모리와 FPGA 회로를 통하여 위치 비교 데이터를 업데이트하는 고속 위치비교출력 기능을 탑재하였습니다. 고속 위치비교출력 기능을 사용하면 최대 40 KHz의 트리거 신호를 발생할 수 있습니다.

하나의 COMI-LX504a 장치는 두 개의 고속 위치비교 출력 회로를 제공합니다. 사용자는 이 두개의 회로를 필요한 축에 임의로 할당하여 고속위치비교출력 기능을 사용할 수 있습니다. 다시 말하면 COMI-LX504a의 어떠한 축도 고속위치비교출력 기능을 사용할 수 있습니다. 하지만 동일 장치 내에서 3 축 이상의 축이 동시에 고속위치비교출력 기능을 사용할 수 없습니다.

고속위치비교출력의 경우 단독 출력은 최대 40Khz 을 지원하며, 2 축을 동시에 출력할 때에는 각 20Khz 속도로 출력하게 됩니다.

12.4.1 함수 요약

Summary of Functions
❒ VT_I4 cmmCmpTrgHigh_WriteData ([in] VT_I4 Axis, [in] VT_I4 CMPH_No, [in] VT_R8 IniPos, [in] VT_R8 Interval) 대상(對象) 모션 채널에 대해서, 고속(高速) 위치 비교 출력(位置比較出力) 위치를 등록합니다.
❒ VT_I4 cmmCmpTrgHigh_ReadData ([in] VT_I4 Axis, [out] VT_PI4 CMPH_No, [out] VT_PR8 IniPos, [out] VT_PR8 Interval) 대상(對象) 모션 채널에 대해서, 등록된 고속(高速) 위치 비교 출력(位置比較出力) 위치를 반환(返還)합니다.
❒ VT_I4 cmmCmpTrgHigh_Start ([in] VT_I4 Axis) 대상(對象) 모션 채널에 대해서, 고속(高速) 위치 비교 출력(位置比較出力) 기능을 활성화(活性化) 합니다.
❒ VT_I4 cmmCmpTrgHigh_Stop ([in] VT_I4 Axis) 대상(對象) 모션 채널에 대해서, 고속(高速) 위치 비교 출력(位置比較出力) 기능을 비활성화(非活性化) 합니다.
❒ VT_I4 cmmCmpTrgHigh_Check ([in] VT_I4 Axis, [out] VT_PI4 IsActive, [out] VT_PI4 OutCount) 대상(對象) 모션 채널에 대해서, 고속(高速) 위치 비교 출력(位置比較出力)의 활성화(活性化) 여부와 출력(出力)된 트리거(Trigger)형태의 펄스의 수를 반환(返還)합니다.

12.4.2 함수 설명

NAME

cmmCmpTrgHigh_WriteData
고속 위치 비교 출력 위치 등록
(高速位置比較出力位置登錄)

INFORMATION

- Compare Method
- VC++/VB
- BCB/Delphi/.NET
- Level 8
- LX504a 전용 기능

SYNOPSIS

❒ VT_I4 cmmCmpTrgHigh_WriteData
([in] VT_I4 Axis, [in] VT_I4 CMPH_No, [in] VT_R8 IniPos, [in] VT_R8 Interval)

DESCRIPTION

이 함수는 특정 축에 고속위치비교출력 회로 모듈을 할당하고, 위치비교 트리거(Trigger) 신호를 출력할 위치를 등록하는 함수입니다. 고속위치비교출력 기능은 연속적인 등간격 위치비교 출력을 지원합니다. IniPos 매개 변수(媒介變數)를 통하여 설정된 초기 위치로부터 시작하여 Interval 매개 변수(媒介變數)에 의해서 지정된 간격의 위치마다 트리거 신호를 출력합니다.

고속위치비교출력 기능을 사용하기 위해서는 cmmCmpTrgHigh_Start()를 사용하십시오.

PARAMETER

▶ Axis: 축번호. 축번호는 상수값으로 [cmX1] 부터 0 번째 축을 기준 축으로 임의의 축을 설정할 수 있습니다.

▶ CMPH_No: 사용할 고속위치비교출력 회로 모듈 번호를 설정합니다. 하나의 COMI-LX504a 장치에는 2 개의 고속위치비교출력 회로 모듈이 제공됩니다. 사용자는 이 2 개의 모듈을 원하는 축에 할당하여 사용할 수 있습니다. 여기에서 설정하는 값은 장치(Board) 내에서의 모듈 번호를 설정합니다. 따라서 이 값은 반드시 0 또는 1 이어야 합니다.

▶ IniPos: 트리거 출력을 발생하는 맨 처음 위치를 설정합니다. 고속위치비교출력 기능이 시작된 이후에 모터의 위치가 이 값이 가리키는 위치가 되면 트리거 신호가 출력되기 시작하고 Interval 에서 지정하는 간격의 위치를 지날 때마다 자동으로 트리거 신호가 출력되게 됩니다.

▶ Interval: 트리거 출력을 발생하는 위치 간격을 설정합니다.

SEE ALSO

cmmCmpTrgHigh_ReadData, cmmCmpTrgSetConfig, cmmCfgSetMioProperty (#, cmCMP_PWIDTH, #)

RETURN VALUE

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

REFERENCE

□ 고속위치비교출력 기능도 CMP 출력을 사용하므로 카운터소스, 위치비교 방식 등의 설정을 하는 cmmCmpTrgSetConfig() 함수는 일반 위치비교 출력 기능과 마찬가지로 적용됩니다.

□ 출력 펄스의 폭은 cmmCfgSetMioProperty () 함수를 사용하여 설정합니다. 이때 PropID 매개 변수(媒介變數)에 cmCMP_PWIDTH 를 지정하고 펄스폭을 설정합니다. cmmCfgSetMioProperty (축번호, cmCMP_PWIDTH, 펄스폭매개변수(媒介變數)) 의 형태로 설정합니다. 펄스폭의 기본 설정은 Command 출력의 한 주기에 해당하는 펄스가 출력되는 것입니다.

	이 함수는 COMI-LX504a 제품에 대해서만 사용 가능한 함수입니다.

EXAMPLE

□ 아래의 예제에서는 X 축을 0 에서 50000 좌표로 이동시키면서 피드백 위치가 1000 부터 시작하여 100 의 간격 마다 트리거 신호를 출력하도록 하는 예제입니다. 다시 말해서 피드백 위치가 1000, 1100, 1200, 1300, … 이 될 때마다 트리거 펄스가 출력되도록 하는 예입니다.

C/C++

```
#define AXIS 12 // 축번호는 통합 축 번호 사용함을 예시하기 위해서 12 번 축으로 하였으며 다른 뜻은 없음
#define CMPH_No 0 // CMPH_No 는 통합축 번호와 관계없이 해당 축이 속한 장치내에서 제공하는 2 개의 모듈 중에
                  // 하나를 지정해야 한다. 따라서 이 값은 항상 0 또는 1 중의 하나의 값이어야 한다.

    cmmSxMoveTo(AXIS, 0, cmFALSE);

    // 비교기 설정: 이 것은 다른 일반 위치비교출력 기능과 같습니다 //
    cmmCmpTrgSetConfig (AXIS, cmCNT_FEED, cmEQ_BIDIR);

    // AXIS 축에 고속위치비교출력 기능을 할당하고 초기 위치와 간격을 설정한다. //
    cmmCmpTrgHigh_WriteData (AXIS, CMPH_No, 1000, 100);

    // 고속위치비교출력 기능 시작 //
    cmmCmpTrgHigh_Start(AXIS);

    // X 축을 50000 포인트로 이동한다. 이동할 때 피드백 위치가 1000, 1100, 1200,… 일때 트리거 신호 출력됨 //
    cmmSxMoveTo(AXIS, 50000, cmFALSE);
    cmmCmpTrgHigh_Stop(AXIS) ; // 고속위치비교 기능 종료
```

Visual Basic

```
Const AXIS = 12 ‘축번호는 통합 축 번호 사용함을 예시하기 위해서 12 번 축으로 하였으며 다른 뜻은 없음
Const CMPH_No = 0 ‘ CMPH_No 는 통합축 번호와 관계없이 해당 축이 속한 장치내에서 제공하는 2 개의 모듈 중에
                  ‘ 하나를 지정해야 한다. 따라서 이 값은 항상 0 또는 1 중의 하나의 값이어야 한다.

    Call cmmSxMoveTo(AXIS, 0, cmFALSE)

    ‘ 비교기 설정: 이 것은 다른 일반 위치비교출력 기능과 같습니다.
    Call cmmCmpTrgSetConfig (AXIS, cmCNT_FEED, cmEQ_BIDIR)

    ‘ AXIS 축에 고속위치비교출력 기능을 할당하고 초기 위치와 간격을 설정한다.
    Call cmmCmpTrgHigh_WriteData (AXIS, CMPH_No, 1000, 100)

    ‘ 고속위치비교출력 기능 시작
    Call cmmCmpTrgHigh_Start(AXIS)

    ‘ X 축을 50000 포인트로 이동한다. 이동할 때 피드백 위치가 1000, 1100, 1200,… 일때 트리거 신호 출력됨
    Call cmmSxMoveTo(AXIS, 50000, cmFALSE)
    Call cmmCmpTrgHigh_Stop(AXIS) ‘ 고속위치비교 기능 종료
```

Delphi

```
Const AXIS = 12;  // 축번호는 통합 축 번호 사용함을 예시하기 위해서 12 번 축으로 하였으며 다른 뜻은 없음
Const CMPH_No = 0;// CMPH_No 는 통합축 번호와 관계없이 해당 축이 속한 장치내에서 제공하는 2 개의 모듈 중에
                  // 하나를 지정해야 한다. 따라서 이 값은 항상 0 또는 1 중의 하나의 값이어야 한다.
```

```
cmmSxMoveTo(AXIS, 0, cmFALSE);

// 비교기 설정: 이 것은 다른 일반 위치비교출력 기능과 같습니다 //
cmmCmpTrgSetConfig(AXIS, cmCNT_FEED, cmEQ_BIDIR);

// AXIS 축에 고속위치비교출력 기능을 할당하고 초기 위치와 간격을 설정한다. //
cmmCmpTrgHigh_WriteData(AXIS, CMPH_No, 1000, 100);

// 고속위치비교출력 기능 시작 //
cmmCmpTrgHigh_Start(AXIS);

// X 축을 50000 포인트로 이동한다. 이동할 때 피드백 위치가 1000, 1100, 1200,… 일때 트리거 신호 출력됨 //
cmmSxMoveTo(AXIS, 50000, cmFALSE);
cmmCmpTrgHigh_Stop(AXIS) ; // 고속위치비교 기능 종료
```

NAME

cmmCmpTrgHigh_ReadData
고속 위치 비교 출력 위치 반환
(高速位置比較出力位置返還)

INFORMATION

- Compare Method
- VC++/VB
- BCB/Delphi/.NET
- Level 8
- LX504a 전용 기능

SYNOPSIS

❒ VT_I4 cmmCmpTrgHigh_ReadData
([in] VT_I4 Axis, [out] VT_PI4 CMPH_No, [out] VT_PR8 IniPos, [out] VT_PR8 Interval)

DESCRIPTION

이 함수는 지정된 축에 고속위치비교출력 회로 모듈이 할당되었는지, 할당되었으면 비교 위치 데이터는 어떻게 설정되었는지를 반환하는 함수입니다.

PARAMETER

▶ Axis: 축번호. 축번호는 상수값으로 [cmX1] 부터 0 번째 축을 기준 축으로 임의의 축을 설정할 수 있습니다.

▶ CMPH_No: 할당된 고속위치비교출력 회로 모듈의 번호를 반환받을 버퍼를 지정합니다. 함수는 이 버퍼에 다음의 표와 같이 해당 축에 고속위치비교출력 회로 모듈이 할당되었는지를 반환합니다.

Value	Meaning
-1	해당 축에 고속위치비교출력 회로가 할당되지 않았습니다.
0	해당 축에 0 번 고속위치비교출력 회로가 할당되었습니다.
1	해당 축에 1 번 고속위치비교출력 회로가 할당되었습니다.

▶ IniPos: 트리거 출력을 발생하는 맨 처음 위치에 대하여 현재 장치에 설정되어 있는 값을 반환 받을 버퍼를 지정합니다.

▶ Interval: 트리거 출력을 발생하는 간격에 대하여 현재 장치에 설정되어 있는 값을 반환 받을 버퍼를 지정합니다.

SEE ALSO

cmmCmpTrgHigh_WriteData

RETURN VALUE

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

이 함수는 COMI-LX504a 제품에 대해서만 사용 가능한 함수입니다.

NAME

cmmCmpTrgHigh_Start
고속 위치 비교 출력 기능 시작
(高速位置比較出力機能始作)

INFORMATION

- Compare Method
- VC++/VB
- BCB/Delphi/.NET
- Level 8
- LX504a 전용 기능

SYNOPSIS

❒ VT_I4 cmmCmpTrgHigh_Start ([in] VT_I4 Axis)

DESCRIPTION

특정 축의 고속위치비교출력 기능을 시작합니다. 이 함수가 호출된 이후에는 cmmCmpTrgHigh_WriteData() 함수를 통하여 해당 축에 고속위치비교출력 회로모듈을 할당하고, 초기 위치와 트리거 간격을 적절하게 설정하였다면, 모션의 위치가 이송될 때 초기 위치부터 트리거 간격의 위치를 지날 때마다 트리거 신호가 출력됩니다.

PARAMETER

▶ Axis: 축번호. 축번호는 상수값으로 [cmX1] 부터 0 번째 축을 기준 축으로 임의의 축을 설정할 수 있습니다.

SEE ALSO

cmmCmpTrgHigh_Stop, cmmCmpTrgHigh_WriteData, cmmCmpTrgSetConfig, cmmCfgSetMioProperty

RETURN VALUE

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

REFERENCE

☐ 고속위치비교출력 기능도 CMP 출력을 사용하므로 카운터소스, 위치비교 방식 등의 설정을 하는 cmmCmpTrgSetConfig() 함수는 일반 위치비교 출력 기능과 마찬가지로 적용됩니다.

☐ 출력 펄스의 폭은 cmmCfgSetMioProperty () 함수를 사용하여 설정합니다. 이때 PropID 매개 변수(媒介變數)에 cmCMP_PWIDTH 를 지정하고 펄스폭을 설정합니다. cmmCfgSetMioProperty (축번호, cmCMP_PWIDTH, 펄스폭매개변수(媒介變數)) 의 형태로 설정합니다. 펄스폭의 기본 설정은 Command 출력의 한 주기에 해당하는 펄스가 출력되는 것입니다.

이 함수는 COMI-LX504a 제품에 대해서만 사용 가능한 함수입니다.

EXAMPLE

□ 아래의 예제에서는 X 축을 0 에서 50000 좌표로 이동시키면서 피드백 위치가 1000 부터 시작하여 100 의 간격 마다 트리거 신호를 출력하도록 하는 예제입니다. 다시 말해서 피드백 위치가 1000, 1100, 1200, 1300, … 이 될 때마다 트리거 펄스가 출력되도록 하는 예입니다.

C/C++

```
#define AXIS 12 // 축번호는 통합 축 번호 사용함을 예시하기 위해서 12 번 축으로 하였으며 다른 뜻은 없음
#define CMPH_No 0 // CMPH_No 는 통합축 번호와 관계없이 해당 축이 속한 장치내에서 제공하는 2 개의 모듈 중에
                  // 하나를 지정해야 한다. 따라서 이 값은 항상 0 또는 1 중의 하나의 값이어야 한다.

    cmmSxMoveTo(AXIS, 0, cmFALSE);

    // 비교기 설정: 이 것은 다른 일반 위치비교출력 기능과 같습니다 //
    cmmCmpTrgSetConfig (AXIS, cmCNT_FEED, cmEQ_BIDIR)

    // AXIS 축에 고속위치비교출력 기능을 할당하고 초기 위치와 간격을 설정한다. //
    cmmCmpTrgHigh_WriteData (AXIS, CMPH_No, 1000, 100);

    // 고속위치비교출력 기능 시작 //
    cmmCmpTrgHigh_Start(AXIS);

    // X 축을 50000 포인트로 이동한다. 이동할 때 피드백 위치가 1000, 1100, 1200,… 일때 트리거 신호 출력됨 //
    cmmSxMoveTo(AXIS, 50000, cmFALSE);
    cmmCmpTrgHigh_Stop(AXIS) ; // 고속위치비교 기능 종료
```

Visual Basic

```
Const AXIS = 12 '축번호는 통합 축 번호 사용함을 예시하기 위해서 12 번 축으로 하였으며 다른 뜻은 없음
Const CMPH_No = 0 ' CMPH_No 는 통합축 번호와 관계없이 해당 축이 속한 장치내에서 제공하는 2 개의 모듈 중에
                  ' 하나를 지정해야 한다. 따라서 이 값은 항상 0 또는 1 중의 하나의 값이어야 한다.

    Call cmmSxMoveTo(AXIS, 0, cmFALSE)

    ' 비교기 설정: 이 것은 다른 일반 위치비교출력 기능과 같습니다.
    Call cmmCmpTrgSetConfig (AXIS, cmCNT_FEED, cmEQ_BIDIR)

    ' AXIS 축에 고속위치비교출력 기능을 할당하고 초기 위치와 간격을 설정한다.
    Call cmmCmpTrgHigh_WriteData (AXIS, CMPH_No, 1000, 100)

    ' 고속위치비교출력 기능 시작
    Call cmmCmpTrgHigh_Start(AXIS)

    ' X 축을 50000 포인트로 이동한다. 이동할 때 피드백 위치가 1000, 1100, 1200,… 일때 트리거 신호 출력됨
    Call cmmSxMoveTo(AXIS, 50000, cmFALSE)
    Call cmmCmpTrgHigh_Stop(AXIS) ' 고속위치비교 기능 종료
```

Delphi

```
Const AXIS = 12;  // 축번호는 통합 축 번호 사용함을 예시하기 위해서 12 번 축으로 하였으며 다른 뜻은 없음
Const CMPH_No = 0;// CMPH_No 는 통합축 번호와 관계없이 해당 축이 속한 장치내에서 제공하는 2 개의 모듈 중에
                  // 하나를 지정해야 한다. 따라서 이 값은 항상 0 또는 1 중의 하나의 값이어야 한다.

    cmmSxMoveTo(AXIS, 0, cmFALSE);

    // 비교기 설정: 이 것은 다른 일반 위치비교출력 기능과 같습니다 //
    cmmCmpTrgSetConfig(AXIS, cmCNT_FEED, cmEQ_BIDIR);

    // AXIS 축에 고속위치비교출력 기능을 할당하고 초기 위치와 간격을 설정한다. //
    cmmCmpTrgHigh_WriteData(AXIS, CMPH_No, 1000, 100);

    // 고속위치비교출력 기능 시작 //
    cmmCmpTrgHigh_Start(AXIS);
```

```
// X 축을 50000 포인트로 이동한다. 이동할 때 피드백 위치가 1000, 1100, 1200,… 일때 트리거 신호 출력됨 //
cmmSxMoveTo(AXIS, 50000, cmFALSE);
cmmCmpTrgHigh_Stop(AXIS) ; // 고속위치비교 기능 종료
```

NAME

cmmCmpTrgHigh_Stop
고속 위치 비교 출력 기능 정지
(高速位置比較出力機能停止)

INFORMATION

- Compare Method
- VC++/VB
- BCB/Delphi/.NET
- Level 8
- LX504a 전용 기능

SYNOPSIS

❒ VT_I4 cmmCmpTrgHigh_Stop ([in] VT_I4 Axis)

DESCRIPTION

고속위치비교출력 기능을 정지합니다.

PARAMETER

▶ Axis: 축번호. 축번호는 상수값으로 [cmX1] 부터 0 번째 축을 기준 축으로 임의의 축을 설정할 수 있습니다.

SEE ALSO

cmmCmpTrgHigh_Start

RETURN VALUE

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

REFERENCE

이 함수는 COMI-LX504a 제품에 대해서만 사용 가능한 함수입니다.

NAME

cmmCmpTrgHigh_Check
– 고속 위치 비교 출력
(高速位置比較出力)펄스(Pulse) 수(數)
반환(返還)

INFORMATION

- Compare Method
- VC++/VB
- BCB/Delphi/.NET
- Level 8
- LX504a 전용 기능

SYNOPSIS

❒ VT_I4 cmmCmpTrgHigh_Check
([in] VT_I4 Axis, [out] VT_PI4 IsActive, [out] VT_PI4 OutCount)

DESCRIPTION

특정 축에 대하여 고속위치비교출력 기능이 활성화되었는지, 활성화되었으며 현재 트리거 펄스는 몇 개가 출력되었는지를 반환해 주는 함수입니다.

PARAMETER

▶ Axis: 축번호. 축번호는 상수값으로 [cmX1] 부터 0 번째 축을 기준 축으로 임의의 축을 설정할 수 있습니다.

▶ IsActive: 지정한 축에 대하여 고속위치비교출력 기능이 활성화 되었는지에 대한 결과를 반환받을 버퍼를 지정합니다. 이 값을 NULL 로 전달하면 활성화 여부에 대한 결과를 반환하지 않습니다.

▶ OutCount: 트리거펄스가 출력된 횟수를 반환받을 버퍼를 지정합니다. 이 값을 NULL 로 전달하면 결과를 반환하지 않습니다.

SEE ALSO

cmmCmpTrgHigh_Start

RETURN VALUE

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

REFERENCE

이 함수는 COMI-LX504a 제품에 대해서만 사용 가능한 함수입니다.

Chapter

13

Universal Digital I/O Control

외부 디지털 입력과 출력을 위해 마련된범용 디지털 입출력 기능에 대해서 CMMSDK 는 매우 유연한 입출력 기능을 제공하고 있습니다. 특히 S/W 노이즈 필터 기능은 보다 안정적인 디지털 입력 신호 처리를 이유로 사용될 수 있으며, 결과적으로 모션 시스템의 전체적인 신뢰성을 동시에 향상시키는 효과를 경험하실 수 있습니다.

모션 시스템에서 범용 (汎用) 디지털 입출력은 주요 외부 입출력 인터페이스(Interface)의 하나로서, 소위 하나로서, 소위 디지털 접점(Digital Contact) 이라는 말로 일컫습니다. CMMSDK 는 모션 제품의 자체 Digital 제품의 자체 Digital I/O 를 포함하여, 넓게는 자사의 SD4xx Series 제품까지도 디지털 입출력 인터페이스를 인터페이스를 통합하였습니다. 이 장에서는 편리한 CMMSDK 의 디지털 입출력 기능들을 자세히 살펴보고, 자세히 살펴보고, 활용할 수 있도록 구성하였습니다.

13 범용 디지털 입출력 편

㈜커미조아 모션컨트롤러는 모션제어전용 I/O 뿐 아니라 범용적으로 사용할 수 있는 디지털입출력 채널을 제공합니다. 지원되는 디지털 입출력 채널 수는 제품모델마다 다르므로 하드웨어 매뉴얼을 참조하시기 바랍니다. 또한, COMIZOA 의 DAQ 제품군의 중 모델명이 SD4xx 제품군에서는 디지털 입출력 채널의 경우 본 CMMSDK 의 통합 범용 디지털 입출력 채널에 일괄적으로 통합되어 포함됩니다.

이때 주의 하실 점은 다음과 같습니다. 범용 디지털 입출력 채널에 대한 명시적인 채널 번호의 관계이므로, 이 점을 잘 숙지해 주시기를 부탁드립니다.

모션 보드에서 제공되는 기본적인 2 축에 각 3 채널(Digital Input 3 채널, Digital Output 3 채널) 의 범용 디지털 입출력 접점의 경우 Windows 2000 / XP 의 운영체제 상에서 CPU 를 기준으로 한 PCI 슬롯 위치에서 상대적으로 CPU 를 기준으로 채널 번호가 증가됩니다.

모션 보드가 아닌 DAQ 제품군 중 SD4xx 제품과 같은 디지털 입출력 제품이 혼용되어 사용될 경우에는 모션 보드의 채널 순서보다 항상 DAQ 제품군의 디지털 입출력 채널 번호가 앞에 존재합니다. 이 경우에는 PCI 슬롯의 상대적 위치와 관계가 없이 적용됩니다.

디지털 입력 채널과 디지털 출력 채널의 번호는 서로 상관관계를 가지지 않습니다. 독립 적인 채널 번호로 순서화 됩니다.

위에서 서술한 모든 경우에서 사용자는 디지털 입출력 채널의 번호를 고객(顧客) 여러분들께서 직접 정의하실 수 있습니다. 이 기능은 API 함수로 제공되지 않으며, 별도로 제공되는 CME2 (COMIZOA Motion Environment 2) 파일을 통해 적용할 수 있습니다. CME 파일에 대한 내용은 부록 편의 CME Builder 를 이용한 환경설정 을 통해 안내 받으시기 바랍니다..

13.1 함수 요약

범용 디지털입출력 기능과 관련된 함수 는 다음의 표와 같습니다.

가) 범용 디지털 입력

Summary of Functions
❒ VT_I4 cmmDiSetInputLogic ([in] VT_I4 Channel, [in] VT_I4 InputLogic) 대상(對象) 디지털 입력(Digital Input) 채널의 입력 논리(Input Logic)를 설정(設定)합니다.
❒ VT_I4 cmmDiGetInputLogic ([in] VT_I4 Channel, [out] VT_PI4 InputLogic) 대상(對象) 디지털 입력(Digital Input) 채널의 입력 논리(Input Logic)의 설정 상태를 반환(返還)합니다.
❒ VT_I4 cmmDiGetOne ([in] VT_I4 Channel, [out] VT_PI4 InputState) 대상(對象) 디지털 입력(Digital Input) 채널의 단일(單一) 디지털입력(入力) 채널의 상태를 반환(返還)합니다.
❒ VT_I4 cmmDiGetMulti ([in] VT_I4 IniChannel, [in] VT_I4 NumChannels, [out] VT_PI4 InputStates) 대상(對象) 디지털 입력(Digital Input) 채널 범위의 다중(多重) 디지털입력(入力) 채널의 상태를 반환(返還)합니다.

나) 범용 디지털 출력

Summary of Functions

❒ VT_I4 cmmDoSetOutputLogic ([in] VT_I4 Channel, [in] VT_I4 OutputLogic) 대상(對象) 디지털 출력(Digital Output) 채널의 출력 논리(Output Logic)를 설정(設定)합니다.
❒ VT_I4 cmmDoGetOutputLogic ([in] VT_I4 Channel, [out] VT_PI4 OutputLogic) 대상(對象) 디지털 출력(Digital Output) 채널의 출력 논리(Output Logic)의 설정(設定) 상태를 반환(返還)합니다.
❒ VT_I4 cmmDoPutOne ([in] VT_I4 Channel, [in] VT_I4 OutState) 대상(對象) 디지털 출력(Digital Output) 채널의 단일(單一) 디지털 채널을 통해 디지털 출력(Digital Output)을 발생시킵니다.
❒ VT_I4 cmmDoGetOne ([in] VT_I4 Channel, [out] VT_PI4 OutState) 대상(對象) 디지털 출력(Digital Output) 채널의 단일(單一) 디지털 채널을 통해 디지털 출력(Digital Output) 상태를 반환합니다.
❒ VT_I4 cmmDoPutMulti ([in] VT_I4 IniChannel, [in] VT_I4 NumChannels, [in] VT_I4 OutputStates) 대상(對象) 디지털 출력(Digital Output) 채널 범위의 다중(多重) 디지털 출력 채널을 통해 동시에 디지털 출력(Digital Output)을 발생시킵니다.
❒ VT_I4 cmmDoGetMulti ([in] VT_I4 IniChannel, [in] VT_I4 NumChannels, [out] VT_PI4 OutputStates) 대상(對象) 디지털 출력(Digital Output) 채널 범위의 다중(多重) 디지털 채널을 통해 동시에 디지털 출력(Digital Output) 상태를 확인(確認)합니다.
❒ VT_I4 cmmDoPulseOne ([in] VT_I4 Channel, [in] VT_I4 IsOnPulse, [in] VT_I4 dwDuration, [in] VT_I4 IsWaitPulseEnd) 대상(對象) 디지털 출력(Digital Output) 채널의 단일(單一) 디지털 채널을 통해 단일 펄스 출력(出力)을 발생시킵니다.
❒ VT_I4 cmmDoPulseMulti ([in] VT_I4 IniChannel, [in] VT_I4 NumChannels, [in] VT_I4 OutStates, [in] VT_I4 dwDuration, [in] VT_I4 IsWaitPulseEnd) 대상(對象) 디지털 출력(Digital Output) 채널 범위의 지정한 다중(多重) 디지털 채널을 통해 단일 펄스 출력(出力)을 발생시킵니다.

다) 기타 함수

Summary of Functions
❒ VT_I4 cmmDiGetOneF ([in] VT_I4 Channel, [in] VT_I4 CutoffTime_us, [out] VT_PI4 InputState) 대상(對象) 디지털 입력(Digital Input) 채널의 단일(單一) 채널을 대상으로 노이즈 필터(Noise Filter) 기능을 활성화 하여, 디지털 입력(入力) 상태를 반환(返還)합니다.
❒ VT_I4 cmmDiGetMultiF ([in] VT_I4 IniChannel, [in] VT_I4 NumChannels, [in] VT_I4 CutoffTime_us, [out] VT_PI4 InputStates) 대상(對象) 디지털 입력(Digital Input) 채널 범위의 다중(多重) 채널을 대상으로 노이즈 필터(Noise Filter) 기능을 활성화 하여, 디지털 입력(入力) 상태를 반환(返還)합니다.

13.2 함수 설명

13.2.1 범용 디지털 입력

NAME

cmmDiSetInputLogic
cmmDiGetInputLogic
– 대상 디지털 입력(入力) 채널의
논리 설정(論理設定) 및 확인(確認)

INFORMATION

- Universal DIO Control
- VC++/VB
- BCB/Delphi/.NET
- Level 1
- 위험 요소 없음

SYNOPSIS

- ❒ VT_I4 cmmDiSetInputLogic ([in] VT_I4 Channel, [in] VT_I4 InputLogic)
- ❒ VT_I4 cmmDiGetInputLogic ([in] VT_I4 Channel, [out] VT_PI4 InputLogic)

DESCRIPTION

cmmDiSetInputLogic() 함수는 지정한 대상 채널의 Digital 입력 채널의 입력 논리(Input Logic)을 설정합니다.
cmmDiGetInputLogic() 함수는 지정한 대상 채널의 Digital 입력 채널의 입력 논리(Input Logic)의 설정 값을 반환합니다.

PARAMETER

▶ Channel : 채널번호는 상수 값으로 0 번째 채널을 기준채널로 임의의 채널을 설정 할 수 있습니다.

▶ InputLogic : cmmDiSetInputLogic 함수의 인자이며, 디지털 입력 채널의 입력 논리(Input Logic) 을 설정합니다.

Value	Meaning
cmLOGIC_A	A 접점 방식 => 평상시 Open, 감지되면 Close 되는 스위치 방식
cmLOGIC_B	B 접점 방식 => 평상시 Close, 감지되면 Open 되는 스위치 방식

▶ InputLogic : cmmDiGetInputLogic 함수의 인자이며, 디지털 입력 채널의 입력 논리(Input Logic) 을 반환합니다.

Value	Meaning
cmLOGIC_A	A 접점 방식 => 평상시 Open, 감지되면 Close 되는 스위치 방식
cmLOGIC_B	B 접점 방식 => 평상시 Close, 감지되면 Open 되는 스위치 방식

RETURN VALUE

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

NAME

cmmDiGetOne
– 단일(單一) 디지털 입력(入力) 채널 상태(狀態) 확인(確認)

INFORMATION

- Universal DIO Control
- VC++/VB
- BCB/Delphi/.NET
- Level 1
- 위험 요소 없음

SYNOPSIS

❒ VT_I4 cmmDiGetOne ([in] VT_I4 Channel, [out] VT_PI4 InputState)

DESCRIPTION

단일 채널에 대한 디지털 입력 상태를 확인(確認)합니다. 이 상태는 cmmDiSetInputLogic() 함수를 통해 설정된 디지털 입력 논리(Digital Input Logic)가 적용됩니다.

PARAMETER

▶ Channel : Digital Input 채널번호. 채널번호는 상수 값으로 0 번째 채널을 기준채널로 임의의 채널을 설정 할 수 있습니다.

▶ InputState : 해당 채널의 디지털 입력 상태를 확인(確認)합니다.

Value	Meaning
0	OFF
1	ON

RETURN VALUE

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

NAME

cmmDiGetMulti
– 다중(多重) 디지털 입력(入力) 채널 상태(狀態) 확인(確認)

INFORMATION

Universal DIO Control
VC++/VB
BCB/Delphi/.NET
Level 1
위험 요소 없음

SYNOPSIS

❒ VT_I4 cmmDiGetMulti
([in] VT_I4 IniChannel, [in] VT_I4 NumChannels, [out] VT_PI4 InputStates)

DESCRIPTION

다중 채널에 대한 디지털 입력 상태를 확인(確認)합니다. 이 상태는 cmmDiSetInputLogic() 함수를 통해 설정된 디지털 입력 논리(Digital Input Logic)가 적용됩니다. 시작 채널부터 설정된 범위의 채널에 대한 Digital 입력 상태값을 얻습니다.

PARAMETER

▶ *IniChannel* : 시작 채널번호. 채널번호는 상수 값으로 0 번째 채널을 기준채널로 임의의 채널을 설정 할 수 있습니다.

▶ *NumChannels* : 시작 채널로부터 몇 개의 채널의 상태를 확인(確認)할 것인지에 대한 값을 전달합니다.

▶ *InputState* : 다중 디지털 입력(Digital Input) 채널의 상태

Value	Meaning
0	OFF
1	ON

RETURN VALUE

사용자가 의도한 지정한 다중 입력 채널에 대해 실제 확인하게 된 하드웨어 채널 갯수를 반환합니다.

13.2.2 범용 디지털 출력

NAME

cmmDoSetOutputLogic
cmmDoGetOutputLogic
– 대상(對象) 디지털 출력(出力) 채널 논리 설정(論理設定) 및 확인(確認)

INFORMATION

- Universal DIO Control
- VC++/VB
- BCB/Delphi/.NET
- Level 1
- 위험 요소 없음

SYNOPSIS

- ❒ VT_I4 cmmDoSetOutputLogic ([in] VT_I4 Channel, [in] VT_I4 OutputLogic)
- ❒ VT_I4 cmmDoGetOutputLogic ([in] VT_I4 Channel, [out] VT_PI4 OutputLogic)

DESCRIPTION

cmmDoSetOutputLogic() 함수는 지정한 채널의 Digital 출력 채널의 출력 논리(Output Logic)를 설정합니다.
cmmDoGetOutputLogic() 함수는 지정한 채널의 Digital 출력 채널의 출력 논리(Output Logic)의 설정 값을 반환합니다.

PARAMETER

▶ ***Channel*** : 채널번호는 상수 값으로 0 번째 채널을 기준채널로 임의의 채널을 설정 할 수 있습니다.

▶ ***OutputLogic*** : cmmDoSetOutputLogic 함수의 인자이며, 디지털 출력 채널의 출력 논리(Output Logic) 를 설정합니다.

Value	Meaning
cmLOGIC_A	A 접점 방식 => 평상시 Open, 감지되면 Close 되는 스위치 방식
cmLOGIC_B	B 접점 방식 => 평상시 Close, 감지되면 Open 되는 스위치 방식

▶ ***OutputLogic*** : cmmDoGetOutputLogic 함수의 인자이며, 디지털 출력 채널의 출력 논리(Output Logic) 를 반환합니다.

Value	Meaning
cmLOGIC_A	A 접점 방식 => 평상시 Open, 감지되면 Close 되는 스위치 방식
cmLOGIC_B	B 접점 방식 => 평상시 Close, 감지되면 Open 되는 스위치 방식

RETURN VALUE

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

NAME

cmmDoPutOne
cmmDoGetOne
– 단일(單一) 디지털 출력(出力) 채널
출력(出力) 발생 및 상태 (狀態) 확인(確認)

INFORMATION

- Universal DIO Control
- VC++/VB
- BCB/Delphi/.NET
- Level 1
- 위험 요소 없음

SYNOPSIS

- ❒ VT_I4 cmmDoPutOne ([in] VT_I4 Channel, [in] VT_I4 OutState)
- ❒ VT_I4 cmmDoGetOne ([in] VT_I4 Channel, [out] VT_PI4 OutState)

DESCRIPTION

cmmDoPutOne() 함수는 지정한 디지털출력 채널을 통해 디지털 출력을 발생합니다.
cmmDoGetOne() 함수는 지정한 디지털출력 채널의 현재 출력 상태를 반환합니다.

PARAMETER

▶ ***Channel*** : 디지털 출력의 대상의 채널번호. 채널번호는 상수 값으로 0 번째 채널을 기준채널로 임의의 채널을 설정할 수 있습니다.

▶ ***OutState*** : cmmDoPutOne 함수의 인자이며, 단일 채널에 대한 디지털 출력 상태를 설정합니다.

Value	Meaning
0	OFF
1	ON

▶ ***OutState*** : cmmDoGetOne 함수의 인자이며, 단일 채널에 대한 디지털 출력 상태를 반환합니다.

Value	Meaning
0	OFF
1	ON

RETURN VALUE

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

NAME

cmmDoPutMulti
cmmDoGetMulti
– 다중(多重) 디지털 출력(出力) 채널 출력(出力) 발생(發生) 및 상태 (狀態) 확인(確認)

INFORMATION

- Universal DIO Control
- VC++/VB
- BCB/Delphi/.NET
- Level 1
- 위험 요소 없음

SYNOPSIS

❒ VT_I4 cmmDoPutMulti
([in] VT_I4 IniChannel, [in] VT_I4 NumChannels, [in] VT_I4 OutputStates)
❒ VT_I4 cmmDoGetMulti
([in] VT_I4 IniChannel, [in] VT_I4 NumChannels, [out]VT_PI4 OutputStates)

DESCRIPTION

다중 채널에 대한 디지털 출력 혹은 출력 상태를 확인(確認)합니다. 이 상태는 cmmDoSetOutputLogic() 함수를 통해 설정된 디지털 출력 논리(Digital Output Logic)가 적용됩니다. 시작 채널부터 설정된 범위의 채널에 대한 Digital 출력 상태값을 얻습니다.

cmmDoPutMulti() 는 IniChannel 에서 지정한 채널부터 NumChannels 의 수 만큼의 다중 채널에 대해서 디지털 출력을 발생합니다.

cmmDoGetMulti() 는 IniChannel 에서 지정한 채널부터 NumChannels 의 수 만큼의 다중 채널에 대해서 디지털 출력 상태를 반환합니다.

PARAMETER

▶ ***IniChannel*** : 시작 채널번호. 채널번호는 상수 값으로 0 번째 채널을 기준채널로 임의의 채널을 설정 할 수 있습니다.

▶ ***NumChannels*** : 대상 채널 개수.

▶ ***OutputStates*** : cmmDoPutMulti 함수의 인자인 OutputStates 는 다중 채널에 대한 출력 상태를 설정합니다. 이 출력 상태는 BitMask (비트 마스크) 로 설정되며, 설정된 비트가 1 일 경우 디지털 출력(Digital Output)이 발생합니다.

▶ ***OutputStates*** : cmmDoGetMulti 함수의 인자인 OutputStates 는 다중 채널에 대한 디지털 출력 상태를 반환합니다. 이 출력 상태는 BitMask (비트 마스크) 로 설정되며, 설정된 비트가 1 일 경우 디지털 출력(Digital Output)이 발생한 경우입니다.

Value	Meaning
0	OFF
1	ON

RETURN VALUE

cmmDoGetMulti 함수의 리턴값은 다음과 같습니다.
사용자가 의도한 지정한 다중 출력 채널에 대해 실제 확인하게 된 하드웨어 채널 갯수를 반환합니다.

cmmDoPutMulti 함수의 리턴값은 다음과 같습니다.

Value	Meaning

음수	수행 실패. 자세한 내용은 '에러처리' 편을 참고합니다
cmERR_NONE	수행 성공

NAME

cmmDoPulseOne
– 단일 (單一) 디지털 출력(出力) 채널의
단일(單一) 펄스 출력(出力) 발생

INFORMATION

- Universal DIO Control
- VC++/VB
- BCB/Delphi/.NET
- Level 1
- 위험 요소 없음

SYNOPSIS

❒ VT_I4 cmmDoPulseOne

([in] VT_I4 Channel, [in] VT_I4 IsOnPulse, [in] VT_I4 dwDuration, [in] VT_I4 IsWaitPulseEnd)

DESCRIPTION

cmmDoPulseOne() 함수는 지정한 단일 디지털출력 채널을 통해 단일 펄스 출력을 발생시킵니다.

PARAMETER

▶ ***Channel*** : 펄스 출력의 대상의 채널번호. 채널번호는 상수 값으로 0 번째 채널을 기준채널로 임의의 채널을 설정할 수 있습니다.

▶ ***IsOnPulse***: 설정된 디지털 출력 논리에 따라 초기 펄스 출력의 형태를 결정합니다. 만약 A 접점의 디지털 출력 논리 상황에서 이 매개변수의 값이 cmTRUE 일 경우에는 Active Low 상태가 되며, B 접점의 디지털 출력 논리 상황에서는 Active High 상태가 됩니다.

Value	Meaning
cmFALSE	Active High
cmTRUE	Active Low

▶ ***dwDuration*** : 펄스 출력 시간을 설정합니다. 이 출력시간은 Active 된 Pulse 출력의 시간을 의미합니다.

▶ ***IsWaitPulseEnd*** : 펄스 출력 동작시에 함수를 바로 반환할 것인지, 아니면 펄스 출력 시간 동안 함수 반환을 대기할 것인지를 결정합니다. 이 매개변수가 cmTRUE 일 경우에는 함수의 반환은 펄스 출력의 종료시점까지 지연됩니다.

Value	Meaning
cmFALSE	펄스출력 시작 후 바로 함수를 반환합니다.
cmTRUE	함수의 반환시점을 펄스 출력종료 시점까지 대기합니다.

RETURN VALUE

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

NAME

cmmDoPulseMulti
– 다중(多重) 디지털 출력(出力) 채널의 단일(單一) 펄스 출력(出力) 발생

INFORMATION

Universal DIO Control

VC++/VB

BCB/Delphi/.NET

Level 1

위험 요소 없음

SYNOPSIS

❒ VT_I4 cmmDoPulseMulti

([in] VT_I4 IniChannel, [in] VT_I4 NumChannels, [in] VT_I4 OutStates, [in] VT_I4 dwDuration, [in] VT_I4 IsWaitPulseEnd)

DESCRIPTION

cmmDoPulseMulti() 함수는 지정한 다중 디지털출력 채널을 통해 단일 펄스 출력을 발생시킵니다.

PARAMETER

▶ ***IniChannel*** : 펄스 출력의 대상의 시작 채널번호. 시작 채널번호는 상수 값으로 0 번째 채널을 기준채널로 임의의 채널을 설정 할 수 있습니다.

▶ ***NumChannels*** : 대상 채널 개수

▶ ***OutStates***: 설정된 디지털 출력 논리에 따라 초기 펄스 출력의 형태를 결정합니다. 만약 A 접점의 디지털 출력 논리 상황에서 이 매개변수의 값이 cmTRUE 일 경우에는 Active Low 상태가 되며, B 접점의 디지털 출력 논리 상황에서는 Active High 상태가 됩니다. 이 매개변수는 개별 채널에 대한 비트 플래그로 구성되어야 하며, 최대 32 개의 채널에 대한 OutStates 를 비트 플래그로 설정할 수 있습니다.

▶ ***dwDuration*** : 펄스 출력 시간을 설정합니다. 이 출력시간은 Active 된 Pulse 출력의 시간을 의미합니다.

▶ ***IsWaitPulseEnd*** : 펄스 출력시 동작시에 함수를 바로 반환할 것인지, 아니면 펄스 출력 시간 동안 함수 반환을 대기할 것인지를 결정합니다. 이 매개변수가 cmTRUE 일 경우에는 함수의 반환은 펄스 출력의 종료시점까지 지연됩니다.

Value	Meaning
cmFALSE	OFF
cmTRUE	ON

RETURN VALUE

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

13.2.3 기타 함수

NAME

cmmDiGetOneF
– 단일(單一) 디지털 입력(入力) 채널의 노이즈 필터 기능 : 디지털 입력(入力) 상태(狀態) 확인(確認)

INFORMATION

- Universal DIO Control
- VC++/VB
- BCB/Delphi/.NET
- Level 1
- 위험 요소 없음

SYNOPSIS

❒ VT_I4 cmmDiGetOneF ([in] VT_I4 Channel, [in] VT_I4 CutoffTime_us, [out] VT_PI4 InputState)

DESCRIPTION

단일 채널에 대상으로 노이즈 필터 기능에 대응 되는 디지털 입력 상태를 확인(確認)합니다. 이 상태는 cmmDiSetInputLogic() 함수를 통해 설정된 디지털 입력 논리(Digital Input Logic)가 적용됩니다. 이 함수에서 전달되는 매개 변수 중 'CutoffTime_us'는 입력 신호 유지 시간(Signal Width)을 의미합니다. 이 함수를 통해 확인(確認)되는 디지털 입력 채널은 입력 신호 유지 시간 이상의 신호가 확인(確認)되어야만 합니다. 원하지 않는 신호(Noise) 나 일정 시간 이상의 펄스 입력(Pulse Input)을 확인(確認)할 때 매우 유용한 디지털 입력 상태 확인(確認) 함수입니다.

PARAMETER

▶ ***Channel*** : Digital Input 채널번호. 채널번호는 상수 값으로 0 번째 채널을 기준채널로 임의의 채널을 설정 할 수 있습니다.

▶ ***CutoffTime_us*** : 디지털 입력 신호 유지 시간을 마이크로 초(us) 단위로 설정합니다.

▶ ***InputState*** : 해당 채널의 디지털 입력 상태를 확인(確認)합니다.

Value	Meaning
0	OFF
1	ON

RETURN VALUE

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

NAME

cmmDiGetMultiF
– 다중(多重) 디지털 입력(入力) 채널의 노이즈 필터 기능 : 디지털 입력(入力) 상태(狀態) 확인(確認)

INFORMATION

Universal DIO Control
VC++/VB
BCB/Delphi/.NET
Level 1
위험 요소 없음

SYNOPSIS

❒ VT_I4 cmmDiGetMultiF
([in] VT_I4 IniChannel, [in] VT_I4 NumChannels, [in] VT_I4 CutoffTime_us, [out] VT_PI4 InputStates)

DESCRIPTION

다중 채널을 대상으로 노이즈 필터 기능에 대응 되는 디지털 입력 상태를 확인(確認)합니다. 이 상태는 cmmDiSetInputLogic() 함수를 통해 설정된 디지털 입력 논리(Digital Input Logic)가 적용됩니다. 이 함수에서 전달되는 매개 변수 중 'CutoffTime_us'는 입력 신호 유지 시간(Signal Width)을 의미합니다. 이 함수를 통해 확인(確認)되는 디지털 입력 채널들은 제한된 입력 신호 유지 시간 이상의 신호가 확인(確認)되어야만 합니다. 원하지 않는 신호(Noise) 나 일정 시간 이상의 펄스 입력(Pulse Input)을 확인(確認)할 때 매우 유용한 디지털 입력 상태 확인(確認) 함수입니다.

PARAMETER

▶ *IniChannel* : 시작 채널번호. 채널번호는 상수 값으로 0 번째 채널을 기준채널로 임의의 채널을 설정 할 수 있습니다.

▶ *NumChannels* : 시작 채널로부터 몇 개의 채널의 상태를 확인(確認)할 것인지에 대한 값을 전달합니다.

▶ *CutoffTime_us* : 디지털 입력 신호 유지 시간을 마이크로 초(us) 단위로 설정합니다.

▶ *InputStates* : 다중 디지털 입력(Digital Input) 채널의 상태

Value	Meaning
0	OFF
1	ON

RETURN VALUE

사용자가 의도한 지정한 다중 입력 채널에 대해 실제 확인하게 된 하드웨어 채널 갯수를 반환합니다.

Chapter

14

Advanced and Extended Interface

다양한 고급 기능 지원을 위해 CMMSDK 가 제공하는 확장된 인터페이스를 사용하실 수 있습니다. 확장 인터페이스에 대한 기능은 고급 개발자나 커미조아 기술진들을 통해 그 사용안내를 받으 실 수 있습니다. 고급 인터페이스에 관련된 모든 기능의 구성은 보다 유연하고 다양(多樣)한 기능의 모션 제어를 위해 사용되어 집니다.

모션 고급 기능은 커미조아의 다양한 확장 모션 제어에 주로 이용되어 집니다. 정확한 모션 제어와 안정적인 제어와 안정적인 모션제어를 위해 제공되는 고급 모션 기능들은 ㈜ 커미조아 기술 진들이나 관련 기술 지원 관련 기술 지원 협약점을 통해서 문의해주시기 바랍니다.

14 고급/확장 인터페이스 편

본 단원에서 사용되는 함수들은 대부분 일반 사용자들이 거의 사용하지 않을 함수들입니다. 하지만 특수한 경우에 이러한 함수들을 사용할 필요가 있을 수 있으므로 이 함수들에 대한 설명을 수록합니다. 단, 자세한 함수의 설명은 생략합니다. 이 함수들에 대한 보다 자세한 설명이 필요한 사용자께서는 ㈜커미조아의 기술지원팀이나 기타 기술 협력 점을 통해 문의해주시기 바랍니다.

고급기능 함수들의 이름은 모두 중간 첨두어 "Adv"를 포함하며 이에 해당하는 함수들의 리스트는 다음의 표와 같습니다.

VT_I4 cmmAdvGetNumAvailAxes (**[**out] VT_PI4 NumAxes)
지정된 모션컨트롤러에서 지원하는 제어축 수를 반환합니다.

VT_I4 cmmAdvGetNumDefinedAxes ([out] VT_PI4 NumAxes)
㈜ 커미조아 CMMSDK 모션 환경 파일인 'CME2 파일'을 통해 정의된 축 개수를 반환합니다.

VT_I4 cmmAdvGetNumAvailDioChan ([in] VT_I4 IsInputChannel, [out] VT_PI4 NumChannels)
현재 사용가능한 범용 디지털 입출력 채널의 수를 반환합니다.

VT_I4 cmmAdvGetNumDefinedDioChan ([in] VT_I4 IsInputChannel, [out] VT_PI4 NumChannels)
㈜ 커미조아 CMMSDK 모션 환경 파일인 'CME2 파일'을 통해 정의된 범용
디지털 입출력 채널의 수를 반환합니다.

VT_I4 cmmAdvGetMotDeviceId ([in] VT_I4 Axis, [out] VT_PI4 DeviceId)
전달된 축의 모션 장치의 장치 ID 를 반환합니다.

VT_I4 cmmAdvGetMotDevInstance ([in] VT_I4 Axis, [out] VT_PI4 DevInstance)
전달된 축의 모션 장치의 인스턴스(Instance) 를 반환합니다.

VT_I4 cmmAdvGetDioDeviceId ([in] VT_I4 Axis, [in] VT_I4 IsInputChannel, [out] VT_PI4 DeviceId)
전달된 축의 범용 디지털 채널이 속한 장치의 ID 를 반환합니다.

VT_I4 cmmAdvGetDioDevInstance ([in] VT_I4 Axis, [in] VT_I4 IsInputChannel, [out] VT_PI4 DevInstance)
전달된 축의 범용 디지털 채널이 속한 장치의 인스턴스(Instance)를 반환합니다.

VT_I4 cmmAdvGetDeviceHandle ([in] VT_I4 DeviceId, [in] VT_I4 DevInstance, [out] VT_HANDLE DevHandle)
지정한 장치의 ID 와 인스턴스에 해당하는 장치 핸들 값을 반환합니다.

VT_I4 cmmAdvWriteMainSpace ([in] VT_I4 Axis, [in] VT_I4 Addr, [in] VT_I4 Value)
Undocument Function 입니다. 이 함수는 기술 지원이나 고객(顧客) 지원용으로 사용됩니다.

VT_I4 cmmAdvReadMainSpace ([in] VT_I4 Axis, [in] VT_I4 Addr, [out] VT_PI4 Value)
Undocument Function 입니다. 이 함수는 기술 지원이나 고객(顧客) 지원용으로 사용됩니다.

VT_I4 cmmAdvWriteRegister ([in] VT_I4 Axis, [in] VT_I4 RegisterNo, [in] VT_I4 RegVal)
Undocument Function 입니다. 이 함수는 기술 지원이나 고객(顧客) 지원용으로 사용됩니다.

VT_I4 cmmAdvReadRegister ([in] VT_I4 Axis, [in] VT_I4 RegisterNo, [out] VT_PI4 RegVal)
Undocument Function 입니다. 이 함수는 기술 지원이나 고객(顧客) 지원용으로 사용됩니다.

VT_I4 cmmAdvGetMioCfg1Dword ([in] VT_I4 Axis, [out] VT_PI4 Mio1Dword)
Undocument Function 입니다. 이 함수는 기술 지원이나 고객(顧客) 지원용으로 사용됩니다.

VT_I4 cmmAdvSetMioCfg1Dword ([in] VT_I4 Axis, [in] VT_I4 Mio1Dword)
Undocument Function 입니다. 이 함수는 기술 지원이나 고객(顧客) 지원용으로 사용됩니다.

VT_I4 cmmAdvSetToolboxMode ([in] VT_I4 EnInterrupt)
Undocument Function 입니다. 이 함수는 기술 지원이나 고객(顧客) 지원용으로 사용됩니다.

VT_I4 cmmAdvGetString ([in] VT_I4 Axis, [in] VT_I4 StringID, [out] VT_STR szBuffer)
Undocument Function 입니다. 이 함수는 기술 지원이나 고객(顧客) 지원용으로 사용됩니다.

VT_I4 cmmAdvErcOut ([in] VT_I4 Axis, [in] VT_I4 IsWaitOff)
Undocument Function 입니다. 이 함수는 기술 지원이나 고객(顧客) 지원용으로 사용됩니다.

VT_I4 cmmAdvErcReset ([in] VT_I4 Axis)
Undocument Function 입니다. 이 함수는 기술 지원이나 고객(顧客) 지원용으로 사용됩니다.

VT_I4 cmmAdvSetExtOptions ([in] VT_I4 OptionId, [in] VT_I4 lParam1, [in] VT_I4 lParam2,
[in] VT_R8 fParam1, [in] VT_R8 fParam2)
Undocument Function 입니다. 이 함수는 기술 지원이나 고객(顧客) 지원용으로 사용됩니다.

VT_I4 cmmAdvEnumMotDevices ([out] TMotDevEnum *EnumBuffer)
Undocument Function 입니다. 이 함수는 기술 지원이나 고객(顧客) 지원용으로 사용됩니다.

VT_I4 cmmAdvGetMotDevMap ([out] TMotDevMap *MapBuffer)
Undocument Function 입니다. 이 함수는 기술 지원이나 고객(顧客) 지원용으로 사용됩니다.

VT_I4 cmmAdvEnumDioDevices ([out] TDioDevMap *EnumBuffer)
Undocument Function 입니다. 이 함수는 기술 지원이나 고객(顧客) 지원용으로 사용됩니다.

VT_I4 cmmAdvGetDioDevMap ([out] TDioDevMap *MapBuffer)
Undocument Function 입니다. 이 함수는 기술 지원이나 고객(顧客) 지원용으로 사용됩니다.

VT_I4 cmmAdvInitFromCmeBuffer ([out] TCmeData_V2 * pCmeBuffer)
Undocument Function 입니다. 이 함수는 기술 지원이나 고객(顧客) 지원용으로 사용됩니다.

VT_I4 cmmAdvInitFromCmeBuffer_MapOnly ([out] TCmeData_V2 * pCmeBuffer, [in] VT_I4 nMapType)
Undocument Function 입니다. 이 함수는 기술 지원이나 고객(顧客) 지원용으로 사용됩니다.

VT_I4 cmmAdvGetLatestCmeFile([in] VT_STR szCmeFile)
Undocument Function 입니다. 이 함수는 기술 지원이나 고객(顧客) 지원용으로 사용됩니다.

VT_I4 cmmAdvGetAxisCapability([in] VT_I4 Channel, [in] VT_I4 CapId, [out] VT_PI4 CapBuffer)
Undocument Function 입니다. 이 함수는 기술 지원이나 고객(顧客) 지원용으로 사용됩니다.

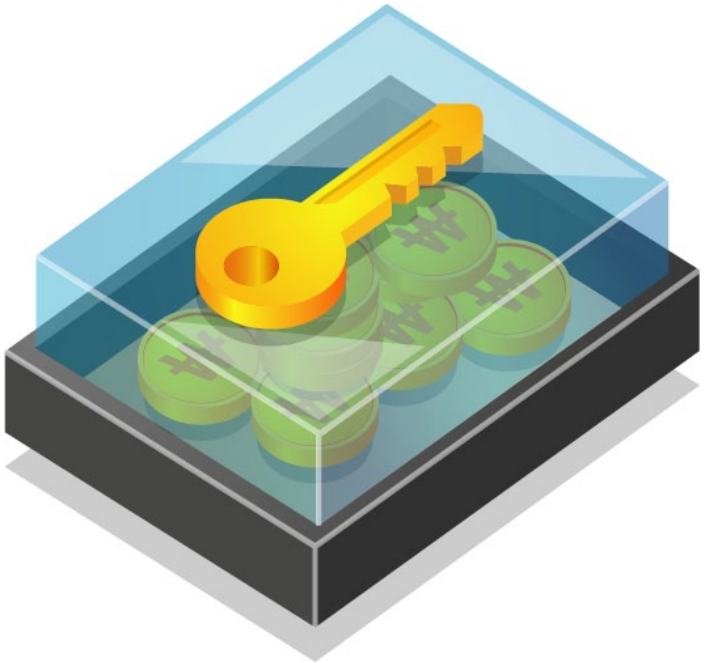
Chapter

15

Debugging and Error Handle Utility Functions

고객(顧客)님의 소중한 응용프로그램에서 모션 라이브러리가 그 기능의 일부를 담당한다면, 안전성과 신뢰성이 결여될 수는 없습니다. 수년간의 다양한 개발 요건들을 충족시켜온 저희㈜커미조아의 모션 라이브러리 기술은 이제 기능적 모션 라이브러리의 기능을 넘어, 감각적 모션 라이브러리의 기능을 추구하고 있습니다. 모션 라이브러리가 가질 수 있는 진정한 세계로 지금 고객(顧客)님을 초대합니다.

최단 시간내에 개발되어야 하는 고객(顧客) 社의 제품 개발에는 분명 개발 과정의 다양한 변수들이 존재하고 변수들이 존재하고 있습니다. CMMSDK 는 개발 과정 전후에 발생할 수 있는 모든 모션 소프트웨어의 소프트웨어의 장애물들을 제거하기 위해 존재하는 강력한 디버그 관련 기능을 제공합니다. 또한 실제적인 또한 실제적인 윈도우 응용프로그램에서 필요한 각종 유틸리티 함수 지원을 통해 고객(顧客) 여러분들에 고객(顧客) 여러분들에 조력자(助力者) 역할을 아끼지 않겠습니다.



15 디버그, 에러 처리 및 유틸리티 함수 편

15.1 디버그 지원

㈜커미조아 모션컨트롤러는 개발자에게 보다 편리한 개발환경을 제공하기 위해 다양한 기능을 바탕으로 "디버그 로그 (Debug Log)" 생성 기능을 제공합니다.

"디버그 로그(Debug Log)" 기능은 사용자의 프로그램에서 모션 라이브러리의 함수가 호출할 때마다 그 시각과 호출된 내용 그리고 그 순간의 에러 상황 등을 지정한 로그 파일에 기록하는 기능입니다.

디버그 기능과 관련된 함수들은 다음과 같습니다.

15.1.1 함수 요약

Summary of Functions
❒ VT_I4 cmmDlogSetup ([in] VT_I4 Level, [in] VT_STR szLogFile) 디버그 로그 기능의 환경 설정(環境設定)을 구성합니다. 디버그 로그 레벨과 로그 파일명을 지정합니다.
❒ VT_I4 cmmDlogAddComment ([in] VT_STR szComment) 디버그 로그시에 사용자 정의 주석(Comment) 를 추가 (追加) 합니다.
❒ VT_I4 cmmDlogGetCurLevel ([out] VT_PI4 CurLevel) 디버그 로그시에 현재 설정(設定)된 디버그 로그 레벨을 반환(返還)합니다.
❒ VT_I4 cmmDlogGetCurFilePath ([out] VT_STR szFilePath) 디버그 로그시에 현재 설정(設定)된 디버그 로그 파일 이름을 반환(返還)합니다.
❒ VT_I4 cmmDlogEnterManMode ([in] VT_I4 nMode) 수동 로깅 제어 모드 시작 지점을 설정 합니다.
❒ VT_I4 cmmDlogExitManMode ([none] VT_EMPTY) 수동 로깅 제어 모드 종료 지점을 설정 합니다.

15.1.2 함수 설명

NAME

cmmDlogSetup
– 디버그 로그(Log) 기능 설정(設定)

INFORMATION

- Debug and Error Handle
- VC++/VB
- BCB/Delphi/.NET
- Level 2
- 위험 요소 없음

SYNOPSIS

❒ VT_I4 cmmDlogSetup
([in] VT_I4 Level, [in] VT_STR szLogFile)

DESCRIPTION

"디버그로깅" 기능의 환경을 설정합니다. 디버깅로깅 기능의 활성/비활성, 그리고 디버깅레벨, 그리고 로깅 파일명 등을 설정합니다.

PARAMETER

▶ ***Level***: 디버깅 레벨을 설정합니다.

Value	Meaning
0	디버그로깅 기능을 Disable 시킵니다.
1	디버깅 레벨을 LEVEL1 으로 설정합니다.
2	디버깅 레벨을 LEVEL2 으로 설정합니다.
3	디버깅 레벨을 LEVEL3 으로 설정합니다.

▶ ***szLogFile*** : 로깅할 파일패스(File path)을 지정합니다.

RETURN VALUE

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

REFERENCE

□ 로깅레벨이 디버그로그에 미치는 영향은 다음과 같습니다.

• LEVEL1 : 디버그 로깅 기능을 Enable 시키고, 로깅되는 함수 유형을 Command 와 관련된 함수들로 한정합니다. Command 와 관련된 함수는 모든 환경을 설정하는 함수 또는 모션을 명령하는 함수들을 말합니다.

• LEVEL2 : 디버그 로깅 기능을 Enable 시키고, 로깅되는 함수 유형을 일부 함수를 제외한 모든 함수들로 설정합니다. 이때 로깅에서 제외되는 함수들은 cmmSxIsDone, cmmMxIsDone, cmmIxIsDone, cmmStGetCount, cmmStGetPosition, cmmStGetSpeed, cmmStReadMotionState, cmmStReadMioStatuses 함수 등과 같이 빈번하게 호출될 수 있는 모션의 상태 확인(確認) 함수들입니다.

• LEVEL3 : 디버그 로깅 기능을 Enable 시키고, 호출되는 모든 함수들을 로깅하도록 합니다.

NAME

cmmDlogAddComment
– 디버그 로그(Log)에 사용자(使用者) 정의(定義) 주석 추가 (追加)

INFORMATION

- Debug and Error Handle
- VC++/VB
- BCB/Delphi/.NET
- Level 1
- 위험 요소 없음

SYNOPSIS

❒ VT_I4 cmmDlogAddComment ([in] VT_STR szComment)

DESCRIPTION

"디버그로깅" 기능이 활성화되었을 때 주석(Comment)을 추가합니다.

PARAMETER

▶ *szComment*: 추가할 주석문자열

RETURN VALUE

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

NAME

cmmDlogGetCurLevel
- 설정(設定) 된 디버그 로그(Log) 레벨 확인(確認)

INFORMATION

Debug and Error Handle

VC++/VB

BCB/Delphi/.NET

Level 1

위험 요소 없음

SYNOPSIS

❒ VT_I4 cmmDlogGetCurLevel ([out] VT_PI4 CurLevel)

DESCRIPTION

현재 설정된 디버깅 레벨을 반환합니다.

RETURN VALUE

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

REFERENCE

□ 로깅레벨이 디버그로그에 미치는 영향은 다음과 같습니다.

• LEVEL1 : 디버그 로깅 기능을 Enable 시키고, 로깅되는 함수 유형을 Command 와 관련된 함수들로 한정합니다. Command 와 관련된 함수는 모든 환경을 설정하는 함수 또는 모션을 명령하는 함수들을 말합니다.

• LEVEL2 : 디버그 로깅 기능을 Enable 시키고, 로깅되는 함수 유형을 일부 함수를 제외한 모든 함수들로 설정합니다. 이때 로깅에서 제외되는 함수들은 cmmSxIsDone, cmmMxIsDone, cmmIxIsDone, cmmStGetCount, cmmStGetPosition, cmmStGetSpeed, cmmStReadMotionState, cmmStReadMioStatuses 함수 등과 같이 빈번하게 호출될 수 있는 모션의 상태 확인(確認) 함수들입니다.

• LEVEL3 : 디버그 로깅 기능을 Enable 시키고, 호출되는 모든 함수들을 로깅하도록 합니다.

NAME

cmmDlogGetCurFilePath
– 설정 (設定) 된 디버그 로그(Log) 파일 경로 확인(確認)

INFORMATION

- Debug and Error Handle
- VC++/VB
- BCB/Delphi/.NET
- Level 1
- 위험 요소 없음

SYNOPSIS

❒ VT_I4 cmmDlogGetCurFilePath
([out] VT_STR szFilePath)

DESCRIPTION

현재 디버그로그 파일로 설정된 파일의 경로를 얻어옵니다.

PARAMETER

▶ ***szFilePath*** : 디버그로그 파일 경로.

RETURN VALUE

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

NAME

cmmDlogEnterManMode
- 수동 로깅 제어 모드 시작 지점 설정

INFORMATION

- Debug and Error Handle
- VC++/VB
- BCB/Delphi/.NET
- Level 2
- ☺ 위험 요소 없음

SYNOPSIS

❒ VT_I4 cmmDlogEnterManMode([in] VT_I4 nMode)

DESCRIPTION

cmmDlogEnterManMode() 함수 호출을 통해 디버그 로깅 기능이 활성화 된 상태에서 원하는 특정 모션 설정 및 제어 부분에 대해 로깅 모드를 변경 하는 기능의 시작 점을 설정 합니다.
반드시 cmmDlogExitManMode() 함수와 짝을 이루어 사용하셔야 합니다.

PARAMETER

▶ nMode : Manual log control mode 설정

Value	Meaning
0	수동 로깅 제어 모드 설정을 무효화 시킵니다. (사용 하지 않음)
1	cmmDlogExitManMode() 함수가 호출될 때까지 로깅을 Skip 시킵니다.
2	cmmDlogExitManMode() 함수가 호출될 때까지 모든 Logging Level 의 모든 함수에 대한 로깅을 합니다.

SEE ALSO

cmmDlogExitManMode

RETURN VALUE

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

NAME

cmmDlogExitManMode
- 수동 로깅 제어 모드 종료 지점 설정

INFORMATION

Debug and Error Handle
VC++/VB
BCB/Delphi/.NET
Level 2
위험 요소 없음

SYNOPSIS

❒ VT_I4 cmmDlogExitManMode([none] VT_EMPTY)

DESCRIPTION

cmmDlogEnterManMode() 함수 호출을 통해 디버그 로깅 기능이 활성화 된 상태에서 원하는 특정 모션 설정 및 제어 부분에 대해 로깅 모드를 변경 하는 기능의 종료 점을 설정 합니다.
반드시 cmmDlogEnterManMode() 함수와 짝을 이루어 사용하셔야 합니다.

PARAMETER

▶ 없음

SEE ALSO

cmmDlogEnterManMode

RETURN VALUE

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

15.2 에러처리 함수

㈜커미조아 모션컨트롤러는 개발자에게 모든 모션 라이브러리 함수에 대한 에러 처리를 보다 편리하게 하도록 제공하기 위해서 에러 코드 및 에러가 발생한 축, 에러 이유 및 에러 문자열을 얻을 수 있는 함수를 제공합니다.

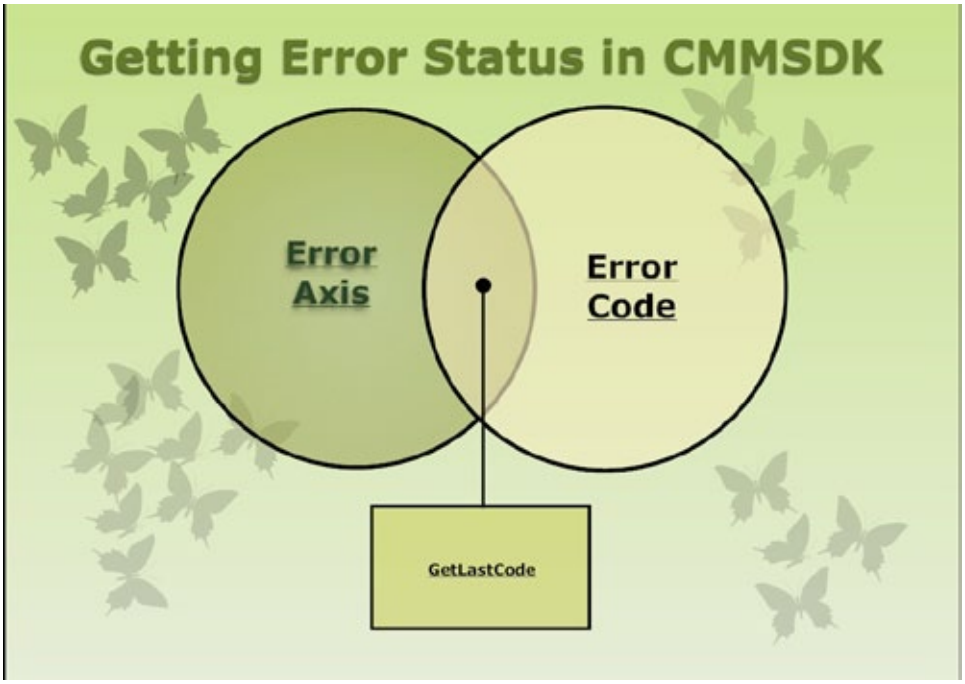


그림 15-1 CMMSDK 의 Error Status 구조

일반적인 모션 응용프로그램에서 CMMSDK 를 통한 에러 상태를 판단할 때에는 먼저 cmmErrGetLastCode 를 통해 원본(Source) 에러 코드를 확인합니다. 이후 다음과 같은 과정으로 에러를 확인 할 수 있습니다.

cmmErrParseAxis 의 매개변수(媒介變數)로 cmmErrGetLastCode 가 제공한 원본 에러코드를 통해 에러를 발생시킨 에러 모션 축을 확인할 수 있습니다.

cmmErrParseReason 의 매개변수(媒介變數)로 cmmErrGetLastCode 가 제공한 원본 에러코드를 통해 최종(最終) 에러 코드를 확인 할 수 있습니다.

마지막으로, cmmErrGetString 의 매개변수(媒介變數)로 cmmErrGetLastCode 가 제공한 원본 에러코드를 통해 해당 에러에 대한 내정된 에러 문자열을 확인 할 수 있습니다.

15.2.1 함수 요약

에러처리 기능과 관련된 함수들은 다음과 같습니다.

Summary of Functions
❒ VT_I4 cmmErrGetLastCode ([out] VT_PI4 ErrorCode) 마지막 발생한 에러 코드를 확인(確認)합니다.

❒ VT_I2 cmmErrParseAxis ([in] VT_I4 ErrorCode)
에러코드로부터 에러를 유발한 축 번호를 얻습니다.

❒ VT_I2 cmmErrParseReason ([in] VT_I4 ErrorCode)
에러코드로부터 에러 원인을 확인(確認)합니다.

❒ VT_I4 cmmErrGetString ([in] VT_I4 ErrorCode, [out] VT_STR Buffer, [in] VT_I4 BufferLen)
에러코드로부터 에러 문자열(文字列)을 확인(確認)합니다.

❒ VT_I4 cmmErrShowLast ([in] VT_HANDLE ParentWnd)
마지막으로 발생한 에러를 화면(畵面) 에 나타냅니다.

❒ VT_I4 cmmErrSetSkipShowMessage ([in] VT_I4 IsSkip)
에러메시지를 화면(畵面)에 나타내지 않도록 합니다.

❒ VT_I4 cmmErrGetSkipShowMessage ([out] VT_PI4 IsSkip)
에러메시지를 화면(畵面)에 나타내지 않도록 설정된 내용을 확인(確認)합니다.

❒ VT_I4 cmmErrSetEnableAutoMessage ([in] VT_I4 Enable)
에러 발생시 화면(畵面)에 자동으로 에러 메시지를 나타낼 것인지에 대한 설정(設定)을 합니다.

❒ VT_I4 cmmErrGetEnableAutoMessage ([out] VT_PI4 Enable)
에러 발생시 화면(畵面)에 자동으로 에러 메시지를 나타낼 것인지에 대한 설정(設定)을 확인(確認)합니다.

15.2.2 함수 설명

NAME

cmmErrGetLastCode
– 마지막에 발생한 에러코드 확인(確認)

INFORMATION

Error Handle
VC++/VB
BCB/Delphi/.NET
Level 1
위험 요소 없음

SYNOPSIS

❒ VT_I4 cmmErrGetLastCode ([out] VT_PI4 ErrorCode)

DESCRIPTION

마지막으로 발생한 에러코드를 확인(確認)합니다.

PARAMETER

▶ ***ErrorCode*** : 마지막으로 발생한 에러코드 값.

RETURN VALUE

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

EXAMPLE

C/C++

```
void SomeFunction(...)
{
   char szTemp[254+1];

   // 각 변수는 2 바이트형임을 주의합니다.
   short int ErrorParseAxis = 0;
   short int ErrorParseReason = 0;

   LONG dwErrCode;

   // Error  코드를 확인합니다.
   cmmErrGetLastCode(&dwErrCode);

   // Error 코드를 통해 에러를 발생시킨 축의 번호를 얻어옵니다.
   ErrorParseAxis = cmmErrParseAxis(dwErrCode);

   // Error 코드를 통해 에러를 발생시킨 에러번호(Error Code)를 얻어옵니다.
   ErrorParseReason = cmmErrParseReason(dwErrCode);

   sprintf(szTemp,"[LastErrorCode %d], [ErrorParseAxis : %d],
                  [ErrorParseReason %d]",dwErrCode,ErrorParseAxis,ErrorParseReason);
}
```

Visual Basic

```
Private Sub SomeFunction(…)

        Dim szTemp(254+1) As Byte
        Dim ErrorParseAxis As Integer
        Dim ErrorParseReason As Integer
        Dim dwErrCode As Long

        ErrorParseAxis = 0
        ErrorParseReason = 0

        ‘ Error  코드를 확인합니다.
        Call cmmErrGetLastCode(dwErrCode)

        ‘ Error 코드를 통해 에러를 발생시킨 축의 번호를 얻어옵니다.
        ErrorParseAxis = cmmErrParseAxis(dwErrCode)

        ‘ Error 코드를 통해 에러를 발생시킨 에러번호(Error Code)를 얻어옵니다.
        ErrorParseReason = cmmErrParseReason(dwErrCode)

End Sub
```

```
Delphi

Procedure SomeFunction(…);
var
        szTemp : array[0..255] of Char;
        ErrorParseAxis : SmallInt;
        ErrorParseReason : SmallInt;
        dwErrCode : LongInt;

begin

        // 각 변수는 2 바이트형임을 주의합니다.
        ErrorParseAxis := 0;
        ErrorParseReason := 0;

        // Error  코드를 확인합니다.
        cmmErrGetLastCode(@dwErrCode);

        // Error 코드를 통해 에러를 발생시킨 축의 번호를 얻어옵니다.
        ErrorParseAxis := cmmErrParseAxis(dwErrCode);

        // Error 코드를 통해 에러를 발생시킨 에러번호(Error Code)를 얻어옵니다.
        ErrorParseReason := cmmErrParseReason(dwErrCode);

        ShowMessage(Format('[LastErrorCode %d],[ErrorParseAxis : %d],
                                [ErrorParseReason %d]',[ dwErrCode,ErrorParseAxis,ErrorParseReason]));

end;
```

NAME

cmmErrClearLastCode
- 마지막에 발생한 에러코드 초기화

INFORMATION

Error Handle

VC++/VB

BCB/Delphi/.NET

Level 1

위험 요소 없음

SYNOPSIS

❒ VT_I4 cmmErrClearLastCode([none] VT_EMPTY)

DESCRIPTION

cmmErrClearLastCode() 함수는 마지막으로 발생한 에러 코드를 초기화 합니다.

SEE ALSO

cmmErrGetLastCode

RETURN VALUE

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

EXAMPLE

C/C++

```
void SomeFunction(...)
{
   char szTemp[254+1];

   // 각 변수는 2 바이트형임을 주의합니다.
   short int ErrorParseAxis = 0;
   short int ErrorParseReason = 0;

   LONG dwErrCode;

   // Error  코드를 확인합니다.
   cmmErrGetLastCode(&dwErrCode);

   // Error 코드를 통해 에러를 발생시킨 축의 번호를 얻어옵니다.
   ErrorParseAxis = cmmErrParseAxis(dwErrCode);

   // Error 코드를 통해 에러를 발생시킨 에러번호(Error Code)를 얻어옵니다.
   ErrorParseReason = cmmErrParseReason(dwErrCode);

   sprintf(szTemp,"[LastErrorCode %d], [ErrorParseAxis : %d],
                  [ErrorParseReason %d]",dwErrCode,ErrorParseAxis,ErrorParseReason);

   cmmErrClearLastCode(); // 마지막으로 발생한 에러 코드를 초기화 시킵니다.
}
```

Visual Basic

```
Private Sub SomeFunction(…)

        Dim szTemp As String
        Dim ErrorParseAxis As Integer
        Dim ErrorParseReason As Integer
        Dim dwErrCode As Long

        ErrorParseAxis = 0
        ErrorParseReason = 0

        szTemp = Space(255)

        ‘ Error  코드를 확인합니다.
        Call cmmErrGetLastCode(dwErrCode)

        ‘ Error 코드를 통해 에러를 발생시킨 축의 번호를 얻어옵니다.
        ErrorParseAxis = cmmErrParseAxis(dwErrCode)

        ‘ Error 코드를 통해 에러를 발생시킨 에러번호(Error Code)를 얻어옵니다.
        ErrorParseReason = cmmErrParseReason(dwErrCode)

        szTemp = “[LastErrorCode “ & dwErrCode & “]” & [ErrorParseAxis “ & ErrorParseAxis & “]”
                    & [ErrorParseReason “ & ErrorParseReason&” ]”

        Call cmmErrClearLastCode()
        ‘ 마지막으로 발생한 에러 코드를 초기화 시킵니다.

End Sub
```

Delphi

```
procedure SomeFunction (…);
var
        szTemp : Array[0 to 255] of Char;
        ErrorParseAxis : LongInt;
        ErrorParseReason : LongInt;
        dwErrCode : LongInt;

begin

        ErrorParseAxis := 0;
        ErrorParseReason := 0;

        // Error  코드를 확인합니다.
        cmmErrGetLastCode(&dwErrCode);

        // Error 코드를 통해 에러를 발생시킨 축의 번호를 얻어옵니다.
        ErrorParseAxis = cmmErrParseAxis(dwErrCode);

        // Error 코드를 통해 에러를 발생시킨 에러번호(Error Code)를 얻어옵니다.
        ErrorParseReason = cmmErrParseReason(dwErrCode);

        ShowMessage(Format('[LastErrorCode %d],[ErrorParseAxis : %d],
                                [ErrorParseReason %d]',[ dwErrCode,ErrorParseAxis,ErrorParseReason]));

        cmmErrClearLastCode(); // 마지막으로 발생한 에러 코드를 초기화 시킵니다.

end;
```

NAME

cmmErrParseAxis
– 에러가 발생된 축(Axis) 번호 확인(確認)

INFORMATION

Error Handle
VC++/VB
BCB/Delphi/.NET
Level 1
위험 요소 없음

SYNOPSIS

❒ VT_I2 cmmErrParseAxis ([in] VT_I4 ErrorCode)

DESCRIPTION

에러코드를 통해 에러(Error)를 유발한 축 번호를 얻습니다. 모션의 이송에 관련된 에러코드에는 에러의 유형뿐만 아니라 에러를 유발한 축번호에 대한 정보도 함께 들어 있습니다.

PARAMETER

▶ ***ErrorCode*** : 마지막으로 발생한 에러코드 값.

RETURN VALUE

에러를 유발한 축 번호입니다. 단, 특정 축에 대한 에러가 아닌 경우에는 -1 을 반환하게 됩니다.

REFERENCE

□ 이 함수는 cmmErrParseReason 와 함께 단 두개 함수만이 리턴타입이 VT_I2 이며, 이 리턴값이 다른 함수들처럼 VT_I4 형태의 에러코드가 아님을 주의해주시기 바랍니다.

EXAMPLE

C/C++

```
void SomeFunction(...)
{
   char szTemp[254+1];

   // 각 변수는 2 바이트형임을 주의합니다.
   short int ErrorParseAxis = 0;
   short int ErrorParseReason = 0;
   LONG dwErrCode;

   // Error  코드를 확인합니다.
   cmmErrGetLastCode(&dwErrCode);

   // Error 코드를 통해 에러를 발생시킨 축의 번호를 얻어옵니다.
   ErrorParseAxis = cmmErrParseAxis(dwErrCode);

   // Error 코드를 통해 에러를 발생시킨 에러번호(Error Code)를 얻어옵니다.
   ErrorParseReason = cmmErrParseReason(dwErrCode);

   sprintf(szTemp,"[LastErrorCode %d], [ErrorParseAxis : %d],
                  [ErrorParseReason %d]",dwErrCode,ErrorParseAxis,ErrorParseReason);
```

```
}
```

Visual Basic

```
Private Sub SomeFunction(…)

        Dim szTemp(254+1) As Byte
        Dim ErrorParseAxis As Integer
        Dim ErrorParseReason As Integer
        Dim dwErrCode As Long

        ErrorParseAxis = 0
        ErrorParseReason = 0

        ‘ Error  코드를 확인합니다.
        Call cmmErrGetLastCode(dwErrCode)

        ‘ Error 코드를 통해 에러를 발생시킨 축의 번호를 얻어옵니다.
        ErrorParseAxis = cmmErrParseAxis(dwErrCode)

        ‘ Error 코드를 통해 에러를 발생시킨 에러번호(Error Code)를 얻어옵니다.
        ErrorParseReason = cmmErrParseReason(dwErrCode)

End Sub
```

Delphi

```
Procedure SomeFunction(…);
var
        szTemp : array[0..255] of Char;
        ErrorParseAxis : SmallInt;
        ErrorParseReason : SmallInt;
        dwErrCode : LongInt;

begin

        // 각 변수는 2 바이트형임을 주의합니다.
        ErrorParseAxis := 0;
        ErrorParseReason := 0;

        // Error  코드를 확인합니다.
        cmmErrGetLastCode(@dwErrCode);

        // Error 코드를 통해 에러를 발생시킨 축의 번호를 얻어옵니다.
        ErrorParseAxis := cmmErrParseAxis(dwErrCode);

        // Error 코드를 통해 에러를 발생시킨 에러번호(Error Code)를 얻어옵니다.
        ErrorParseReason := cmmErrParseReason(dwErrCode);

        ShowMessage(Format('[LastErrorCode %d],[ErrorParseAxis : %d],
                                [ErrorParseReason %d]',[ dwErrCode,ErrorParseAxis,ErrorParseReason]));

end;
```

NAME

cmmErrParseReason
– 에러 코드로부터 오류 원인 (原因) 확인(確認)

INFORMATION

Error Handle
VC++/VB
BCB/Delphi/.NET
Level 1
위험 요소 없음

SYNOPSIS

❒ VT_I2 cmmErrParseReason ([in] VT_I4 ErrorCode)

DESCRIPTION

에러코드로부터 에러(Error) 원인(Reason)을 얻습니다.

PARAMETER

▶ ***ErrorCode*** : 마지막으로 발생한 에러코드 값.

RETURN VALUE

발생한 에러에 대한 원인 코드.

REFERENCE

□ 이 함수는 cmmErrParseReason 와 함께 단 두개 함수만이 리턴타입이 VT_I2 이며, 이 리턴값이 다른 함수들처럼 VT_I4 형태의 에러코드가 아님을 주의해주시기 바랍니다.

EXAMPLE

C/C++

```
void SomeFunction(...)
{
   char szTemp[254+1];

   // 각 변수는 2 바이트형임을 주의합니다.
   short int ErrorParseAxis = 0;
   short int ErrorParseReason = 0;

   LONG dwErrCode;

   // Error  코드를 확인합니다.
   cmmErrGetLastCode(&dwErrCode);

   // Error 코드를 통해 에러를 발생시킨 축의 번호를 얻어옵니다.
   ErrorParseAxis = cmmErrParseAxis(dwErrCode);

   // Error 코드를 통해 에러를 발생시킨 에러번호(Error Code)를 얻어옵니다.
   ErrorParseReason = cmmErrParseReason(dwErrCode);

   sprintf(szTemp,"[LastErrorCode %d], [ErrorParseAxis : %d],
                  [ErrorParseReason %d]",dwErrCode,ErrorParseAxis,ErrorParseReason);
```

```
}
```

Visual Basic

```
Private Sub SomeFunction(…)

        Dim szTemp(254+1) As Byte
        Dim ErrorParseAxis As Integer
        Dim ErrorParseReason As Integer
        Dim dwErrCode As Long

        ErrorParseAxis = 0
        ErrorParseReason = 0

        ‘ Error  코드를 확인합니다.
        Call cmmErrGetLastCode(dwErrCode)

        ‘ Error 코드를 통해 에러를 발생시킨 축의 번호를 얻어옵니다.
        ErrorParseAxis = cmmErrParseAxis(dwErrCode)

        ‘ Error 코드를 통해 에러를 발생시킨 에러번호(Error Code)를 얻어옵니다.
        ErrorParseReason = cmmErrParseReason(dwErrCode)

End Sub
```

Delphi

```
Procedure SomeFunction(…);
var
        szTemp : array[0..255] of Char;
        ErrorParseAxis : SmallInt;
        ErrorParseReason : SmallInt;
        dwErrCode : LongInt;

begin

        // 각 변수는 2 바이트형임을 주의합니다.
        ErrorParseAxis := 0;
        ErrorParseReason := 0;

        // Error  코드를 확인합니다.
        cmmErrGetLastCode(@dwErrCode);

        // Error 코드를 통해 에러를 발생시킨 축의 번호를 얻어옵니다.
        ErrorParseAxis := cmmErrParseAxis(dwErrCode);

        // Error 코드를 통해 에러를 발생시킨 에러번호(Error Code)를 얻어옵니다.
        ErrorParseReason := cmmErrParseReason(dwErrCode);

        ShowMessage(Format('[LastErrorCode %d],[ErrorParseAxis : %d],
                                [ErrorParseReason %d]',[ dwErrCode,ErrorParseAxis,ErrorParseReason]));

end;
```

NAME

cmmErrGetString
– 에러 코드를 통한 에러 문자열(String) 반환(返還)

INFORMATION

Error Handle
VC++/VB
BCB/Delphi/.NET
Level 1
위험 요소 없음

SYNOPSIS

❒ VT_I4 cmmErrGetString ([in] VT_I4 ErrorCode, [out] VT_STR Buffer, [in] VT_I4 BufferLen)

DESCRIPTION

에러코드로부터 에러 문자열을 전달된 매개변수(媒介變數)를 통해 확인(確認)합니다.

PARAMETER

▶ ***ErrorCode*** : 마지막으로 발생한 에러코드 값.

▶ ***Buffer*** : 에러 문자열을 전달 받을 문자열 버퍼 공간.

▶ ***BufferLen*** : 문자열 버퍼의 길이

RETURN VALUE

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

NAME

cmmErrShowLast
– 최종(最終) 발생한 에러에 대한
문자열(String) 화면표시(畵面表示)

INFORMATION

Error Handle
VC++/VB
BCB/Delphi/.NET
Level 1
위험 요소 없음

SYNOPSIS

❒ VT_I4 cmmErrShowLast ([in] VT_HANDLE ParentWnd)

DESCRIPTION

마지막으로 발생한 에러에 대한 문자열을 화면(畵面)에 나타냅니다.

PARAMETER

▶ ***ParentWnd*** : 에러 메시지창을 띄울 부모 윈도우 핸들값을 지정.

RETURN VALUE

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

NAME

cmmErrSetSkipShowMessage
cmmErrGetSkipShowMessage
– 에러 표시 활성 상태 설정 및 확인(確認)

INFORMATION

- Error Handle
- VC++/VB
- BCB/Delphi/.NET
- Level 1
- 위험 요소 없음

SYNOPSIS

- VT_I4 cmmErrSetSkipShowMessage ([in] VT_I4 IsSkip)
- VT_I4 cmmErrGetSkipShowMessage ([out] VT_PI4 IsSkip)

DESCRIPTION

cmmErrSetSkipShowMessage() 함수는 cmmErrShowLast() 함수가 호출되었거나 또는 자동으로 에러 디스플레이 하는 옵션(cmmErrSetEnableAutoMessage())**이 활성화된 경우에도 에러 디스플레이를 생략하도록 하는 기능을 설정하는 것입니다.**
전체적으로는 자동으로 에러를 디스플레이하고 싶지만 일부 루틴에서는 에러가 너무 많이 디스플레이될 소지가 있어서 생략하고 싶은 경우에 이 함수를 사용할 수 있습니다. 또한 Emergency 가 걸려있는 경우에는 모든 이송 함수들이 에러를 디스플레이할 수 있으므로 이 함수를 이용하여 디스플레이를 생략할 수 있습니다.

PARAMETER

▶ ***IsSkip*** : cmmErrSetSkipShowMessage 함수의 인자이며, 이 값이 1 이면 에러 디스플레이 생략 기능이 활성되어 cmmErrShowLast() 함수는 무시됩니다. 또한 자동으로 에러를 디스플레이하라는 옵션도 무시됩니다. 이 값이 0 이면 에러 디스플레이 생략 기능을 비활성화합니다.

▶ ***IsSkip*** : cmmErrGetSkipShowMessage 함수의 인자이며, 에러 디스플레이 생략기능의 활성화 여부를 반환합니다. , 이 값이 1 이면 에러 디스플레이 생략 기능이 활성되어 cmmErrShowLast() 함수는 무시되는 상태입니다.

RETURN VALUE

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

NAME

cmmErrSetEnableAutoMessage
cmmErrGetEnableAutoMessage
– 자동 에러 표시 기능 설정 및 확인(確認)

INFORMATION

- Error Handle
- VC++/VB
- BCB/Delphi/.NET
- Level 1
- 위험 요소 없음

SYNOPSIS

- ❒ VT_I4 cmmErrSetEnableAutoMessage ([in] VT_I4 Enable)
- ❒ VT_I4 cmmErrGetEnableAutoMessage ([out] VT_PI4 Enable)

DESCRIPTION

자동으로 화면에 에러메시지창을 표시 하도록 설정 하거나, 설정 상태를 반환합니다.

PARAMETER

▶ Enable : cmmErrSetEnableAutoMessage 함수의 인자이며, 자동 에러메세지 표시 여부를 설정합니다.

Value	Meaning
0	자동으로 에러 메시지를 띄우지 않음.
1	자동으로 에러 메시지를 띄우도록 설정

▶ Enable : cmmErrGetEnableAutoMessage 함수의 인자이며, 자동 에러메세지 표시 여부를 반환합니다.

Value	Meaning
0	자동으로 에러 메시지를 띄우지 않는 상태
1	자동으로 에러 메시지를 띄우는 상태

RETURN VALUE

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

15.3 유틸리티 함수

㈜커미조아 모션컨트롤러는 개발자에게 특수한 처리를 위해 유틸리티 함수를 제공합니다. 이 함수들은 직접적인 모션 컨트롤러의 기능에는 관련이 없으나, 고객(顧客)님들께서 구현하시는 응용프로그램이나 관련된 응용 범위에서 유용하게 사용할 수 있는 주요한 기능으로 준비되었습니다. 최적화된 모션 라이브러리에 다양한 유틸리티 함수들을 이용함으로써 최적의 모션 소프트웨어를 구현할 수 있도록 안내하여 드립니다.

15.3.1 함수 요약

CMMSDK 에서 제공하는 유틸리티 함수들은 다음과 같습니다. 해당 함수 목록은 실제 모션 동작이나 범용 디지털 입출력 보다는 실제 응용프로그램의 개발(開發)에 있어, 고객(顧客)님들의 프로그램과 보다 효과적(效果的)으로 작용할 수 있는 기능을 제공합니다. 강력한 모션 라이브러리 기능을 바탕으로 제공되는 유틸리티 함수는 고객(顧客)님들의 응용프로그램에서 뛰어난 성능을 발휘합니다.

Summary of Functions
❒ VT_EMPTY cmmSetBit ([in] VT_I4 TargetVariable, [in] VT_I4 BitNumber, [in] VT_I4 State) 대상(對象) 변수의 특정 비트값을 설정하는 매크로 함수입니다.
❒ VT_I4 cmmGetBit ([in] VT_I4 TargetVariable, [in] VT_I4 BitNumber) 대상(對象) 변수의 특정 비트값을 반환하는 매크로 함수입니다.
❒ VT_I4 cmmUtlProcessWndMsgS ([in] VT_HANDLE WndHandle, [out] VT_PI4 IsEmpty) 단일(單一)윈도우 메시지를 처리합니다. 이 함수는 응용프로그램에서 단일 윈도우 메시지를 처리하기 위한 용도로 사용(使用)가능합니다.
❒ VT_I4 cmmUtlProcessWndMsgM ([in] VT_HANDLE WndHandle, [in] VT_I4 TimeOut, [out] VT_PI4 IsTimeOuted) 지정된 시간 내에 다중(多重) 윈도우 메시지를 처리합니다. 이 함수는 응용프로그램에서 다중(多重) 윈도우 메시지를 처리하기 위한 용도로 사용(使用) 가능합니다.
❒ VT_I4 cmmUtlDelayMicroSec ([in] VT_I4 Delay_us) 마이크로초 단위의 정확한 단위로 시스템 지연을 발생하여 주는 함수 입니다.
❒ VT_I4 cmmUtlReadUserTable ([in] VT_I4 nAddress, [out] VT_I4 nSize, UCHAR * pBuffer) CMMSDK 에서 제공하는 사용자(使用者) 데이터 테이블에 기록한 데이터를 읽어들이는 함수입니다.
❒ VT_I4 cmmUtlWriteUserTable ([in] VT_I4 nAddress, [in] VT_I4 nSize, UCHAR * pBuffer) CMMSDK 에서 제공하는 사용자(使用者) 데이터 테이블에 데이터를 기록하는 함수입니다

15.3.2 함수 설명

NAME

cmmSetBit
- 대상(對象) 변수의 특정 비트 값의 변경(變更)

INFORMATION

Utility Function

VC++/VB

BCB/Delphi/.NET

Level 1

위험 요소 없음

SYNOPSIS

❒ VT_EMPTY cmmSetBit

([in] VT_I4 TargetVariable, [in] VT_I4 BitNumber, [in] VT_I4 State)

DESCRIPTION

대상 변수의 특정 비트 값을 변경하기 위해서 사용되는 매크로 함수(Macro Function) 입니다. 해당 함수의 원형은 제공되는 헤더파일에 선언되어 있습니다. 이 매크로 함수를 사용하시면 특정 비트의 값을 편리하게 변경하실 수 있습니다. 이 함수는 VC++ 과 Borland C++ 사용자에게만 제공됩니다. Visual Basic 사용자께서는 cmmGnBitShift 를 사용하시기 바랍니다.

PARAMETER

▶ *TargetVariable* : 비트 값의 변경이 적용될 변수

▶ *BitNumber* : 변경할 비트 번호

▶ *State* : 비트 상태를 의미합니다. 0 이나 1 을 입력합니다.

RETURN VALUE

이 함수는 매크로 함수(Macro Function) 이며, 반환값이 존재하지 않습니다.

NAME

cmmGetBit
- 대상(對象) 변수의 특정 비트 값을 반환(返還)

INFORMATION

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

SYNOPSIS

❒ VT_I4 cmmGetBit
([in] VT_I4 TargetVariable, [in] VT_I4 BitNumber)

DESCRIPTION

대상 변수의 특정 비트 값을 반환하기 위해서 사용되는 매크로 함수(Macro Function) 입니다. 해당 함수의 원형은 제공되는 헤더파일에 선언되어 있습니다. 이 매크로 함수를 사용하시면 특정 비트의 값을 편리하게 반환 받을수 있습니다.

PARAMETER

▶ ***TargetVariable*** : 변수 명(Variable Name)

▶ ***BitNumber*** : 반환 받을 비트 번호

RETURN VALUE

이 함수는 매크로 함수(Macro Function) 이며, TargetVariable 의 BitNumber 에 해당하는 비트 값을 반환합니다.

NAME

cmmUtlProcessWndMsgS
- 단일(單一) 윈도우 메시지 처리(處理)

INFORMATION

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

SYNOPSIS

❒ VT_I4 cmmUtlProcessWndMsgS
([in] VT_HANDLE WndHandle, [out] VT_PI4 IsEmpty)

DESCRIPTION

단일 윈도우 메시지를 처리합니다. 이 함수는 단일 윈도우 메시지를 처리하며, 응용프로그램의 주소(Handle) 을 요구합니다.
실제 윈도우 메시지 처리가 필요한 상황에 대해서 'IsEmpty'의 매개변수로 확인(確認)할 수 있으며, 이 매개변수는 VT_PI4 형 인자를 요구합니다.

PARAMETER

▶ ***WndHandle*** : 대상 응용프로그램의 식별할 수 있는 주소, 혹은 폼의 핸들을 요구합니다.

▶ ***IsEmpty*** : 실제 처리할 윈도우 메시지가 응용프로그램 큐 혹은 쓰레드 큐에 존재하는지 확인(確認)합니다. 처리할 윈도우 메시지가 없으면, cmTRUE 를 반환합니다.

RETURN VALUE

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

NAME

cmmUtlProcessWndMsgM
- 다중(多重) 윈도우 메시지 처리(處理)

INFORMATION

Utility Function

VC++/VB

BCB/Delphi/.NET

Level 1

위험 요소 없음

SYNOPSIS

❒ VT_I4 cmmUtlProcessWndMsgM

([in] VT_HANDLE WndHandle, [in] VT_I4 TimeOut, [out] VT_PI4 IsTimeOuted)

DESCRIPTION

다중 윈도우 메시지를 처리합니다. 이 함수는 다중 윈도우 메시지를 처리하며, 응용프로그램의 주소(Handle) 을 요구합니다.
응용프로그램 혹은 쓰레드(Thread) 가 메시지 큐(Queue) 에 있는 윈도우 메시지 처리에 너무 많은 시간을 할당하지 않도록 TimeOut 을 밀리초(millisecond) 단위로 설정할 수 있습니다. 이 시간안에 지정된 다중 윈도우 메시지가 처리됩니다.
IsTimeOut 매개변수는 실제 다중 윈도우 메시지 처리에 대한 설정된 시간이 경과했는지에 대한 확인(確認)을 위한 매개변수 입니다.

PARAMETER

▶ ***WndHandle***: 대상 응용프로그램의 식별할 수 있는 주소, 혹은 폼의 핸들을 요구합니다.

▶ ***TimeOut***: 지정된 시간 (millisecond) 단위안에 다중 윈도우 메시지를 처리합니다.

▶ ***IsTimeOuted***: 이 전달인자로 지정된 시간이 경과되었는 지를 판단할 수 있습니다.

RETURN VALUE

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

NAME

cmmUtlDelayMicroSec
– 하드웨어적인 시스템 지연 발생(發生)

INFORMATION

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

SYNOPSIS

❒ VT_I4 cmmUtlDelayMicroSec
([in] VT_I4 Delay_us)

DESCRIPTION

지정된 마이크로초(Micro-sec) 단위의 시스템 지연을 발생시킵니다. 이 함수는 일반적인 x86 계열의 CPU 에서 하드웨어적인 정밀 계수기(High-Resolution Performance Counter) 의 주파수 값을 읽어, 매우 정확한 시간 단위의 시스템 지연을 발생시킬 수 있습니다.

PARAMETER

▶ ***Delay_us*** : 마이크로초(Micro-sec) 단위의 지연 시간

RETURN VALUE

Value	Meaning
0	수행 실패. 전달된 인자가 음수이거나, 시스템 하드웨어가 정밀 계수기를 지원하지 않음
실제 지연시간	마이크로초(Micro-sec) 단위의 실제 시스템 지연 시간

NAME

cmmUtlReadUserTable
– 사용자정의(使用者定義) 테이블 읽기 동작(動作)

INFORMATION

Utility Function

VC++/VB

BCB/Delphi/.NET

Level 1

다소 주의
이 함수는 별다른 용도가 필요하지 않다면 사용하지 않을 수 있습니다.

SYNOPSIS

❒ VT_I4 cmmUtlReadUserTable
([in] VT_I4 nAddress, [in] VT_I4 nSize, [out] UCHAR * pBuffer)

DESCRIPTION

이 함수는 cmmUtlWriteUserTable() 함수를 통하여 CMMSDK 에서 제공하는 사용자 데이터 테이블에 기록한 데이터를 읽어들이는 함수입니다.

CMMSDK 에서 제공하는 사용자 데이터 테이블은 COMI-LX5xx 장치의 드라이버에서 제공하는 적은양의 메모리 공간을 의미합니다.

COMI-LX5xx 장치의 드라이버 프로그램은 윈도우 O/S 가 부팅될 때 자동으로 실행되어서 윈도우가 종료될 때까지 실행 상태를 유지합니다. 따라서 CMMSDK 를 사용하는 응용프로그램이 종료되었다가 다시 실행되어도 사용자 데이터 테이블은 이전의 상태를 그대로 유지하게 됩니다. 그러므로 CMMSDK 에서 제공하는 사용자 데이터 테이블은 윈도우 O/S 가 실행되고 있는 동안에는 응용프로그램 입장에서는 비휘발성 메모리와 같은 역할을 수행하다가 윈도우 O/S 가 재부팅되면 그 값이 리셋되는 특성(特性)을 가지는 데이터 저장 공간으로 생각하면 됩니다.

사용자가 사용할 수 있는 사용자 테이블의 크기는 장치 하나당 100 바이트 입니다.

PARAMETER

▶ nAddress : 읽고자하는 사용자 데이터 테이블의 주소값을 지정합니다. 모션 장치 하나에서 제공하는 Address 크기는 100(0~99)입니다. 만일 모션 장치가 2 개가 있다면 Address 범위는 200(0~199)가 되고, 모션 장치가 N 개 있다면 Address 범위는 N*100 이 됩니다.

▶ nSize : 읽고자하는 데이터블럭의 크기를 바이트 단위로 지정합니다. 이 값은 100 보다 작거나 같아야 합니다.

▶ pBuffer : 읽은 데이터를 전달받을 버퍼를 지정합니다. 이 버퍼의 크기는 nSize 보다 크거나 같아야 합니다.

RETURN VALUE

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

NAME

cmmUtlWriteUserTable
– 사용자정의(使用者定義) 테이블 쓰기 동작(動作)

INFORMATION

Utility Function

VC++/VB

BCB/Delphi/.NET

Level 1

다소 주의
이 함수는 별다른 용도가 필요하지 않다면 사용하지 않을 수 있습니다.

SYNOPSIS

❒ VT_I4 cmmUtlWriteUserTable
([in] VT_I4 nAddress, [in] VT_I4 nSize, [in] UCHAR * pBuffer)

DESCRIPTION

이 함수는 CMMSDK 에서 제공하는 사용자 데이터 테이블에 데이터를 기록하는 함수입니다. 기록된 값은 cmmUtlReadUserTable() 함수를 통하여 읽을 수 있습니다.

CMMSDK 에서 제공하는 사용자 데이터 테이블은 COMI-LX5xx 장치의 드라이버에서 제공하는 적은양의 메모리 공간을 의미합니다.

COMI-LX50x 장치의 드라이버 프로그램은 윈도우 O/S 가 부팅될 때 자동으로 실행되어서 윈도우가 종료될 때까지 실행 상태를 유지합니다. 따라서 CMMSDK 를 사용하는 응용프로그램이 종료되었다가 다시 실행되어도 사용자 데이터 테이블은 이전의 상태를 그대로 유지하게 됩니다. 그러므로 CMMSDK 에서 제공하는 사용자 데이터 테이블은 윈도우 O/S 가 실행되고 있는 동안에는 응용프로그램 입장에서는 비휘발성 메모리와 같은 역할을 수행하다가 윈도우 O/S 가 재부팅되면 그 값이 리셋되는 특성(特性)을 가지는 데이터 저장 공간으로 생각하면 됩니다.

사용자가 사용할 수 있는 사용자 테이블의 크기는 장치 하나당 100 바이트 입니다.

PARAMETER

▶ nAddress : 쓰고자하는 사용자 데이터 테이블의 주소값을 지정합니다. 모션 장치 하나에서 제공하는 Address 크기는 100(0~99)입니다. 만일 모션 장치가 2 개가 있다면 Address 범위는 200(0~199)가 되고, 모션 장치가 N 개 있다면 Address 범위는 N*100 이 됩니다.

▶ nSize : 쓰고자하는 데이터블럭의 크기를 바이트 단위로 지정합니다. 이 값은 100 보다 작거나 같아야 합니다.

▶ pBuffer : 기록할 데이터를 담고 있는 버퍼를 지정합니다. 이 버퍼의 크기는 nSize 보다 크거나 같아야 합니다.

RETURN VALUE

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

TEST & MEASUREMENT & AUTOMATION / COMIZOA

CMMSDK Manual / Appendix

부록편 내용 요약

A. CME Builder 를 이용한 환경설정

CME Builder 는 복잡한 모션 주변 신호의 논리(Logic) 설정이나 모션과 디지털 입출력 채널의 순서를 실제 환경과 구성 유도 조건에 따라서 자유롭게 변경하는 맵핑(Mapping) 기능을 가지고 있는 다재 다능한(Multi-talented) 소프트웨어입니다. CMMSDK 의 Active X 형태의 라이브러리는 Form Based 기반의 윈도우 응용프로그램 개발시에 GUI 환경에서 환경설정(環境設定) 이 가능했지만, CMMSDK v2 에서는 해당하는 이점이 존재하지 않았습니다.

이를 보완하기 위해 CME Builder 는 각종 모션 및 디지털 입출력 환경설정(環境設定)을 GUI(Graphics User Interface) 상에서 가능하도록 하고 있습니다. 부록 0 에서는 CME Builder 에 대한 다양한 기능에 대해서 소개하며, 사용 방법과 실제 적용 방법을 상세히 안내받을 수 있습니다.

B.모션 장치초기화시의 초기(Default) 값

모션 장치의 초기화 상태는 고객(顧客) 여러분들께서 실제 제어하지 않는 신호 논리(Logic) 이나 설정 값(Configure Value) 등이 어떠한 값으로 초기에 설정되어 있는지를 안내하고 있습니다. 이 내용은 전체적인 모션 환경 설정과 디지털 입출력 환경설정시에 매우 중요하게 작용하는 부분으로서, 실수하기 쉬운 내용들을 보다 명료하고, 정확하게 정리하였습니다. 다양한 환경설정 사항에서 필요한 사항들을 반드시 참조하시어, 초기 설정 값을 통해 더욱더 명확한 모션 환경설정에 대한 이해(理解)와 도움이 되시기를 바랍니다.

C.에러코드(Error Codes)

CMMSDK v2 에서는 다양한 종류의 에러와 주의, 그리고 정상 상태에 대해서 명확한 코드 일람표로 통일화하여 고객(顧客)여러분들의 예외(Exception) 상황에 대한 처리를 돕고 있습니다.
모션 동작 시에 에러 상황에 대해서 보다 자세한 처리를 원하시거나, 유발된 에러의 속성을 판단하고자 하실 때 유용한 내용으로 구성되어 있습니다. 참고적으로 CMMSDK v2 에서 다루는 에러의 범위는 음수를 기준으로 한 값을 범위로 기준하기 때문에 소프트웨어의 코드 분류에 있어, 단순 2 진 상태에서는 음수에 대한 처리를 반드시 고려해 주시기 바랍니다.

D.다른 개발환경 사용 고객(顧客)님들을 위한 라이브러리 안내

커미조아 모션 라이브러리의 자유로운 결합은 Microsoft Visual C++, Visual Basic, C#, VB.NET 을 시작으로 Delphi, BCB 등을 기본으로 합니다. 이외에도 Digital Fortran 사용자를 위한 CMMSDK 사용과 Power Builder 사용자들을 돕고 있습니다. C# 을 비롯한 .NET Framework 의 고객(顧客)님들께서는 포함된 예제와 인터페이스 파일(Header File) 을 통해 매우 편리하게 CMMSDK v2 를 사용하실 수 있으며, 이외에 다양한 Managed 개발환경 역시 저희 ㈜ 커미조아 고객(顧客) 지원팀에서 그 사용에 있어 편의성과 고객(顧客) 만족을 도모하고 있습니다.

E. Frequently Asked Questions (FAQ)

고객님들께서 자주 접하시게 되는 질문 사항이나 현상에 대해서, 보다 빠르고 정확하게 대응하실 수 있도록 빈번하게 질문되는 내용에 대해서 다루고 있습니다. 실제 모션 구동 상황에서 발생할 수 있는 사항을 비롯하여, 하드웨어, 소프트웨어에 대한 문의 사항들을 저희 ㈜ 커미조아로 문의 하시기 전에 본 장을 통해서, 질문 사항에 대한 답변을 먼저 확인 하실 수 있도록 준비되어 있습니다.

F. Index of CMMSDK Functions

커미조아의 모션 라이브러리 매뉴얼에서 보다 쉽고 빠르게 함수를 확인할 수 있는 안내 페이지를 제공하고 있습니다. PDF 화된 문서에서 보다 빠른 시간으로 함수를 찾을 수 있도록 하이퍼 링크(Hyper Link) 기능을 제공하고 있으며, 페이지와 함수명이 바로 연결되었으므로, 편리하게 원하는 함수의 설명을 확인 할 수 있습니다.

Appendix

A

COMIZOA Motion Environment Builder

다양한 모션의 환경설정을 모두 코드로 구현하실 필요는 없어졌습니다. 윈도우의 그래픽 사용자 인터페이스에 충실한 커미조아 독자적인 환경설정 도구인 CME 빌더가 여러분들을 기다리고 있습니다. 모션 설정에 대한 다양한 변수를 코드로 모두 구현해야 하는 부담에서 이제 벗어 나십시요. CME 빌더는 환경 설정뿐만아니라 다양한 실제 환경설정 소스 코드 역시, 자유롭게 고객(顧客) 여러분들을 위해서 사용가능하도록 제공하여 드립니다.

본 장에서는 CME Builder 를 이용하여 CMMSDK 의 환경을 설정하는 것에 대한 자세한 설명을 수록하였습니다. 수록하였습니다. CME Builder 는 크게 Device Mappings, Axis Definition, Motion Setup 및 Digital I/O Setup Setup 부분으로 구성되어 있으며, 당사의 CMMSDK 를 사용시에 각종 속성을 런타임(Run-time)시에 코드를 time)시에 코드를 통해서만 설정하여 사용하는 단점을 보완하고자 각종 설정 방법을 GUI(Graphics User GUI(Graphics User Interface)환경을 통해 제공하므로써 보다 직관적이고 편리하게 속성을 설정할 수 있도록 제공되는 별도의 응용 프로그램입니다.

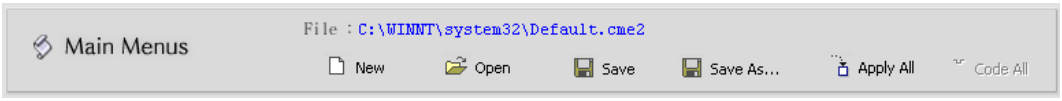
I CME Builder 를 이용한 환경설정

CME Builder 를 사용하면 프로그래머가 속도 패턴, 원점복귀시 설정 및 알람 로직 등에 대한 별도의 코드를 작성하지 않고 GUI(Graphic User Interface) 환경 하에서 각종 모션에 관련된 설정을 할 수 있으며, 설정된 내용을 환경파일을 통해 실제 고객(顧客)님의 응용프로그램에 바로 적용할 수 있다는 것이 장점입니다.

설명드리는 CME Builder 를 통해 작성된 환경설정 파일(.CME2) 은 CMMSDK 함수인 cmmGnInitFromFile 함수를 사용해서 개발하고 있는 고객(顧客)님의 실제 소스 코드를 통해 명시적으로 파일 단위로 호출하여, 응용 프로그램에 포함시킬 수 있습니다.

I.I Main Menus

주 매뉴(Main Menus) 에서는 환경설정 파일 Load, Save 및 Save As 메뉴등을 제공하고 있습니다.



- New : 새로운 환경 설정을 시작합니다.
- Open : 기존에 존재하는 환경 설정 파일을 로드합니다. 기본 적으로 명시된 절대 경로는 윈도우 시스템 폴더[4] 입니다.
- Save : 환경 설정값이 변경된 경우에는 "Save" 버튼이 활성화 됩니다. 이 버튼을 통해 실제 변경된 환경 파일의 설정을 기록할 수 있습니다.
- Save As... : 현재 설정된 값들을 다른 파일 이름으로 저장합니다.
- Apply All : 현재 설정된 값들을 대상 모션 보드 혹은 디지털 입출력 보드에 즉시 적용하여 설정합니다.
- Code All : 현재 설정된 값들을 통해, VC++ 와 Borland C++ 호환 소스 코드를 생성합니다.

Note

Code All 기능에서 Visual Basic, Delphi 및 .NET 환경의 코드 생성 기능은 현재 버전(Version)에서 지원하지 않습니다. 추후 지원 가능시에 별도로 공지되오니, 참고하시기 바랍니다.

I.II Device Mappings

Motion 및 Digital I/O 에 대해 설치된 이용가능한 채널 맵 정보를 보여주고 이를 참고해서 새롭게 정의 할 수 있는 사용자 인터페이스를 제공합니다.

I.II.i Motion Device Map

기본적으로 아래 그림 15-2 의 하단에 보이는 "Current Installed Devices" 에서 처럼 PCI Slot 에 설치된 순서대로 축(Axis) 번호가 지정됩니다. 화면 상단에 보이는 "Current Defined Map" 에서 "Up" 및 "Down"

[4] 윈도우 시스템 폴더는 명시적인 윈도우 운영체제의 기본 폴더 하위 System32 폴더를 지칭합니다. 운영체제에 따라서 이 폴더의 경로가 변경될 수 있으며, 기본적으로 Windows NT 이나 Windows 2000 에서는 C:\WINNT\SYSTEM32 입니다. 이외 윈도우 XP 등의 운영체제는 C:\WINDOWS\SYSTEM32 경로 입니다.

버튼으로 이 순서를 변경할 수 있으며 디바이스 자체를 “Add” 및 “Delete” 버튼으로 추가 혹은 삭제 할 수 있습니다. 설치된 디바이스 맵의 디폴트 값으로 설정하려면 “Renew” 버튼을 클릭하면 됩니다.

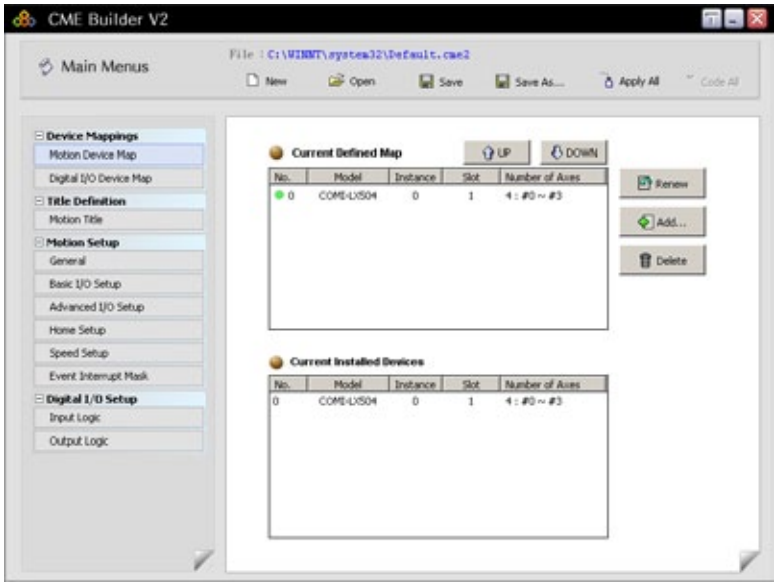


그림 15-2 Motion Device Map 설정

I.II.ii Digital I/O Device Map

Motion Device Map 과 설정 방법은 동일하며, Digital Input 과 Output 이 각각 갖는 채널 수는 Motion Device 의 2 축 당 3 채널이라고 보면 됩니다. 즉 68 핀 스카시(SCSI) 커넥터 하나당 Input 및 Output 3 채널을 각각 지원합니다. COMI-LX508 의 경우를 예를 들면, 8 축을 사용하므로 채널수는 Input 12 개 Output 12 개 입니다.

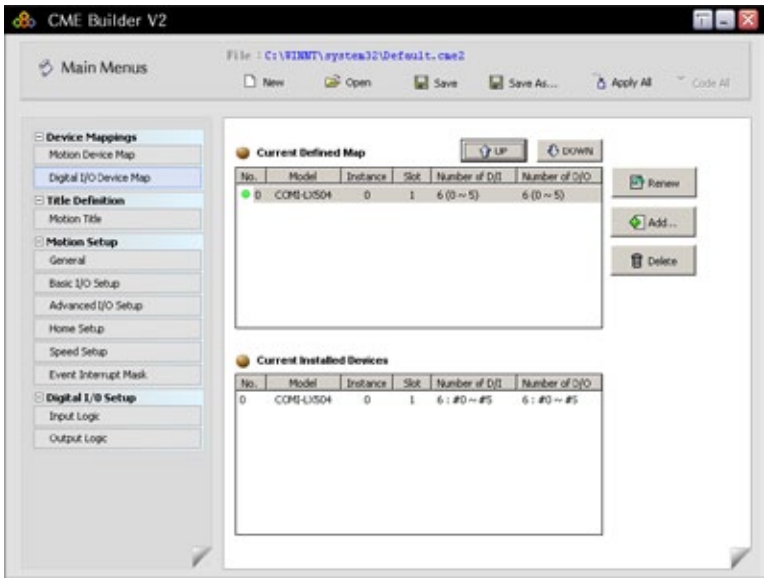


그림 15-3 Digital I/O Device Map 설정

I.III Axis Definition

I.III.i Motion Axis Title Definition

Motion 축이 가질 수 있는 별칭 이름(Alias)을 이야기 하며, 거리단위 및 속도 단위등을 문자열 값으로 설정할 수 있습니다.

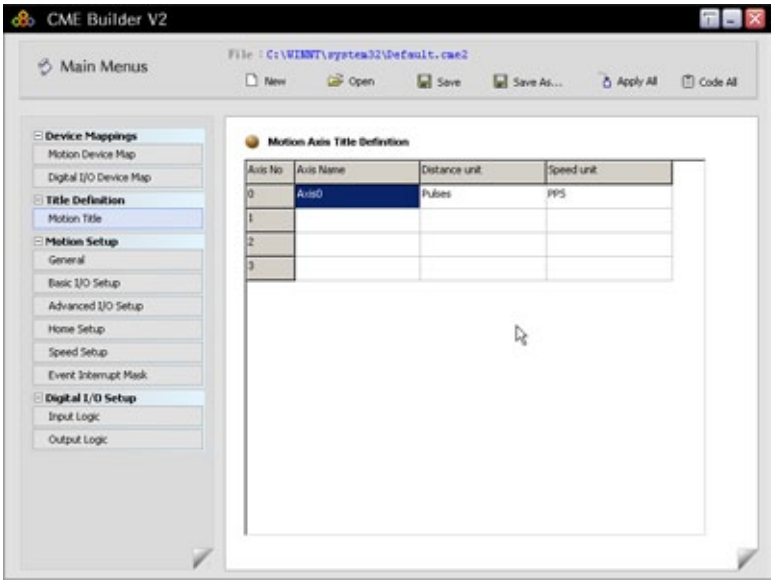


그림 15-4 Motion Axis Title Definition

I.IV Motion Setup

I.IV.i General

에러 발생시 자동으로 에러 메시지를 화면에 표시할 지에 대한 여부 및 서보드라이버로부터의 ㈜ 커미조아 모션 컨트롤러를 통해 전달되는 인코더 속도 확인(確認) 주기 설정 그리고 디버깅을 위한 레벨 및 디버그 로그 파일을 지정합니다. 디버그 로그 수준은 별도의 라이브러리 함수 설명을 참고해주시기 바랍니다. .

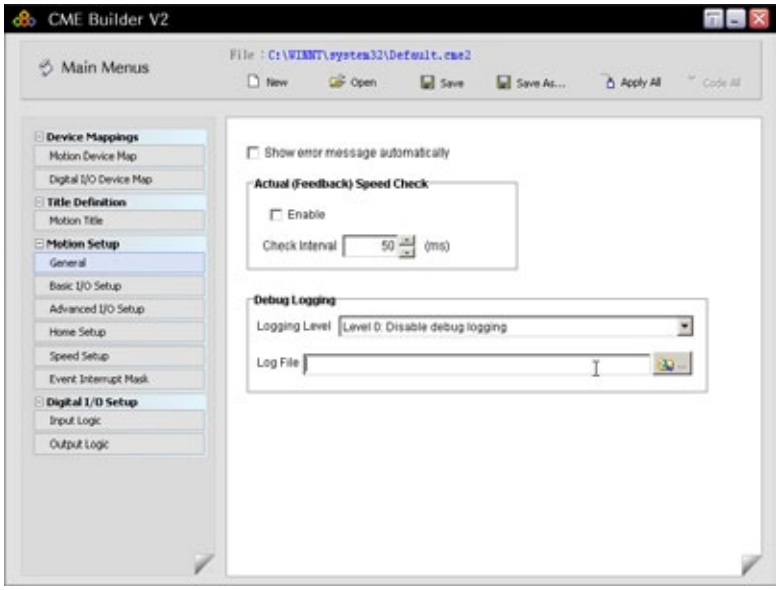


그림 15-5 General 설정

I.IV.ii Basic I/O Setup

모션 축에 대한 Command Pulse 출력 모드, Encoder Feedback 입력 모드, Unit Distance, Unit Speed 설정 및 INP, ALM, External Limit(-EL, +EL) 신호에 대한 로직을 설정합니다.

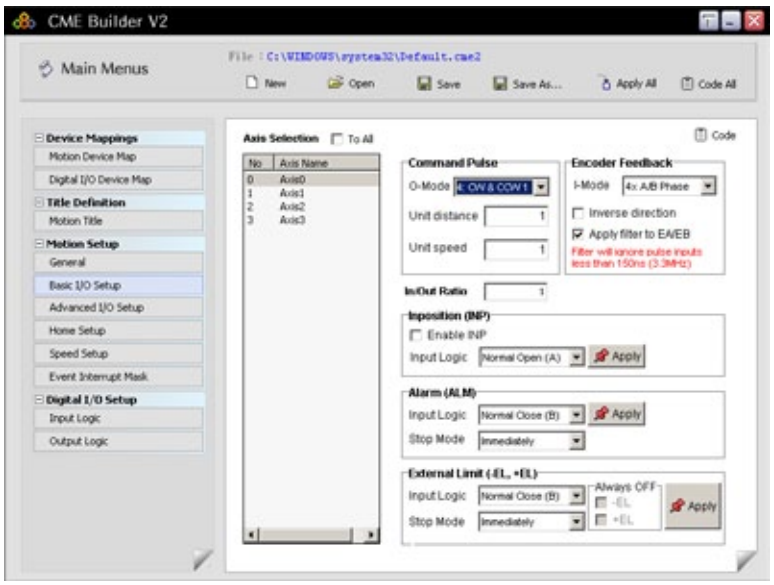


그림 15-6 Basic I/O 설정

I.IV.iii Advanced I/O Setup

위치값 래치(Position Latch) 에 대한 설정을 비롯한, 위치 비교 출력(CMP), 모션 카운터 클리어 신호(CLR) 및 서보 편차 카운터 초기화 신호(ERC), 외부 입력 신호를 통한 동기 제어(DR/SD, STA/STP)등에 대한 설정을 하실 수 있으며, 소프트웨어 동작 범위 제한(Software Limit) 에 대한 설정을 할 수 있습니다.

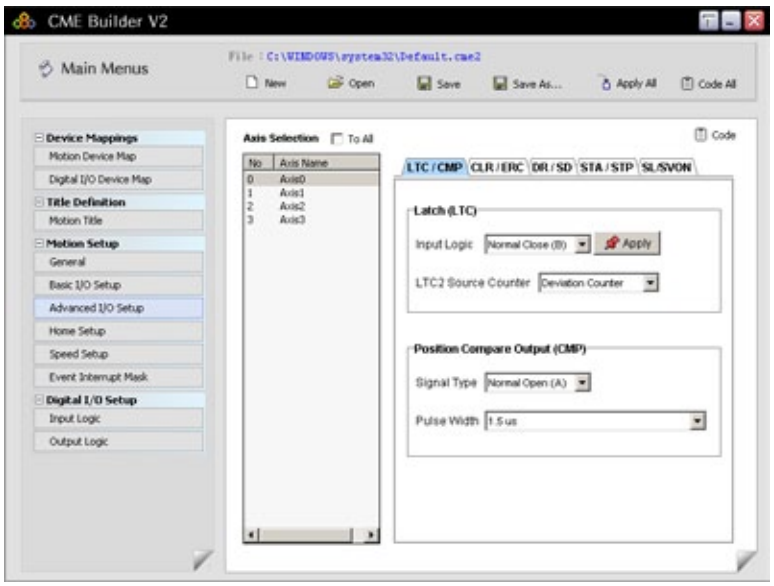


그림 15-7 Advanced I/O Setup

I.IV.iv Home Setup

원점 복귀에 대한 설정을 하는 부분으로, 원점 복귀 모드 및 원점 센서 로직, EZ 입력 로직, 탈출 거리 및 원점 복귀시 속도 패턴 및 속도 설정 등을 할 수 있습니다.

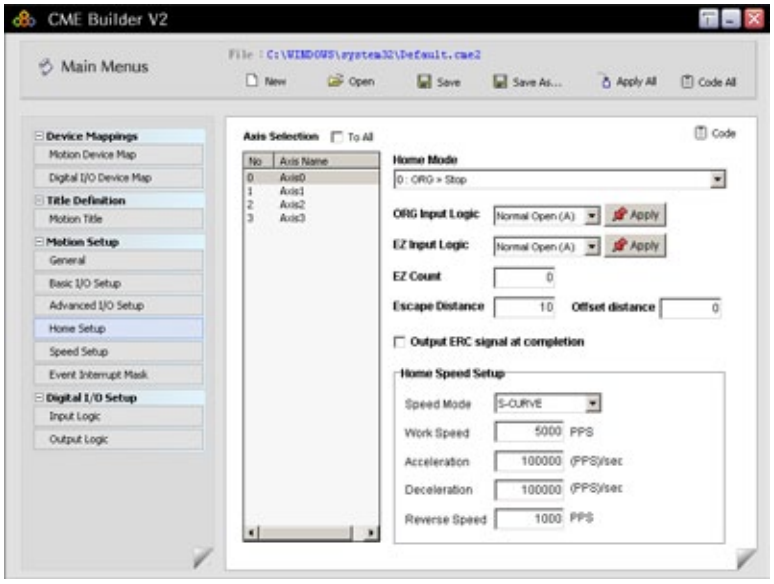


그림 15-8 원점 복귀 설정

I.IV.v Speed Setup

모션 각 축에 대한 속도 모드(Constant, Trapezoidal 및 S-Curve), 최대 속도, 초기속도, 작업 속도 및 가감속도를 설정 할 수 있습니다.

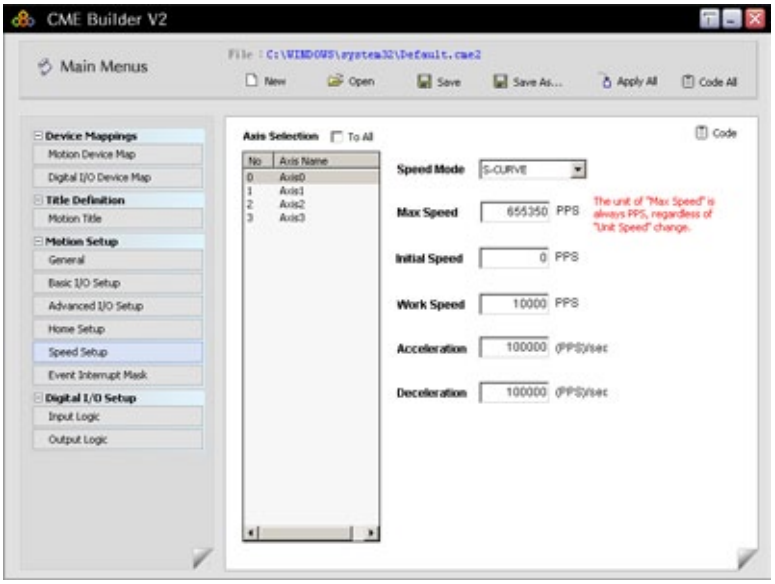

CME Builder V2
Main Menus
File : C:\WINDOWS\system32\Default.cme2
New
Open
Save
Save As...
Apply All
Code All
Device Mappings
Motion Device Map
Digital I/O Device Map
Title Definition
Motion Title
Motion Setup
General
Basic I/O Setup
Advanced I/O Setup
Home Setup
Speed Setup
Event Interrupt Mask
Digital I/O Setup
Input Logic
Output Logic
Axis Selection
To All
Code
No
Axis Name
0 Axis0
1 Axis1
2 Axis2
3 Axis3
Speed Mode
S-CURVE
Max Speed
655350
PPS
The unit of "Max Speed" is always PPS, regardless of "Unit Speed" change.
Initial Speed
0
PPS
Work Speed
10000
PPS
Acceleration
100000
(PPS)/sec
Deceleration
100000
(PPS)/sec

그림 15-9 속도 설정

I.IV.vi Event Interrupt Mask

Interrupt Event 에 대한 항목을 설정합니다. 인터럽트 이벤트는 CMMSDK 운영 중에 설정된 인터럽트의 항목에 따라 해당 인터럽트가 발생할 수 있으며, 고객(顧客) 여러분들께서는 적절한 인터럽트 핸들러(Handler)를 사용하여, 인터럽트에 대한 처리를 하실 수 있습니다.

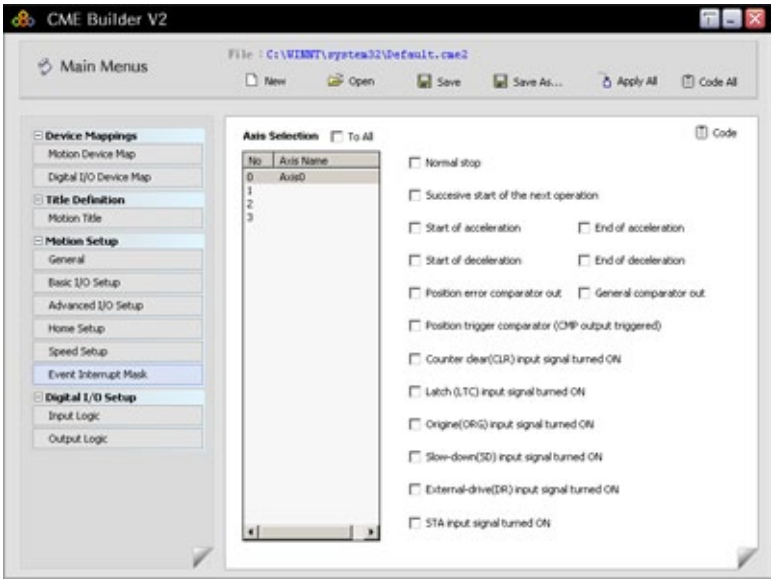


그림 15-10 Event Interrupt Mask 설정

I.V Digital I/O Setup

I.V.i Input Logic

범용 디지털 입력에 관계된 입력 논리(Logic)를 설정합니다.

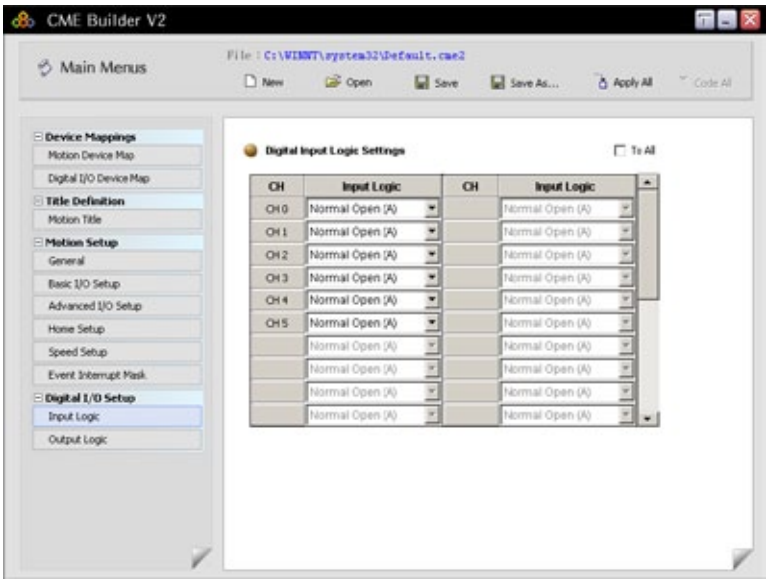


그림 15-11 Digital 입력 신호 논리(Logic) 설정

I.V.ii Output Logic

범용 디지털 출력에 관계된 출력 논리(Logic)를 설정합니다.

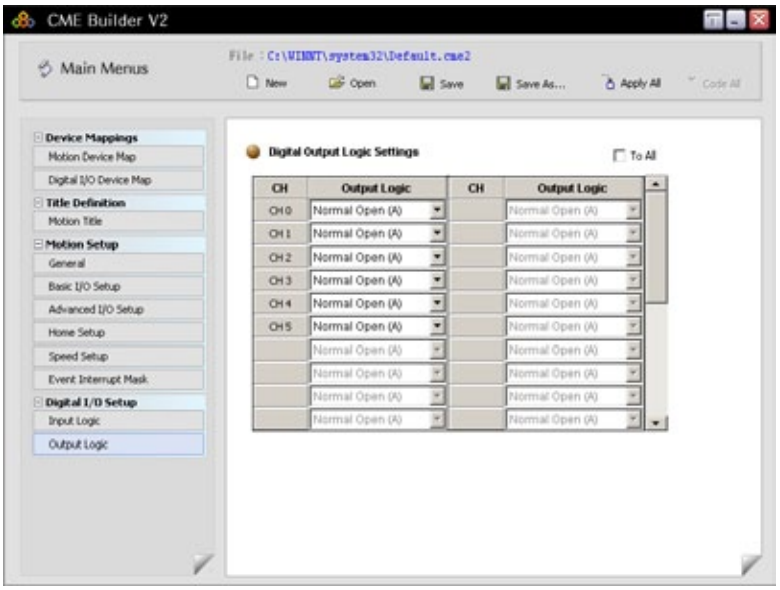


그림 15-12 Digital 출력 신호 논리(Logic) 설정

Appendix

B

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 매개 변수(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 장치 초기화는 CMMSDK 의 함수를 통해 초기화된 상태 혹은 그 값(Value)을 의미하며, 고객(顧客) 고객(顧客) 여러분들께서 해당 부분에 대해서 설정하지 않으셨거나, 초기화 설정값을 알기 위한 용도로 위한 용도로 다양하게 사용될 수 있습니다.

II 모션장치초기화시의 초기(Default) 값

II.I Command & Feedback

항목구분		기본(초기) 값		관련 함수
		값	의미	
Command	Output mode	4	CW & CCW 1	cmmCfgSetOutMode
	Unit distance	1	거리의 단위를 펄스로 사용	cmmCfgSetUnitDist
	Unit speed	1	속도의 단위를 PPS 로 사용	cmmCfgSetUnitSpeed
Feedback	Input Mode	0	1x A/B Phase	cmmCfgSetInMode
	Inverse direction	0	방향을 바꾸지 않음	cmmCfgGetInMode
In/Out Ratio		1	FeedBack / Command Ratio	cmmCfgSetInOutRatio

II.II INP, ALM , EL

항목구분		기본(초기) 값		관련 함수
		값	의미	
Inposition(INP)	Enable INP	0	INP 신호를 모션에 적용하지 않음	cmmCfgSetMioProperty (cmINP_EN)
	Input Logic	0	Normal Open(A 접점)	cmmCfgSetMioProperty (cmINP_LOGIC)
Alarm(ALM)	Input Logic	0	Normal Open(A 접점)	cmmCfgSetMioProperty (cmALM_LOGIC)
	Stop Mode	0	즉시 정지(停止)	cmmCfgSetMioProperty (cmALM_MODE)
External Limit (-EL, +EL)	Input Logic	0	Normal Open(A 점점)	cmmCfgSetMioProperty (cmEL_LOGIC)
	Stop Mode	0	즉시 정지(停止)	cmmCfgSetMioProperty (cmEL_MODE)

II.III LTC, CMP, CLR, ERC

항목구분		기본(초기) 값		관련 함수
		값	의미	
Latch (LTC)	Input Logic	0	Normal Open(A 접점)	cmmCfgSetMioProperty (cmLTC_LOGIC))
	LTC2 Source	0	Deviation Counter	cmmCfgSetMioProperty (cmLTC_LTC2SRC)
Position Compare Output (CMP)	Signal Type	0	Normal Open(A 점점)	cmmCfgSetMioProperty (cmCMP_LOGIC))
	Pulse Width	0	same width as command pulse	cmmCfgSetMioProperty (cmCMP_PWIDTH)
Clear Counter Input (CLR)	Input Signal Type	0	Falling edge	cmmCfgSetMioProperty (cmCLR_SIGTYPE)
	Target counters to clear	0	No counter selected (CLR 신호 사용안함)	cmmCfgSetMioProperty (cmCLR_CNTR)
ERC Output	Output Logic	0	Normal Open(A 점점)	cmmCfgSetMioProperty (cmERC_LOGIC)
	On-time	6	104ms	cmmCfgSetMioProperty (cmERC_OUT)

II.IV DR, SD, STA, STP

항목구분		기본(초기) 값		관련 함수
		값	의미	
DR Input Signal Logic		0	Normal Open(A 접점)	cmmCfgSetMioProperty(cmDR_LOGIC)
Slow down (SD) Input	Enable SD	0	사용안함	cmmCfgSetMioProperty(cmSD_EN)
	Input logic	0	Normal Open(A 접점)	cmmCfgSetMioProperty(cmSD_LOGIC)
	SD Mode	0	SD 가 입력되면 초기속도까지 감속 한후 초기속도로 이송	cmmCfgSetMioProperty(cmSD_MODE)
	SD Latch	0	SD 신호를 래치하지 않는다.	cmmCfgSetMioProperty(cmSD_LATCH)
Start (STA) Input	Mode	0	Software 적으로 이송명령이 내려지면 즉시 이송 시작(Hardware STA 신호 사용안함)	cmmCfgSetMioProperty(cmSTA_MODE)
	Signal Type	0	Level(low active) Input	cmmCfgSetMioProperty(cmSTA_TRG)
Stop (STP) Input Mode		0	Hardware 적인 STP 신호 사용안함	cmmCfgSetMioProperty(cmSTP_MODE)

II.V Software Limit

항목구분		기본(초기) 값		관련 함수
		값	의미	
Software Limit	Enable	0	소프트웨어 Limit 해제	cmmCfgSetSoftLimit
	N-Limit value	-99999999	음의 방향의 최대값	
	P-Limit value	99999999	양의 방향의 최대값	

II.VI Servo ON Input Logic

항목구분	기본(초기) 값		관련 함수
	값	의미	
Servo ON Input Logic	0	Normal open (A 접점)	cmmCfgSetMioProperty (cmSVON_LOGIC)

II.VII 원점복귀 환경

항목구분	기본(초기) 값		관련 함수
	값	의미	
Home mode	0	원점복귀 모드 0 번	cmmHomeSetConfig
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	cmmCfgSetMioProperty (cmERC_OUT)
Speed Mode	2	S-CURVE 모드	cmmHomeSetSpeedPattern
Work Speed	5000	원점복귀 정속도	
Acceleration	100000	원점복귀 가속도	
Deceleration	100000	원점복귀 감속도	

Reverse Speed	1000	원점복귀 역방향 이송 속도	

II.VIII 모션 정격 속도 환경

항목구분	기본(초기) 값		관련 함수
	값	의미	
Max Speed	655350	655350 (PPS)	cmmCfgSetSpeedRange
Initial Speed	0	초기 속도	cmmSxOptSetIniSpeed
Speed Mode	2	S-CURVE 모드	cmmCfgSetSpeedPattern
Work Speed	10000	정속도	
Acceleration	100000	가속도	
Deceleration	100000	감속도	

II.IX Event Interrupt

항목구분	기본(초기) 값		관련 함수
	값	의미	
Event Interrupt	0	인터럽트 이벤트 대상에 대한 초기 값	cmmIntSetMask

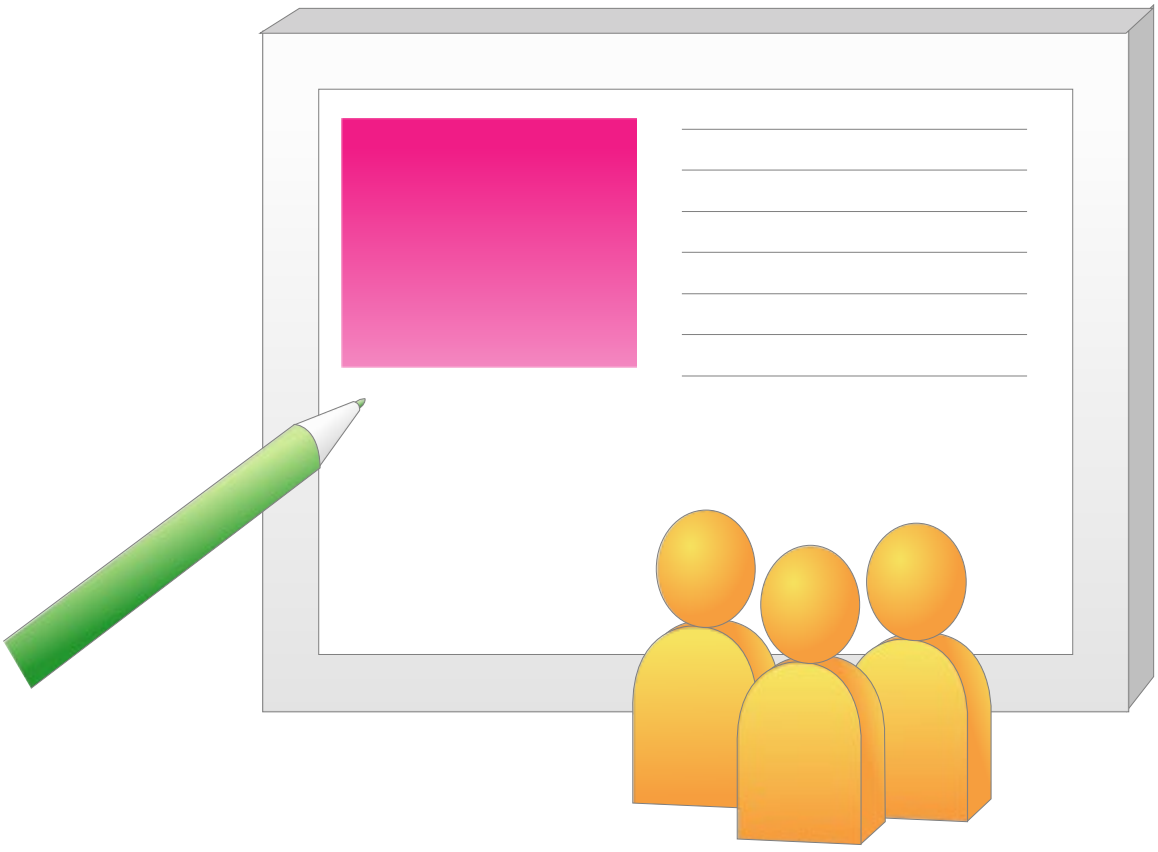
Appendix

C

List of Error Codes

에러코드는 모션 응용프로그램 뿐만 아니라 , 모든 응용프로그램에서 필요한 주요 정보 중 하나입니다. 커미조아의 CMMSDK 에서는 자세하고, 일괄적인 에러코드를 제공함으로써, 모션 에러 상태에 대한 명확한 원인과 분석이 가능할 수 있도록 돕고 있습니다. 본장에서 열거한 에러코드는 다양한 모션 에러 코드 상황에서, CMMSDK 를 통해 모션 에러 상태를 효율적으로 자세하고 명확히 판단할 수 있는 근거를 제공합니다.

통합 라이브러리에서 제공하는 에러코드 리스트를 일람표로 정리하여 수록하였으며, 명시된 에러코드를 에러코드를 통해 효과적으로 에러처리를 하는 방법에 대해서 안내하고 있습니다. 일반적으로 CMMSDK 를 일반적으로 CMMSDK 를 사용하다가 에러가 발생하면 cmmGnGetErrorCode() 함수를 사용하여 에러코드를 사용하여 에러코드를 참조할 수 있습니다. 이때 참조되는 에러코드에 대한 의미를 에러코드 일람표를 일람표를 참조하여 파악하시기 바랍니다.



II.X 에러 코드 일람표

ERROR CODE	VALUE	MEANING
cmERR_NONE	0	No error
cmERR_MEM_ALLOC_FAIL	-290	Memory allocation fail
cmERR_GLOBAL_MEM_FAIL	-292	Global memory allocation fail
cmERR_ISR_CONNEC_FAIL	-310	ISR(Interrupt Service Routine) registration fail
cmERR_DIVIDE_BY_ZERO	-400	Cause divide by zero error
cmERR_WORNG_NUM_DATA	-500	Number of data is too small or too big
cmERR_VER_MISMATCH	-600	Version(of file or device) mismatch
cmERR_INVALID_DEVICE_ID	-1010	User set invalid device id. Refer to "DeviceId" property.
cmERR_INVALID_HANDLE	-1020	Device handle is not valid. This means that loading a device has been failed or not performed. Refer to "GnLoadDevcie" function.
cmERR_UNSUPORTED_FUNC	-1030	User called an unsupported function for the specified product.
cmERR_INVALID_PARAMETER	-1101	Some of the function parameters are invalid.
cmERR_INVALID_CHANNEL	-1105	The channel setting parameter(s) is(are) invalid
cmERR_INVALID_INPUT_RANGE	-1111	Invalid range value (AI, AO)
cmERR_INVALID_FREQ_RANGE	-1121	User selected invalid frequency range
cmERR_FILE_CREATE_FAIL	-1501	File creation has been failed
cmERR_FILE_OPEN_FAIL	-1511	File opening has been failed
cmERR_FILE_READ_FAIL	-1522	File reading fail
cmERR_EVENT_CREATE_FAIL	-1550	Event handle creation has been failed
cmERR_INT_INSTANCE_FAIL	-1560	Interrupt event instance creation has been failed
cmERR_DITHREAD_CRE	-1570	D/I state change monitor thread creation fail
cmERR_BUFFER_SMALL	-1580	Buffer size is too small
cmERR_HIGH_TIMER_UNSUPP	-1590	The installed hardware doesn't support a high-resolution performance counter(when cmmUtlDelayMicroSec() had failed)
cmERR_OUT_OF_RANGE	-1600	The range of some parameter is out of range that it is occured
cmERR_ON_MOTION	-5001	This code is just a symbolic code. This error will never occur.
cmERR_STOP_BY_SLP	-5002	Abnormally stopped by positive soft limit
cmERR_STOP_BY_SLN	-5003	Abnormally stopped by negative soft limit
cmERR_STOP_BY_CMP3	-5004	Abnormally stopped by comparator3
cmERR_STOP_BY_CMP4	-5005	Abnormally stopped by comparator4
cmERR_STOP_BY_CMP5	-5006	Abnormally stopped by comparator5
cmERR_STOP_BY_ELP	-5007	Abnormally stopped by (+) external limit
cmERR_STOP_BY_ELN	-5008	Abnormally stopped by (-) external limit
cmERR_STOP_BY_ALM	-5009	Abnormally stopped by alarm input signal
cmERR_STOP_BY_CSTP	-5010	Abnormally stopped by CSTP input signal
cmERR_STOP_BY_CEMG	-5011	Abnormally stopped by CEMG input signal
cmERR_STOP_BY_SD	-5012	Abnormally stopped by SD input signal
cmERR_STOP_BY_DERROR	-5013	Abnormally stopped by operation data error
cmERR_STOP_BY_IP	-5014	Abnormally stopped by other axis error during interpolation
cmERR_STOP_BY_PO	-5015	An overflow occurred in the PA/PB input buffer
cmERR_STOP_BY_AO	-5016	Out of range position counter during interpolation
cmERR_STOP_BY_EE	-5017	An EA/EB input error occurred (does not stop)
cmERR_STOP_BY_PE	-5018	An PA/PB input error occurred (does not stop)
cmERR_STOP_BY_SLVERR	-5019	Abnormally stopped because slave axis has been stopped (Only in Master/Slave mode)
cmERR_STOP_BY_SEMG	-5020	Abnormally stopped by "software emergency" setting
cmERR_MOT_MAOMODE	-5110	Master output mode is not CW/CCW mode during Master/Slave operation
cmERR_MOT_SLAVE_SET	-5120	Slave start has been failed during Master/Slave operation
cmERR_SPEED_RANGE_OVER	-5130	Speed setting value exceeds setting range
cmERR_INVALID_SPEED_SET	-5140	Speed setting value is not valid

ERROR CODE	VALUE	MEANING
cmERR_INVALID_IXMAP	-5150	Invalid Interpolation Map
cmERR_INVALID_LMMAP	-5160	Invalid List-Motion Map
cmERR_MOT_SEQ_SKIPPED	-5170	Motion command has been skipped because the axis is already running
cmERR_UNKNOWN	-9999	Unknown error.

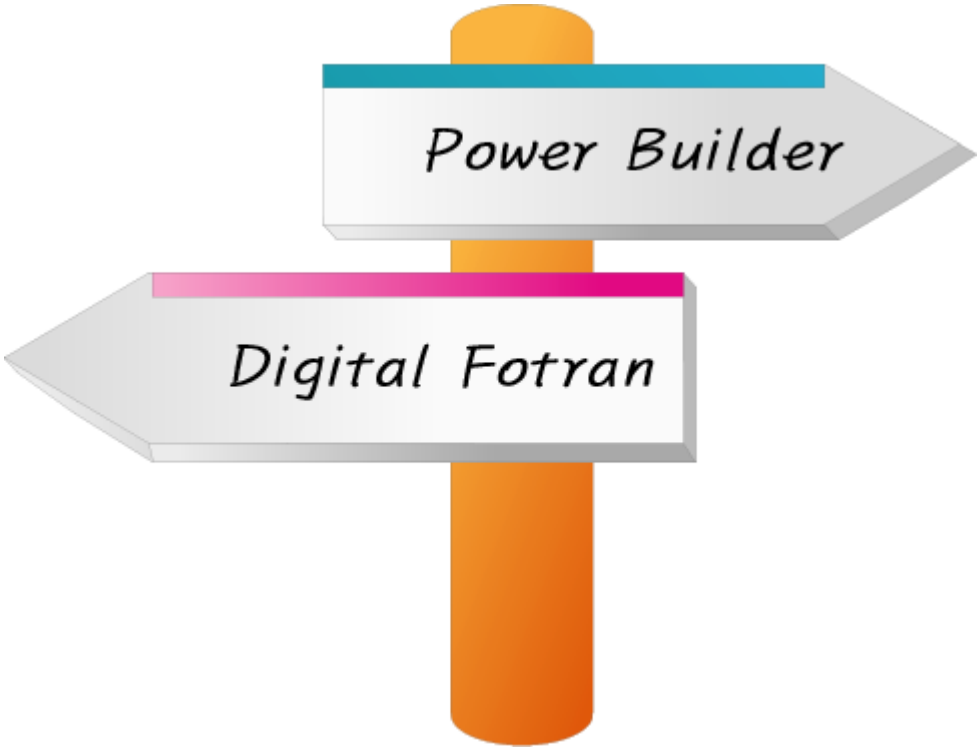
Appendix

D

Other Development Environment Support

CMMSDK 는 다른 개발 환경을 사용하는 고객(顧客)여러분들께도 지원을 아끼지 않습니다. 이미 지원 가능한 닷넷 환경을 포함하여, 파워 빌더 및 디지털 포트란등의 다양한 개발환경을 만족 시켜 드리면서, 명실 상부한 모션 라이브러리의 자부심을 지키고 있습니다.

다른 개발 환경 Power Builder 및 Digital Visual Fortran 에서 CMMSDK 를 사용하는 방법에 대해서 안내하고 안내하고 있습니다. ㈜ 커미조아 모션 라이브러리는 다양한 기능과 강력한 윈도우 응용프로그램 개발을 응용프로그램 개발을 위해 출시된 제품으로서, 고객님들께서 익숙한 환경에서 개발을 진행 할 수 있도록 할 수 있도록 최선을 다하여 지원하고 있습니다.



III 다른 개발환경 고객(顧客)님들을 위한 라이브러리 안내

III.I Digital Fortran 사용자를 위한 CMMSDK 안내

Digital Fortran 사용자는 아래와 같이 CMMSDK를 호출 사용하실 수 있습니다.

```
! Sample Fortran program that finds a dll library and calls
!  a routine in it.

      use CmmSDK

! First, locate the dll and load it into memory
    p = loadlibrary("Cmmsdk.dll"C)
     if (p == 0) then
          type *, "Error occurred opening  CmmSDK. DLL "
          type *, "Program aborting"
          goto 1000
     endif
```

III.II PowerBuilder 사용자를 위한 CMMSDK 사용안내

PowerBuilder 사용자는 'http://www.sybase.com' PowerBuilder 배포 社를 통해 CMMSDK 사용에 근간이 되는 DLL 라이브러리 인터페이스 방법을 배우실 수 있습니다. 이에 대한 자세한 기술 지원이 필요하시면 저희 ㈜커미조아 고객(顧客) 지원팀으로 연락주시기 바랍니다.

III.III C#(C Sharp), Visual Basic.NET 및 Visual C++.NET 개발자를 위한 안내

본 CMMSDK라이브러리는 Microsoft.NET 개발 환경에서도 아무런 제약없이 사용 할 수 있습니다. 다만, Visual C++.NET (Managed C++ 인 경우), VB.NET 및 C# 에서 CMMSDK 라이브러리를 사용하려면 다음의 절차를 따라야 합니다.

가) Cmmsdk.h 파일에 정의된 라이브러리 함수 원형에 대해 해당 언어에 맞게 데이터 타입을 마샬링(Marshalling)하여 새로운 함수로 정의 합니다. 이때 명시적으로 Cmmsdk.dll 라이브러리의 각 함수의 Entry Point 및 함수 호출 방법 등을 지정하기 위해 [import] 구문을 사용 합니다.

나) 마샬링(Marshalling)을 통해 재정의된 함수를 사용하기 위해서는 Cmmsdk.dll 파일이 윈도우즈의 실행 경로(예: 윈도우즈 시스템 폴더) 또는 실행하려는 프로그램과 동일한 폴더에 위치하면 됩니다. 일반적으로 COMI-AUTOMATION v2 를 설치하게 되면, 윈도우 시스템 디렉토리 혹은 그에 상응하는 디렉토리에 라이브러리가 설치되기 때문에 이 부분은 자동으로 진행됩니다.

다) 필요에 따라 CmmsdkDef.h 에 정의된 각종 Define 된 상수를 해당 언어에 맞게 재 정의 합니다.

본 매뉴얼에서는 C# 개발자를 위해 라이브러리 함수를 재정의 하는 방법과 재정의된 함수를 사용하는 방법을 아래와 같이 제공합니다. 개발자는 아래의 내용을 참고해서 별도의 소스 파일을 작성한 뒤 프로젝트에 포함시켜 사용하시면 됩니다. 프로젝트를 생성하거나 소스파일을 추가시키는 방법에 대해서는 해당 개발환경에서의 도움말을 참고하시기 바랍니다.

Visual Basic.NET 및 Visual C++.NET (Managed C++ 인 경우) 개발 환경에서는 C# 에서의 라이브러리 사용방법과 유사한 방법으로 마샬링 함수를 정의하셔서 사용하시면 됩니다.

C# 에서 CMMSDK 라이브러리 함수를 마샬링(Marshalling) 하는 코드의 내용일부를 아래와 같이 표시하였습니다.

```
using System;
using System.Runtime.InteropServices;
namespace YourProject.ImportDLLs
{
    public unsafe class CMDLL {
        //====================== General FUNCTIONS ==================================================//
        // 1. cmmGnDeviceLoad
        [DllImport("Cmmsdk.dll", EntryPoint = "cmmGnDeviceLoad", ExactSpelling = true,
            CharSet = CharSet.Ansi, CallingConvention = CallingConvention.Cdecl)]
        public static extern unsafe int cmmGnDeviceLoad([MarshalAs(UnmanagedType.I4)] int IsResetDevice,
            [MarshalAs(UnmanagedType.LPArray, ArraySubType = UnmanagedType.I4)] int[] NumAxes);

        // 2. cmmGnDeviceUnload
        [DllImport("Cmmsdk.dll", EntryPoint = "cmmGnDeviceUnload", ExactSpelling = true, CharSet = CharSet.A
        public static extern unsafe int cmmGnDeviceUnload( );

        // 3. cmmGnSetServoOn
        [DllImport("Cmmsdk.dll", EntryPoint = "cmmGnSetServoOn", ExactSpelling = true, CharSet = CharSet.Ans
        public static extern unsafe int cmmGnSetServoOn([MarshalAs(UnmanagedType.I4)] int Channel,
            [MarshalAs(UnmanagedType.I4)] int Enable);

        // 4. cmmGnGetServoOn
        [DllImport("Cmmsdk.dll", EntryPoint = "cmmGnGetServoOn", ExactSpelling = true, CharSet = CharSet.Ans
        public static extern unsafe int cmmGnGetServoOn([MarshalAs(UnmanagedType.I4)] int Channel,
            [MarshalAs(UnmanagedType.LPArray, ArraySubType = UnmanagedType.I4)] int[] Enable);

        //====================== Single Axis Move FUNCTIONS ========================================//
        // 1. cmmSxMove
        [DllImport("Cmmsdk.dll", EntryPoint = "cmmSxMove", ExactSpelling = true, CharSet = CharSet.Ansi, Cal
        public static extern unsafe int cmmSxMove([MarshalAs(UnmanagedType.I4)] int Channel,
            [MarshalAs(UnmanagedType.R8)] double Distance, [MarshalAs(UnmanagedType.I4)] int IsBlocking);

        // 2. cmmSxStop
        [DllImport("Cmmsdk.dll", EntryPoint = "cmmSxStop", ExactSpelling = true, CharSet = CharSet.Ansi, Cal
        public static extern unsafe int cmmSxStop([MarshalAs(UnmanagedType.I4)] int Channel,
            [MarshalAs(UnmanagedType.I4)] int IsWaitComplete, [MarshalAs(UnmanagedType.I4)] int IsBlocking);

        // (생략...)
    }
}
```

그림 15-13 라이브러리 함수 재정의 예

위 “그림 15-13” 와 같이, unsafe 형태의 CMDLL 클래스는 다른 소스파일에서 자유롭게 호출이 가능한 정적함수(static function) 들의 집합으로 구성되어 있습니다.

먼저 DllImport 구문을 사용하기 위해서는 ‘using System.Runtime.InteropServices;’를 추가시켜줍니다.

DllImport 구문의 매개 변수(媒介變數)중에 위 그림에서 붉은 색 사각형 으로 표시한 것들은 순서대로 호출하려는 DLL 라이브러리 이름(Cmmsdk.dll), 라이브러리내의 실제 함수 이름(EntryPoint = “cmmGnDeviceLoad”) 그리고 함수 호출 방법(CallingConvension = CallingConvension.Cdecl) 입니다.

DllImport 구문 아래 부분은 프로그램에서 실제로 사용하기 위해 데이터 타입을 마샬링해서 재정의한 함수 선언부 이며, 위에서 언급했듯이 다른 소스파일에서 정적 호출이 가능하도록 static 으로 선언되었고 DLL 함수내부에서 포인터 사용을 하는 등의 이유로 안전하지 못한 코드를 실행하는 경우 보호받기 위해 unsafe 라는 키워드를 사용했습니다.

라이브러리의 4 바이트 정수값을 받기 위해 Return Type 으로 ‘int’ 를 사용하였으나 함수의 매개 변수(媒介變數)에서는 4 바이트 정수값을 전달 하기 위해 ‘int’타입 앞에 4 바이트 정수형 Unmanaged Type 임(UnmanagedType.I4) 을 명시해줘야 합니다. Floating Point 형태의 변수인 경우는 ‘UnmanagedType.I4’대신 ‘UnmanagedType.R8’ 을 사용하며 다음에 오는‘int’ 대신 ‘double’ 로 선언하면 됩니다.

라이브러리에서 포인터 변수를 마샬링 하는 방법은 마샬링 타입을 ‘LPArray’ 로 선언하고 그 서브타입을 정수형 및 실수형에 맞게 선언해주시면 됩니다. 그리고 데이터 타입은 ‘int’ 또는 ‘double’ 대신 ‘int[]’ 및 ‘double[]’ 을 각각 사용하시면 됩니다. 자세한 예제는 “그림 15-13”을 참고하시면 편리합니다.

다음은 위에서 재정의된 함수를 다른 소스에서 사용하는 예를 부분적으로 표시한 내용 입니다.

```
using System;
```

```
using System.Diagnostics;
using System.Windows.Forms;
using YourProject.ImportDLLs;

namespace YourProject
{
   public partial class MainForm : Form
   {

      enum _TCmBool { cmFALSE, cmTRUE };
      enum DIR { MODE_CW, MODE_CCW };
      const int cmERR_NONE = 0;

      // 생성자 함수
      public MainForm()
      {
         // 1. Motion Device Loading
         int[] NumAxes = new int[1];
         if (CMDLL.cmmGnDeviceLoad((int)_TCmBool.cmTRUE, NumAxes) != cmERR_NONE)
         {
            CMDLL.cmmErrShowLast(this.Handle);
            return;
         }
         Debug.WriteLine("로드된 축의 개수는 : " + NumAxes[0].ToString() + " 개 입니다");

      }

      // 단축 구동
      public void SxMove()
      {
         CMDLL.cmmSxMove(0, (int)DIR.MODE_CW, (int)_TCmBool.cmFALSE);
      }

      private void MainForm_FormClosing(object sender, FormClosingEventArgs e)
      {
            if(CMDLL.cmmGnDeviceUnload() != cmERR_NONE)
            {
                  Debug.WriteLine("cmmGnDeviceUnload() 에 실패하였습니다");
            }
      }
                  // (생략..)
   }
}
```

표 12 재정의 된 함수를 사용하는 코드의 C# 코드의 일부

Appendix

E

Frequently Asked Questions

자주 발생하는 질문이나, 고객(顧客) 기술 지원에 대한 내용은 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 CMMSDK 사용에 있어, 문의 사항이나 궁금하신 점을 FAQ 로 안내하였습니다. 안내하였습니다. 각 FAQ 는 ㈜ 커미조아의 웹 사이트(http://www.comizoa.com) 이나 별도로 등록된 등록된 고객(顧客)님의 정보를 통해 전달 될 수 있도록 노력하겠으며, 본 FAQ 장을 고객(顧客) 문의를 고객(顧客) 문의를 하시기전에 미리 확인(確認)하시어, 좋은 참고가 되시기를 간절히 바라겠습니다.
바라겠습니다.

IV Frequently Asked Questions (FAQ)

IV.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 CMMSDK 는 표준 ANSI Libraries API 함수를 사용하는데, 현재 프로젝트의 설정이 Unicode 로 되어 있어 다음과 같은 에러가 발생할 수 있습니다.

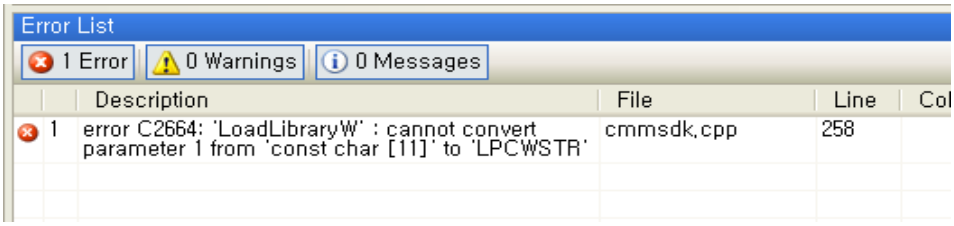


그림 15-14 Unicode 로 되어 있을 경우 발생하는 에러 화면

A

해당 문제를 해결 하는 방법은 다음과 같습니다. MS VC++ 의 Solution Explorer 에서 현재 작업중인 프로젝트를 선택합니다.

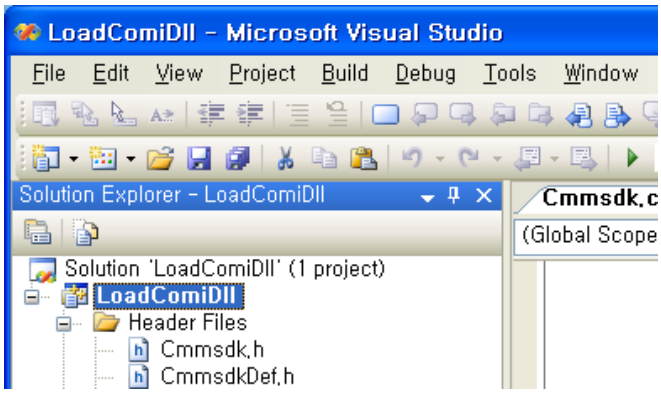


그림 15-15 사용자 생성 프로젝트 선택 화면

메뉴에서 [Project]->[Properties]를 선택하여 [Property Page]창을 엽니다.

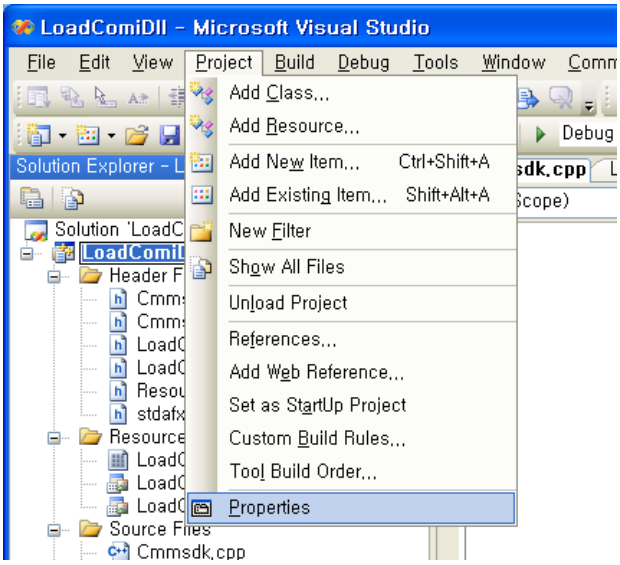


그림 15-16 사용자 생성 프로젝트의 등록정보 선택 화면

[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.

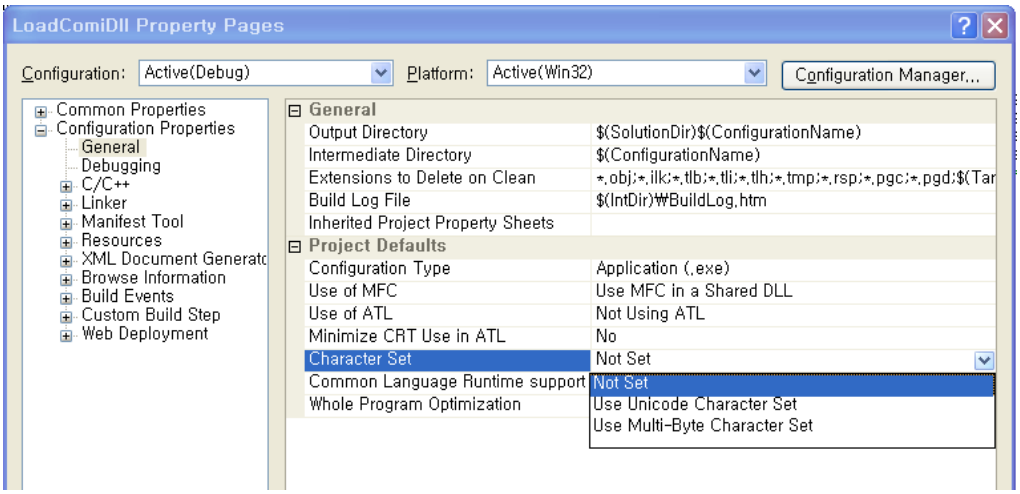


그림 15-17 문자 집합을 단일 바이트 타입으로 변경하는 화면

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.

IV.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.

IV.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 OCX를 복사한 후 regsvr32 명령으로 OCX를 등록시킵니다.
   (시작->실행 메뉴에서 : regsvr32 c:\windows\system32\ComiSliderCtrl.ocx)

2. ComiSliderCtrl.ocx를 [..\CBuilder5\Bin\] 폴더로 복사합니다.
tlibimp.exe 프로그램을 이용하여 Import Type Library File을 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.

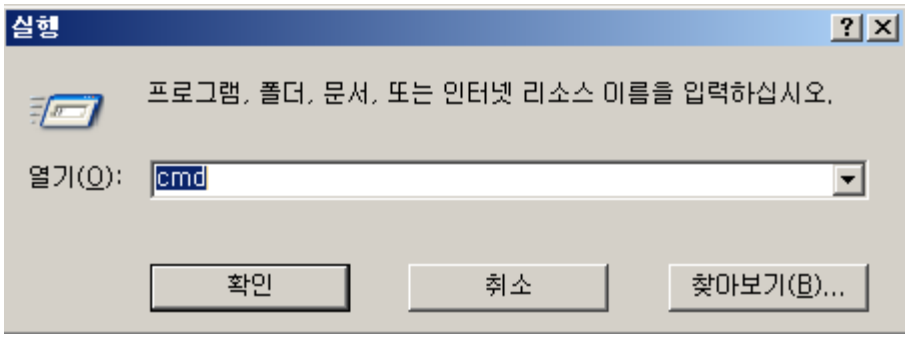


그림 15-18 [작업표시줄]-[시작]-[실행] 창에서 cmd 명령어 입력

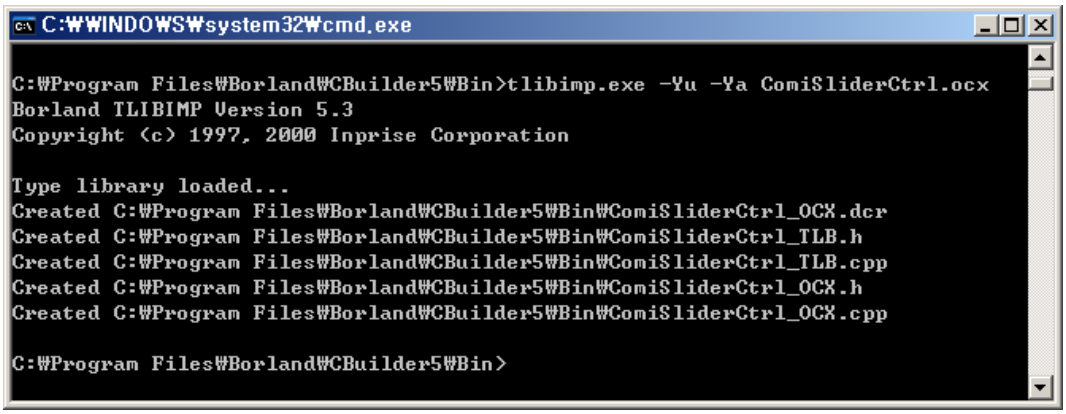


그림 15-19 Tlibimp.exe 파일을 이용 Import Type Library File 생성

4. Builder 5 에서 OCX를 등록합니다.

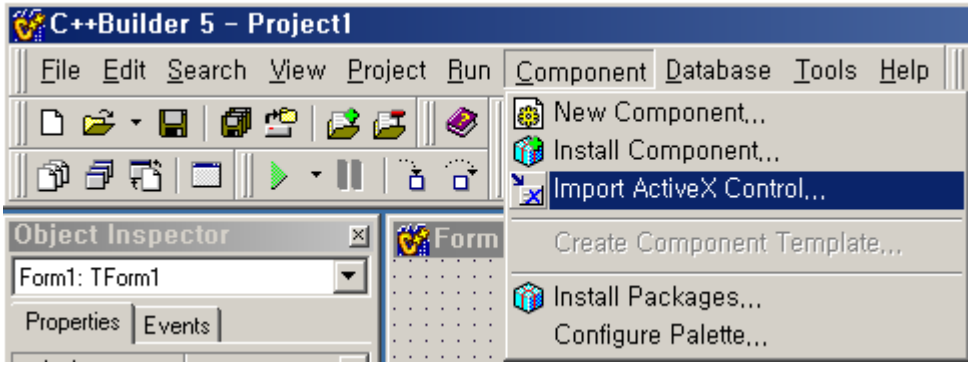


그림 15-20 C++ Builder 5 의 ActiveX Component 등록

5. Import ActiveX 윈도우가 화면에 나타나면, ComiSliderCtrl을 선택하고 원하는 Palette Page를 선택한뒤 Install 버튼을 눌려 설치를 시작합니다.

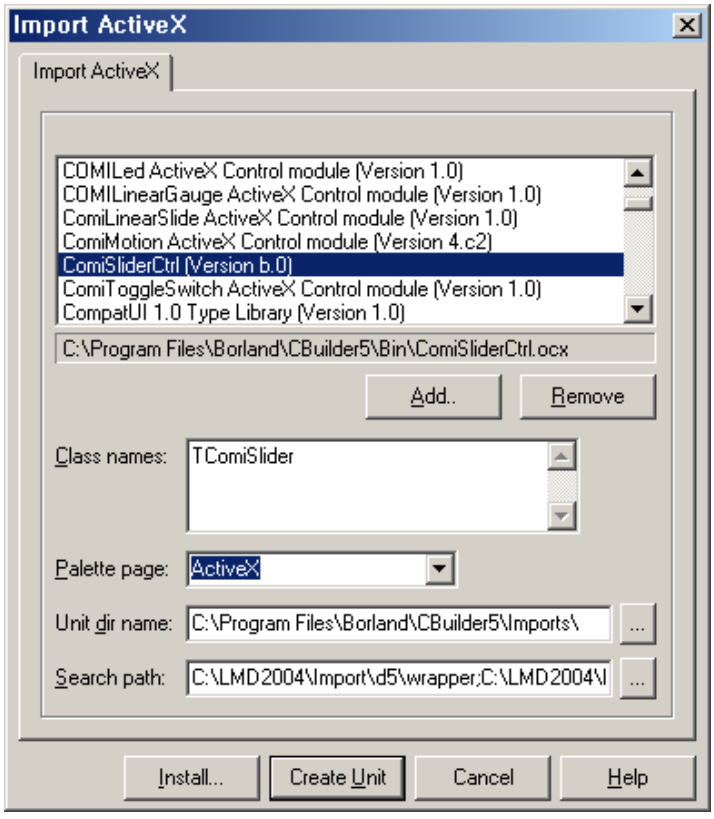


그림 15-21 C++ Builder 5 의 ActiveX Component 를 등록하기 위한 Import Active X 화면

6. Install창이 화면에 나타나면 Into new package 탭에서 생성을 원하는 Package 파일 이름 및 설명을 입력하여 새로운 bpk를 생성합니다.

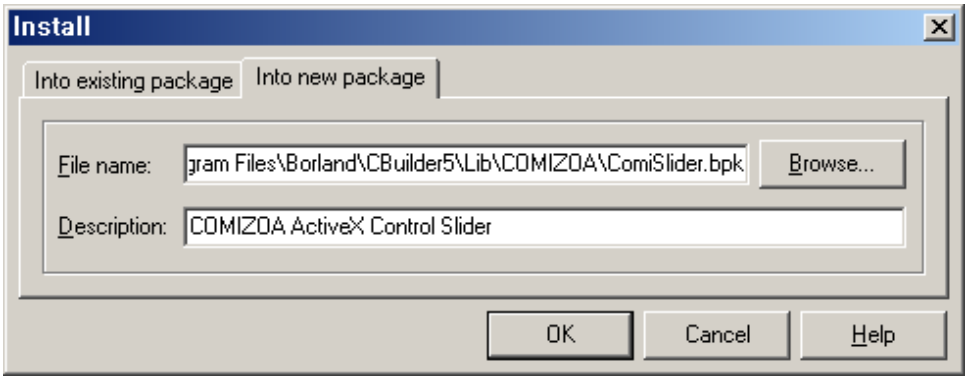


그림 15-22 C++ Builder 5 의 ActiveX Component 설치 화면

7. Package를 Install한다는 확인(確認) 창이 화면에 표시되면, "No" 를 선택합니다.

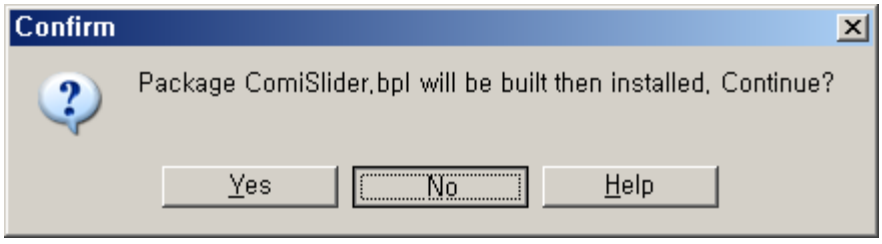


그림 15-23 C++ Builder 5 에서 ActiveX Component 설치 화면

8. Tlibimp.exe파일을 이용해 생성한 5 개의 파일을 ..\Borland\CBuilder5\Imports 에 덮어쓰기(Overwrite) 합니다

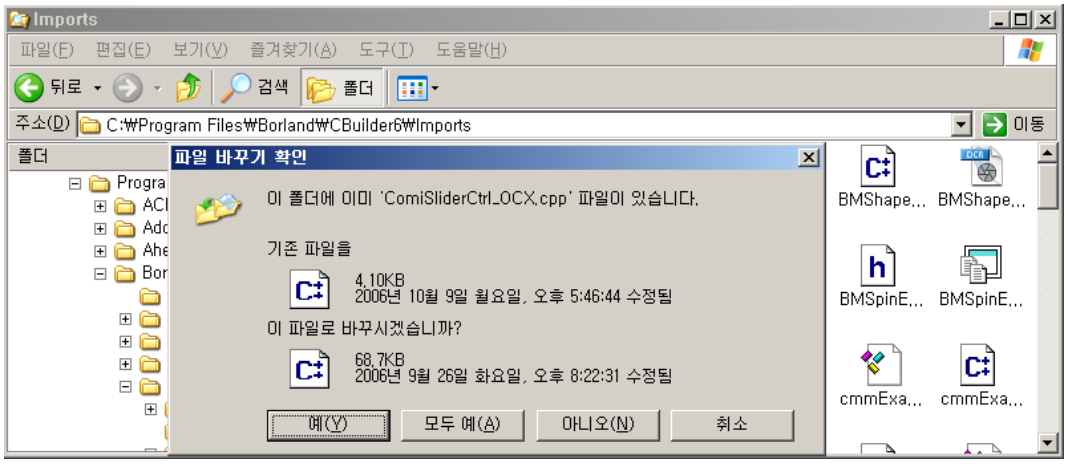


그림 15-24 Tlibimp.exe 가 생성한 파일을 통해 기존 파일을 대체하는 작업 화면

9. 덮어쓰기가 끝나면 생성한 Package 파일을 Install합니다.

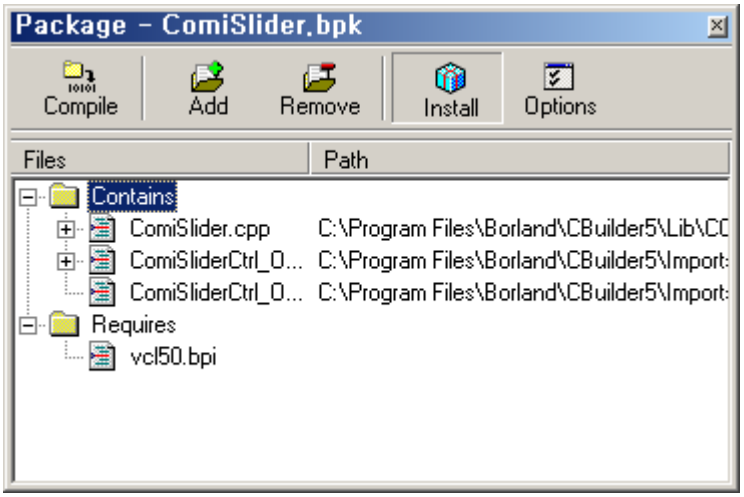


그림 15-25 C++ Builder 5 의 Package 표시 화면

10. Package가 Install되었다는 메시지와 함께 툴 팔래트에 OCX가 정상적으로 등록된 것을 확인(確認)할 수 있습니다.

그림 15-26 C++ Builder 5 의 패키지 설치 완료 화면

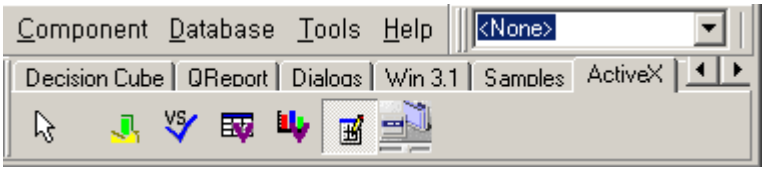


그림 15-27 C++ Builder 5 에서 팔레트(Pallette) 에 등록된 ComiSlider Active X Control

* Builder 6 에서 ComiSliderCtrl.ocx 사용방법.

11. [Tool]-[Environment Options..]를 선택합니다..

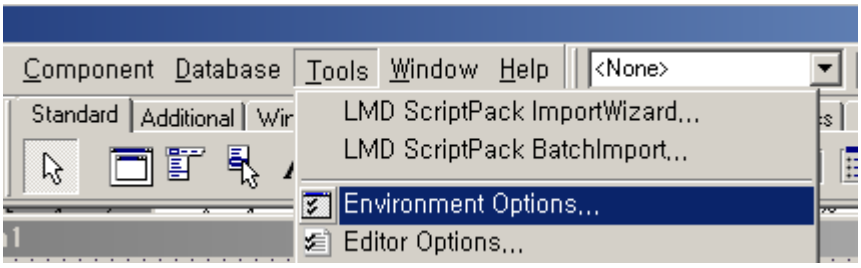


그림 15-28 C++ Builder 6 에서 ActiveX Component 등록 1

12. Enviroment Options창의 Type Library 탭에서 “Ignore special CoClass Flags when importing”, “Can Create” - Check 한뒤 OK버튼을 클릭합니다.

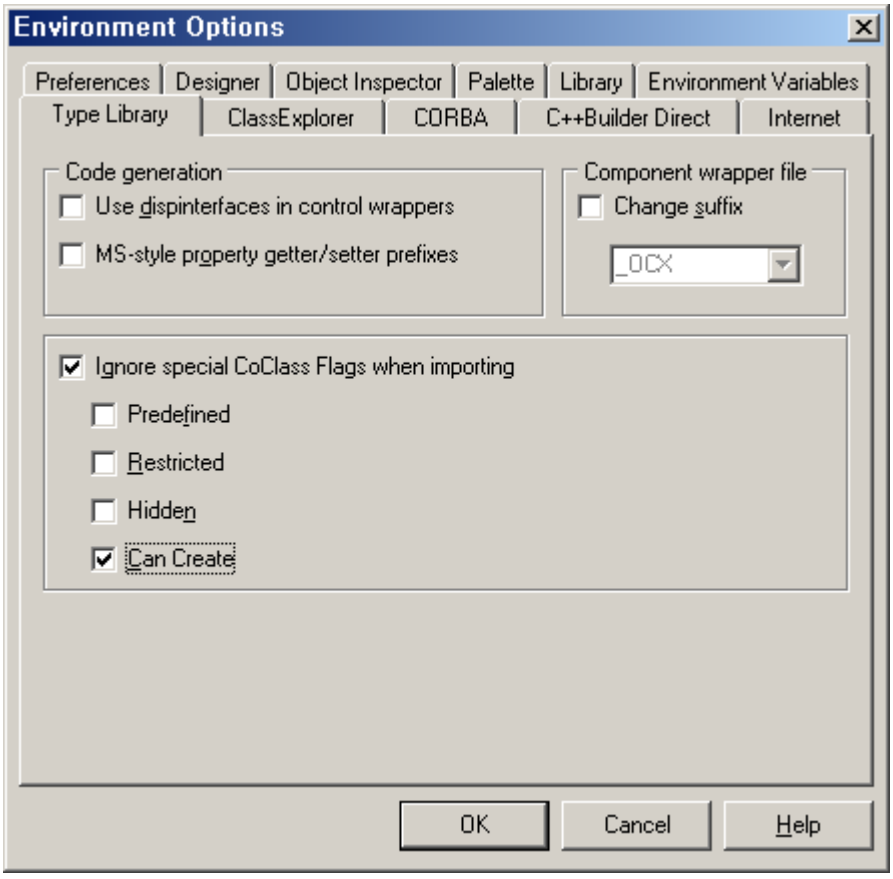


그림 15-29 C++ Builder 6 의 Environment Options 창의 화면

13. [Component]-[Import ActiveX Control..]를 선택한뒤 절차에 따라 OCX를 등록하면 됩니다.

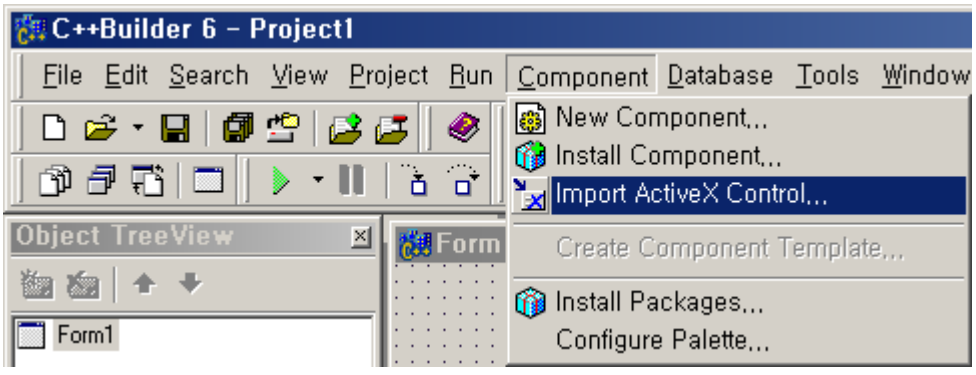


그림 15-30 C++ Builder 6 에서 ActiveX Component 등록 3

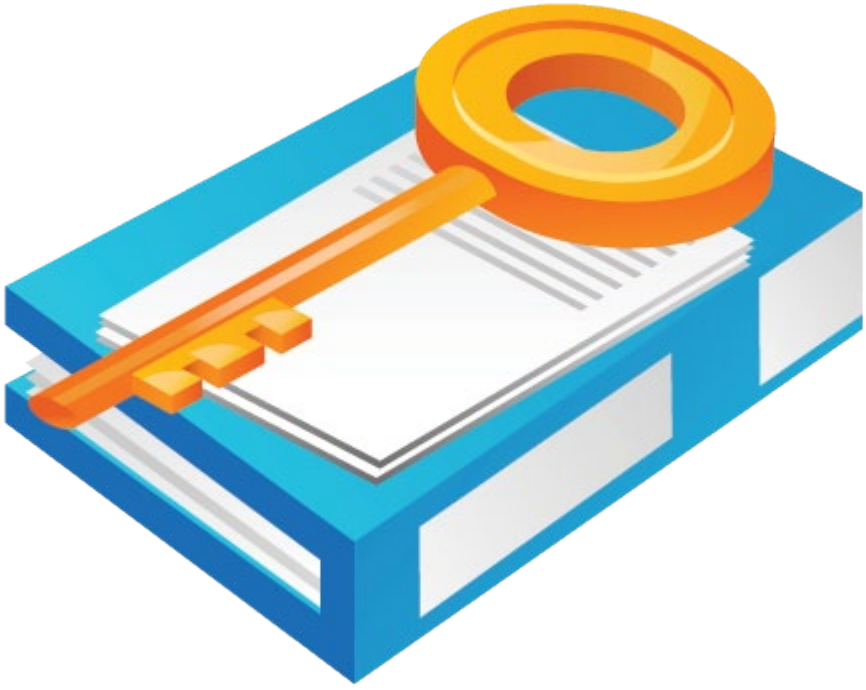
Chapter

F

Index of CMMSDK Functions

필요한 함수(函數)를 가장 빠르고 쉽게 찾으십시요. CMMSDK 매뉴얼에서는 고객(顧客)님들께서 원하시는 함수들을 일목요연하게 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 함수(函數)를 찾으실 수 있도록 구성(構成)하였습니다. Adobe 社의 Acrobat Reader 와 같은 전자 문서 전자 문서 뷰어(Viewer) 를 통해 최단 시간내에 원하시는 함수(函數)를 찾을 수 있습니다.



V Index of CMMSDK Functions

V.I Quick Reference to CMMSDK Functions

General Functions

TEST & MEASUREMENT & AUTOMATION / COMIZOA

CMMSDK Manual

저작권자 : ㈜커미조아
Copyright (c) by COMIZOA CO.,LTD. All right reserved.
JULY 2009 P/N 0715-2009-02

2007 년 8 월 1 일 1 판 인쇄
2011 년 4 월 1 일 2 판 인쇄
매뉴얼 자료 번호 : 4.0.5

㈜커미조아
http://www.comizoa.com
Tel) 042 – 936 – 6500~6
Fax) 042 – 936 – 6507

이 사용자 설명서 상의 삽입된 삽화 및 예제 프로그램을 포함한 전체 내용은 대한민국 저작권법에 의해 보호되고 있습니다.
㈜커미조아의 사전 서면 동의 없이 사용자설명서의 일부 또는 전체를 어떤 형태로든 복사, 전재할 수 없습니다.