COMIZOA Compact EIP Series

cEIP Software Development Kit

CESDK API REFERENCE MANUAL

MARCH 2009

P/N 0403-2009-01

API Reference Manual

TEST & MEASUREMENT & AUTOMATION / COMIZOA

ceSDK Manual

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 기술 지원

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

본사 안내
대전 광역시 유성구 탑립동 707 번지 DTV-PostBI 센터

전화번호 : 042-936-6500
모사전송 : 042-936-6507

COMIZOA cEIP System Integrated Control Library Reference

Table of Contents

Chapter

1

Introduction

본 장에서는 제품 보증안내를 비롯한 제품 보증 규정, 저작권, 상표안내, 주의사항을 설명하고 있습니다. 다양한 제어 환경에서 보다 강력하고 빠른 제어를 위해, 그리고 안정적인 제어를 위해 선택하신 저희 ㈜ 커미조아 제품은 이제 고객님에게 큰 감사와 보답으로 이 바지 하도록 하겠습니다.

가장 안정적이고 정밀한 제어, 그리고 고객 중심에서 제품의 완성을 추구하는 ㈜ 커미조아는 고객님들을 위해 언제나 최선과 정성을 다하는 자세로 지금 이 시간에도 보다 나은 제품 개발을 위해 성실과 열정을 바탕으로 제품에 꿈을 담고 있습니다.

먼저, 국내 최고의 모션 및 DAQ 기술력을 자랑하는 저희 ㈜ 커미조아 제품을 선택하여 주신 여러 분들께 다시 한번 감사의 말씀을 드립니다. 본 장에서는 제품 보증 안내, 규정, 저작권, 상표 안내에 대한 내용을 설명하고 있습니다. 이 내용은 cEIP 제품 운용과는 별개의 내용이오나, 제품의 보증과 규정 그리고 저작권에 대한 내용을 설명 드리고 있으므로 숙지하여 주시기 바랍니다.

1 cEIP 사전 안내 사항

1.1 Overview

1.1.1 제품 보증 안내

저희 ㈜ 커미조아는 고객 여러분들께 가장 안정된 소프트웨어와 하드웨어를 공급함으로써, 고객 여러분들을 만족시켜드리는 것을 최우선의 목표로 하고 있습니다. 만약 구입하신 제품에 외관상의 하자, 동작 이상 또는 불량이 발견되는 경우에는 언제든지 저희 ㈜ 커미조아를 통해 문의하여 주시기 바라며, 대리점을 통해 구입하신 경우에는 해당 구입처를 통해 문의하시면, 더욱 빠른 기술 지원 을 받으실 수 있습니다.

1.1.2 제품 보증 규정

구입하신 당사의 제품은 소비자의 과실 이외의 자체 결함 및 동작 이상에 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 저희 ㈜ 커미조아 혹은 대리점 등 해당 구입처를 통해 문의하여 주시기 바랍니다.

1.1.3 저작권

1.1.4 상표안내

본 매뉴얼에서는 인용된 각 회사(Corporation) 의 제품 상표 및 저작권을 사전에 명시합니다. 해당 사항은 대한민국 저작권법에 보호되며, 각 회사의 저작권에 대한 모든 권리나 이에 관계된 내용을 보호합니다.

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 제품군에는 제품의 특성에 따라서 하드웨어 및 소프트웨어 기술 지원이 필요한 경우가 있습니다. 필요하신 경우 본사 혹은 대리점을 통해 제품 구입 이전에 점검 또는 요청하여 주시기 바랍니다.

㈜ 커미조아의 제품군은 제품의 성능 향상을 위해 예고 없이 변경될 수 있습니다. 고객님들께서는 본 매뉴얼을 읽기 전에 하드웨어 및 소프트웨어의 최신 변경사항에 대한 정보를 ㈜ 커미조아를 통해 요청하실 수 있습니다.

본 매뉴얼은 ㈜커미조아 ceSDK 에 대한 정보를 포함한, 실제 라이브러리를 통한 모션 및 DAQ 제품군의 제어 사항에 대한 기반 설명을 포함하고 있습니다.

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

1.1.6 매뉴얼 용어 안내

본 매뉴얼에서는 필요에 따라 매뉴얼에 기술된 내용을 설명하기 위해 함축된 의미의 용어나 현장 용어를 사용할 수도 있습니다. 보충 설명이 필요한 내용에 대해서는 해당 단원을 통해 안내하여 드립니다.

매뉴얼 전체 범위에서 사용되는 범용적인 용어의 의미는 다음과 같습니다.

명칭	의미
모터 (Motor)	서보 모터를 비롯한 스텝 모터.
서보팩 (Servo Pack)	서보 앰프.
서보 드라이버 (Servo Drive)	서보 모터와 서보 앰프의 편성.
스텝 드라이버 (Step Drive)	스텝 모터와 스텝 모터 제어 회로의 편성.
서보 시스템 (Servo System)	서보 드라이버와 ㈜ 커미조아 상위 제어기를 조합한, 일련의 완성된 시스템.
지령 위치 출력 신호 (Command Pulse)	모션 제어 제품에서 서보 팩 혹은 스텝 드라이버 측으로 출력하는 전기적 펄스 (Pulse) 열 신호.
실제 위치 입력 신호 (Feedback Pulse)	모션 제어 제품에서 위치 검출기(Encoder) 등으로 입력 받는 전기적 펄스 (Pulse) 열 신호.
원격 노드 (Node)	㈜ 커미조아의 cEIP 제품 중 노드 마스터 모듈을 의미하며, ceNM-SE 제품 혹은 노드 마스터 모듈 및 슬레이브 모듈을 종합적으로 일컫는 명칭.
모션 모듈 (Motion Module)	㈜ 커미조아의 cEIP 제품 중 ceMC02P, ceMC04P 등의 슬레이브 모듈을 의미하며, 모션 제어를 위한 제품을 일컫는 명칭.
디지털 입·출력 모듈 (Digital I/O Module)	㈜ 커미조아의 cEIP 제품 중 ceD16CM, ceDI32N, ceDO32N 등의 슬레이브 모듈을 의미하며, 범용 디지털 입·출력 제어를 위한 제품을 일컫는 명칭.
아날로그 입력 모듈 (Analog Input Module)	㈜ 커미조아의 cEIP 제품 중 ceAI08A 등의 슬레이브 모듈을 의미하며, 아날로그 입력(A/D) 제어를 위한 제품을 일컫는 명칭.
아날로그 출력 모듈 (Analog Output Module)	㈜ 커미조아의 cEIP 제품 중 ceAO02A, ceAO04A 등의 슬레이브 모듈을 의미하며, 아날로그 출력(D/A) 제어를 위한 제품을 일컫는 명칭.
카운터 모듈 (Counter Module)	㈜ 커미조아의 cEIP 제품 중 ceCN08A 등의 슬레이브 모듈을 의미하며, 카운터 입력 제어를 위한 제품을 일컫는 명칭.
시리얼 통신 모듈 (Serial Module)	㈜ 커미조아의 cEIP 제품 중 ceSC04A 등의 슬레이브 모듈을 의미하며, 시리얼 통신 (RS232, RS422, RS485) 제어를 위한 제품을 일컫는 명칭.
원격 노드 ID (Node ID)	노드 마스터 모듈(ceNM-SE)의 Hardware I.P 를 의미합니다.

표 1 매뉴얼 용어 정리

1.1.7 매뉴얼 아이콘의 설명

매뉴얼에서 안내되는 내용을 보다 신속하고 정확하게 알 수 있도록 하며, 그 의미가 바르게 전달되고 이해를 돕기 위해 다음과 같은 아이콘을 사용하고 있습니다.

	이해하기 어려운 용어나 보충이 필요한 용어, 해설 및 사전에 설명 없이 새로 나온 용어에 대해 설명합니다.
	고객님의 편의와 기능의 자세한 내용에 대해 부가적으로 안내되는 사항입니다.
	해당 동작이나 함수(Function) 실행에 있어, 동작의 주의를 요하는 내용을 나타냅니다.
	해당 동작이나 함수(Function) 실행에 있어, 동작의 이상이나 문제가 발생할 경우 이 사항에 대한 경고의 의미를 나타냅니다.

표 2 매뉴얼 아이콘의 설명

1.2 Features

㈜ 커미조아의 cEIP 제어 라이브러리인 ceSDK 의 장점은 다음과 같습니다.

1.2.1 독립성

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

1.2.2 호환성

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

고객님께서는 언제나 ceSDK 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 만들어 드립니다. 특히, ceSDK 의 모든 함수 명에는 의미적 명명 규칙을 내포하였으며, 이것은 분명 실무 개발 환경에서 많은 부분 이점으로 작용 할 것입니다.

1.2.4 확장성

ceSDK 라이브러리는 커미조아의 기존 제품군과 함께 사용할 수 있습니다. COMI-AX Pro 및 기존 커미조아의 다른 제품군의 라이브러리와도 상호 호환성을 보장하며, 기존 라이브러리 사용자도 빠른 시간 내에 ㈜ 커미조아 ceSDK 를 통해 제품 개발을 더욱 향상시키실 수 있습니다.

1.2.5 신뢰성

㈜ 커미조아의 제품군들에대한 라이브러리는 오랜 시간 산업 현장에서의 현장 경험을 바탕으로 형성된 신뢰성 있는 제품군 입니다. '부드럽고, 정확하고, 빠르고, 쉬운' 모션 제어 기능을 통해 모션의 기본에 충실하였으며, 각 동작에 대한 입력과 출력을 ceSDK 인터페이스 전역에 걸쳐 사용하기 쉽도록 안내해 놓았습니다.

또한, 응용 프로그램에 기반하지 않는 자체 디버그 기능을 바탕으로 응용 프로그램에 의한 오류를 최단시간 내에 해결할 수 있도록 합니다. 디버그 모드의 지원과 향상된 디버그 정보를 바탕으로 오류 발생에 대한 원인을 신속히 분석하실 수 있도록 도움을 드리며, 저희 ㈜ 커미조아는 고객님께 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 지원해 드리려고 노력하고 있으며, 예제간 중복 코드와 라이브러리 구성 이외의 부분을 최소화하여 빠른 시간 내에 예제의 코드를 바로 적용시켜드릴 수 있도록 노력하고 있습니다.

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

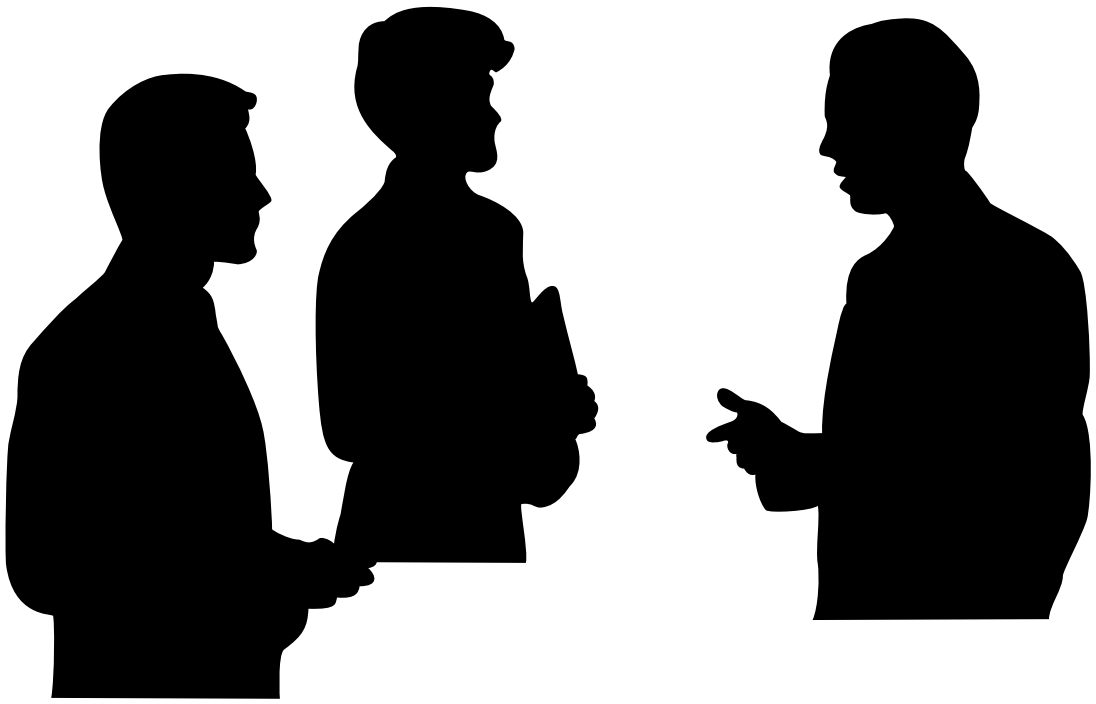
Chapter

2

Development Environment for ceSDK

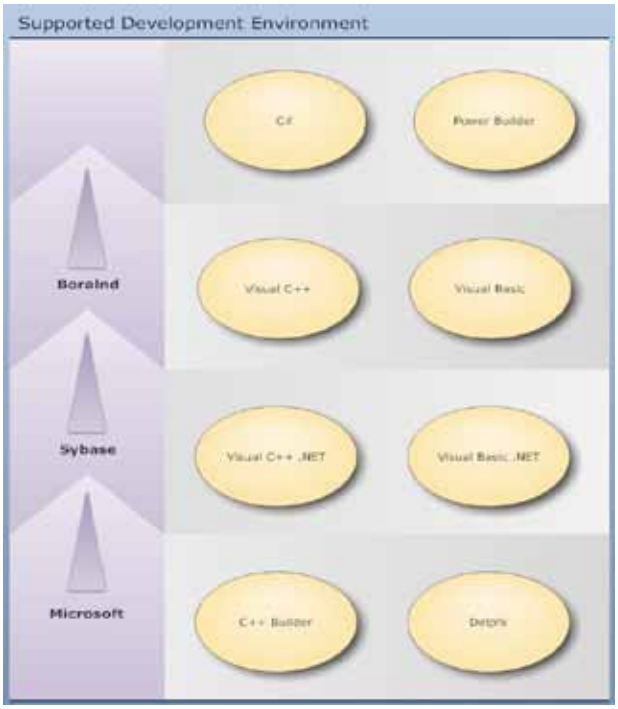
㈜커미조아는 ceSDK 라이브러리를 통해 다양한 최신 개발환경을 지원하기 위해 노력하고 있습니다. 본 장에서 다루지 않는 개발환경을 이용하시는 고객님께서는 저희 ㈜커미조아를 통해 문의하여 주시면 신속히 대처해 드리도록 하겠으며, 제공되는 ceSDK 인터페이스를 통해 보다 편리하고 빠르게 라이브러리를 사용할 수 있도록 지원하여 드립니다.

커미조아 cEIP 제품군의 ceSDK 라이브러리는 다양한 고객 여러분들의 요구에 발맞추어 사용의 편리성과 다중 플랫폼의 지원을 목표로 개발되었습니다. 제품에 대한 문의는 본사로 문의를 주시거나, 가까운 대리점을 통해 문의하시면, 더욱 빠른 기술 지원을 받으실 수 있습니다.



2 개발 환경 별 ceSDK 사용 안내

2.1 개발 환경 지원 안내



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

표 3 ceSDK 지원 개발 환경

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

2.2 ceSDK 구성

ceSDK 는 아래 그림과 같이 실제 COMIZOA 제품군의 하드웨어를 추상화하여, 공통 API 구조를 제공하 고 있습니다.

ceSDK 는 “Integration cEIP System Control Application Programming Interface” 라 명명하며, 그 의미 는 저희 ㈜ 커미조아 제품군의 기능과 그 기능을 사용하는 방법을 정의한 함수(Function)의 집합이라고 말할 수 있습니 다. 고객님께서는 이제 저희 ㈜ 커미조아 API 를 통해 제품군이 가지고 있는 다양한 기능을 사용할 수 있습 니다.

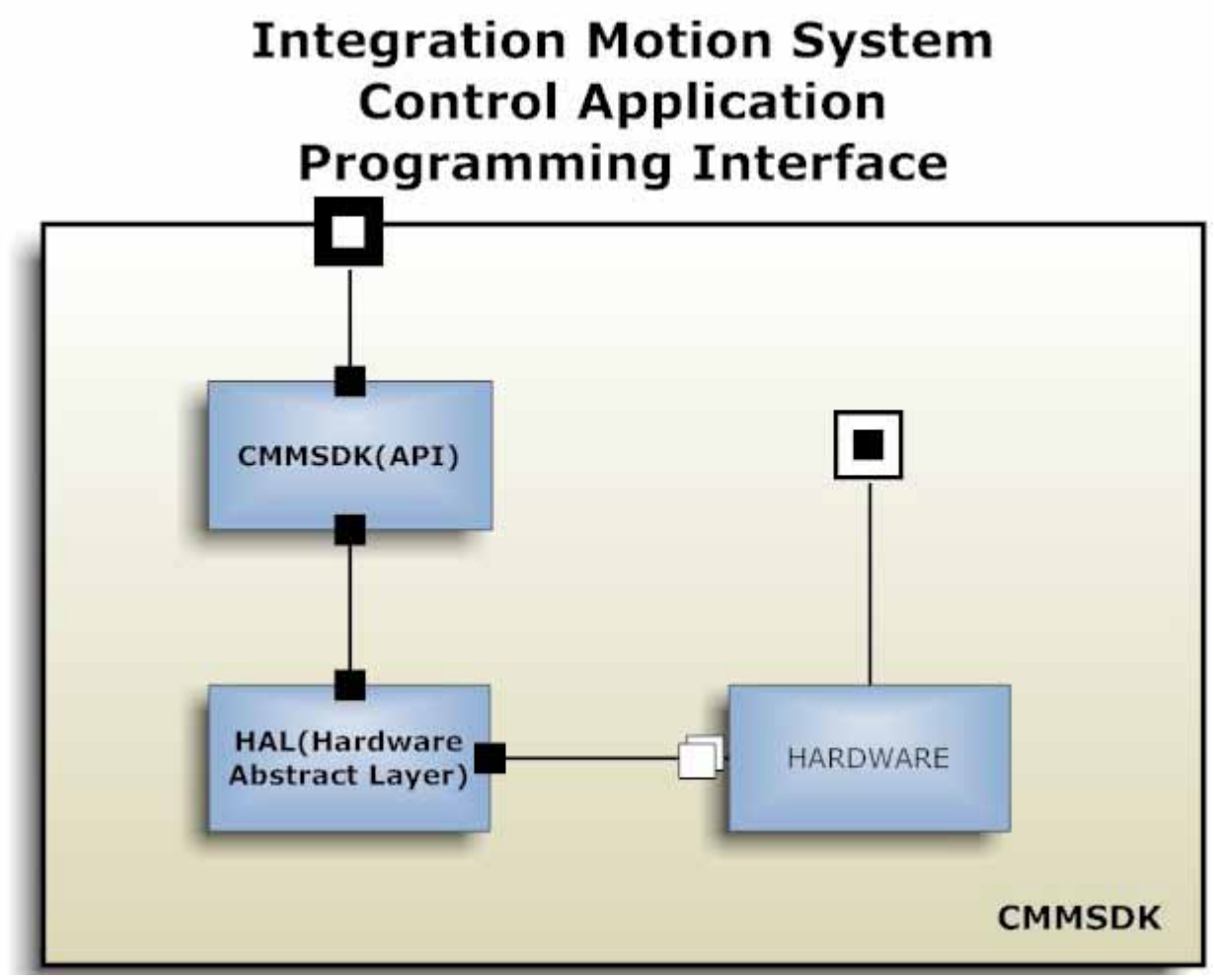


그림 2-1 ceSDK 의 구조

2.2.1 HARDWARE Layer

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

2.2.2 HAL(Hardware Abstract Layer)

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

2.2.3 ceSDK Layer (API Layer)

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

2.2.4 ceSDK 인터페이스 구성 파일

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

표 4 ceSDK 인터페이스 구성 파일 안내

Microsoft 의 윈도우 운영체제에서 동작하는 DLL(Dynamic Link Library) 형태의 ceSDK 는 상기 표에 서 나타낸 것처럼, 라이브러리 인터페이스를 위해 “ceSDK 인터페이스” 파일과 라이브러리의 반환값 및 전달인자, 각 데이터 표기등을 위한 “ceSDK 상수 정의 파일”을 제공하고 있습니다.

㈜ 커미조아 고객님께서는 “ceSDK 인터페이스 파일”을 통해 ceSDK 의 함수를 명시적으로 응용 프로그램에 포함시킬 수 있으며, 이에 따라 상기 표기한 개발 환경을 지원해드리고 있습니다. 만약 이외 의 환경에서 저희 ceSDK 를 사용하시기 원하신다면, ‘Appendix C, 다른 개발 환경 사용자를 위한 라이브러리 사용 안내’ 편을 참고해주시거나, 저희 ㈜ 커미조아 고객 지원 팀으로 연락 주시기 바랍니다.

2.3 각 개발 환경 별 안내

저희 ㈜ 커미조아에서는 고객님들의 개발 환경에 대한 고민을 덜어드리기 위하여, 범용적인 객체지향 언어인 C++ 부터 Borland International Object Pascal 의 Delphi 제품군, 그리고 최신 .NET 환경까지 고려한 라이브러리 인터페이스를 제공하고 있습니다. 특히 이번 ceSDK 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 개발 언어 및 환경에서 Header 파일또는 프로젝트 구성 파일로 반드시 사용됩니다. 저희 ㈜ 커미조아 ceSDK 에서 제공하는 인터페이스 파일의 경로는 다음과 같습니다.

다음 경로는 ㈜ 커미조아에서 제공하는 cEIP 설치 프로그램이 기본 경로인 'C:' 드라이버에 설치된 것을 기준으로 합니다.

Visual C++	
경로 명	C:\Program Files\COMIZOA\cEIPSDK\LIB\VC++
파일 명	ceSDK.cpp, ceSDK.h, ceSDKDef.h
Borland C++ Builder	
경로 명	C:\Program Files\COMIZOA\cEIPSDK\LIB\C++ Builder
파일 명	ceSDK.cpp, ceSDK.h, ceSDKDef.h
Delphi	
경로 명	C:\Program Files\COMIZOA\cEIPSDK\LIB\Delphi
파일 명	ceSDK.PAS
Visual Basic	
경로 명	C:\Program Files\COMIZOA\cEIPSDK\LIB\Visual Basic
파일 명	ceSDK.BAS
C# (CSharp)	
경로 명	C:\Program Files\COMIZOA\cEIPSDK\LIB\C Sharp
파일 명	ceSDK.cs ceSDKDef.CS

표 5 ceSDK 인터페이스 파일 경로

인터페이스 파일에 대해서 자세히 설명해 주십시오.

㈜ 커미조아에서 제공하는 ceSDK 는 DLL 형태의 독립 라이브러리 인터페이스를 제공하는 윈도우 표준 라이브러리를 말합니다. 이것은 당사의 제품 기능을 담당하는 가장 중요한 함수의 구성 집합입니다.

고객님들께서는 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 형태, 다시 말해 함수의 사용법을 알 수가 없습니다.

각 개발 환경에서 ceSDK 라이브러리를 사용할 수 있도록 인터페이스(Interface) 파일을 제공하고 있습니다. 각 개발 환경(VC++, Delphi 등) 에서 개발을 하시는 고객 여러분들께서는 반드시 개발 환경에 맞는 인터페이스 파일을 사용하셔야만, ceSDK 를 사용할 수 있습니다.

2.3.1 Visual C++ 6.x 개발자를 위한 안내

Microsoft Visual C++ 6.x 에서 ceSDK 를 사용하시려면 다음 절차에 따라 사용하시면 됩니다.

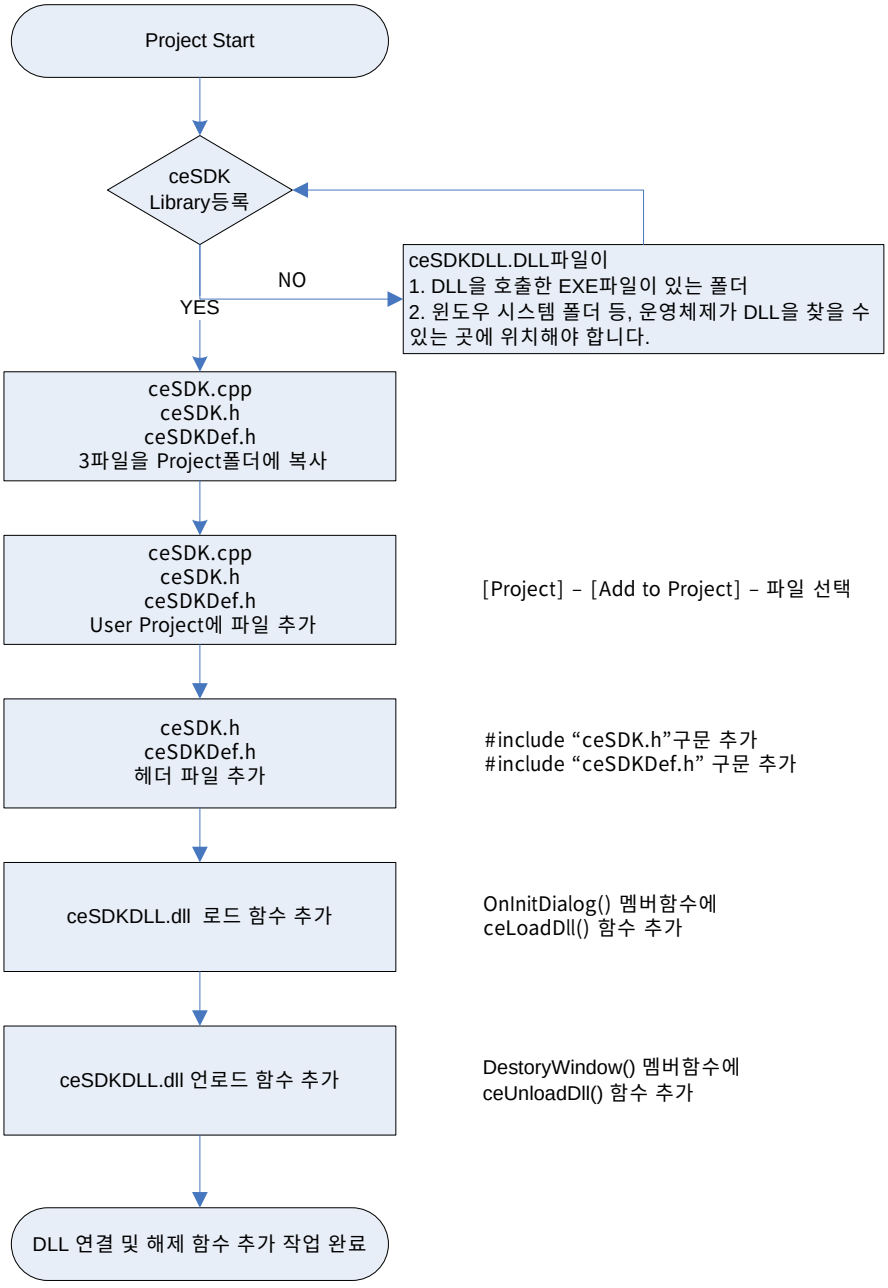


그림 2-2 Visual Studio 6.x 에서의 ceSDK 사용 순서도

Visual C++ 6.x 을 실행합니다. 메뉴에서 [File]->[New] 명령을 통해 새로운 프로젝트를 생성합니다.

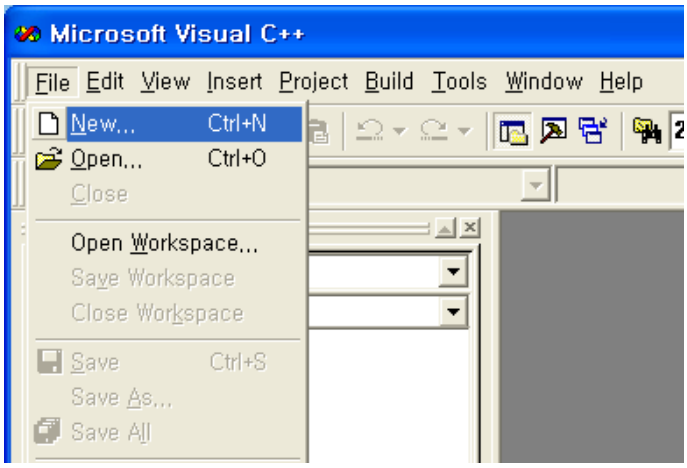


그림 2-3 Visual C++ 6.x 의 새로운 프로젝트 생성 화면

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

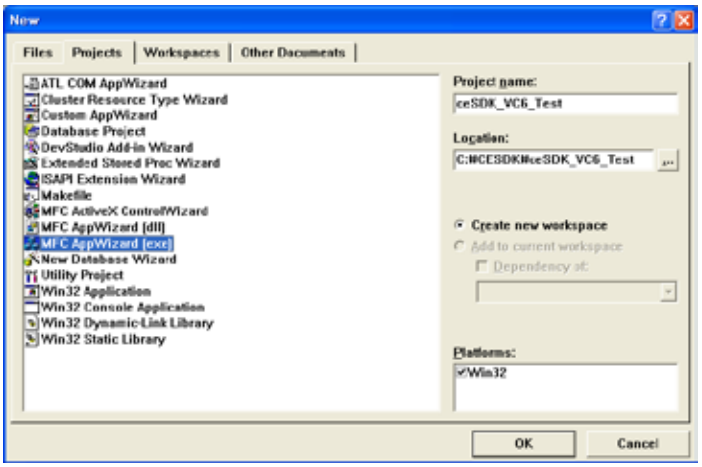


그림 2-4 새로운 프로젝트 생성 화면

'MFC AppWizar'd 창이 나타나면 [Dialog based]를 선택하고 [Finish]버튼을 클릭합니다.

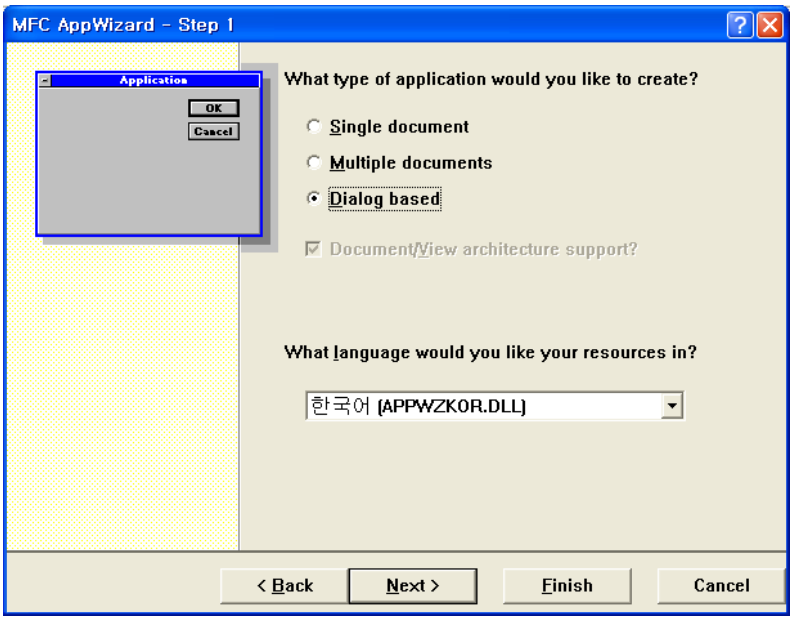


그림 2-5 MFC AppWizard 의 Application Type 선택 화면

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

메뉴에서 [Project]->[Add To Project]->[Files] 명령을 통해 ceSDK 인터페이스 파일을 추가합니다.

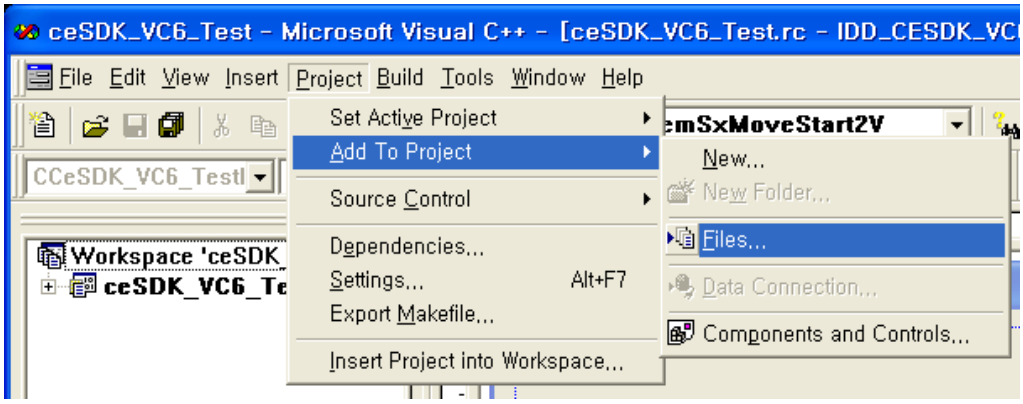


그림 2-6 프로젝트에 새로운 파일 추가 선택 화면

추가될 파일을 선택한 후 [OK]버튼을 클릭하여 ceSDK 인터페이스 파일을 프로젝트에 추가 합니다.

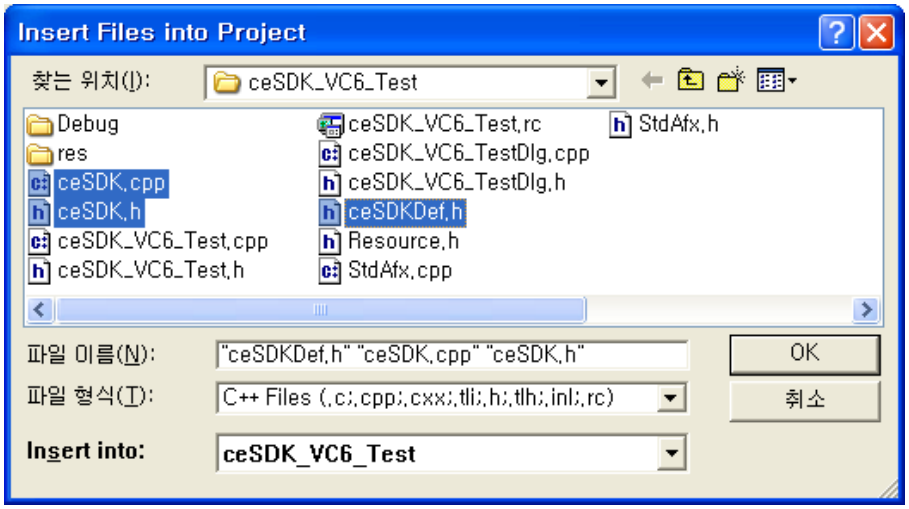


그림 2-7 프로젝트에 추가할 파일 선택 화면

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

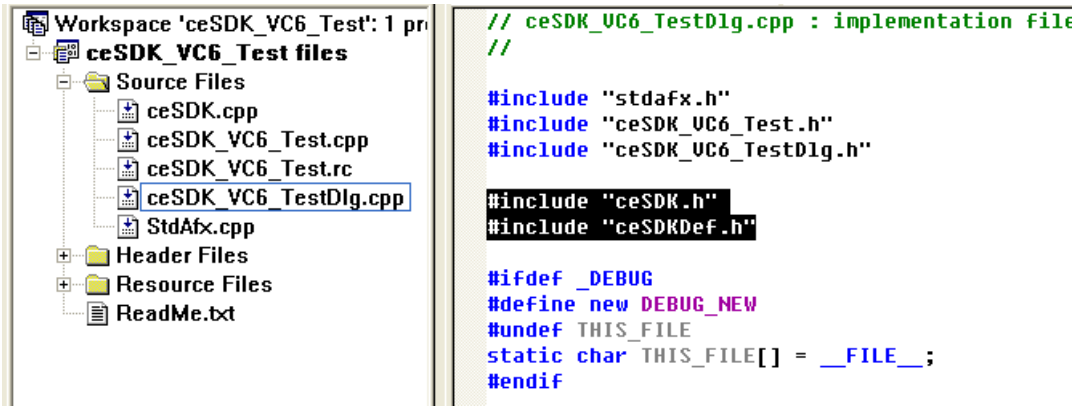


그림 2-8 사용자가 MFC AppWizard 를 통해서 생성한 소스코드에 ceSDK Header 파일을 추가 함

(생성한 프로젝트 이름)+Dlg.cpp 파일의 OnInitDialog()함수 내부의 "TODO" 아래에 ceLoadDll() 함수를 추가하여 ceSDK 라이브러리를 로드합니다.

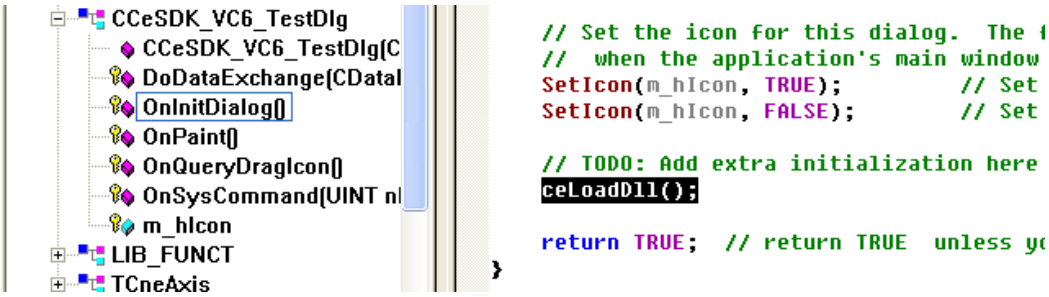


그림 2-9 DLL 로드 함수 호출 화면

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

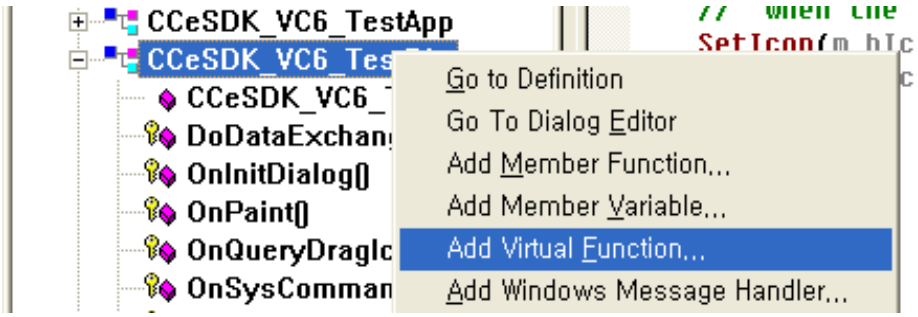


그림 2-10 가상 함수 추가

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

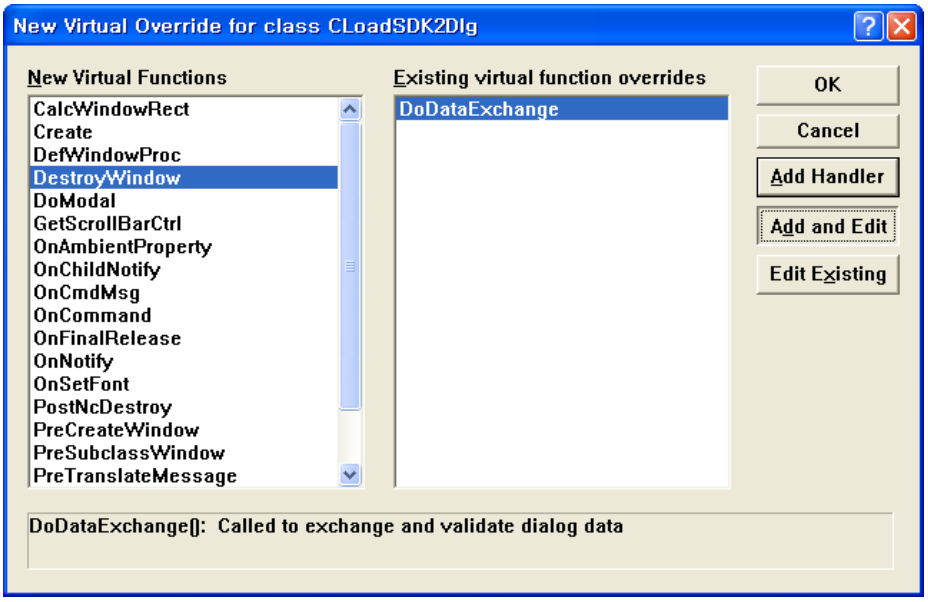


그림 2-11 DestroyWindow 함수 추가

(생성한 프로젝트 이름)+Dlg 클래스의 멤버 함수인 DestroyWindow()에 ceUnloadDll() 함수를 추가 합니다. ceUnloadDll() 함수를 추가 하면 윈도우가 종료 될 때 자동으로 DLL 이 해제 됩니다.

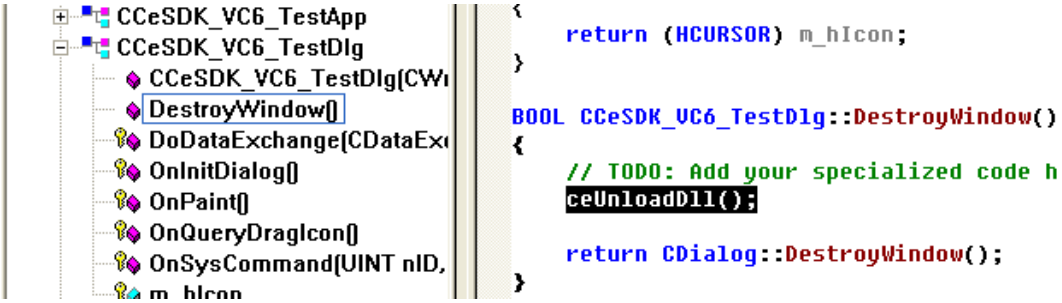


그림 2-12 DLL 로드 및 언로드 코드 추가

2.3.2 Visual C++ 7.x 개발자를 위한 안내

Microsoft Visual C++ 7.x(Visual Studio 2003)에서 ceSDK 를 사용하시려면 다음 절차에 따라 사용하시면 됩니다.

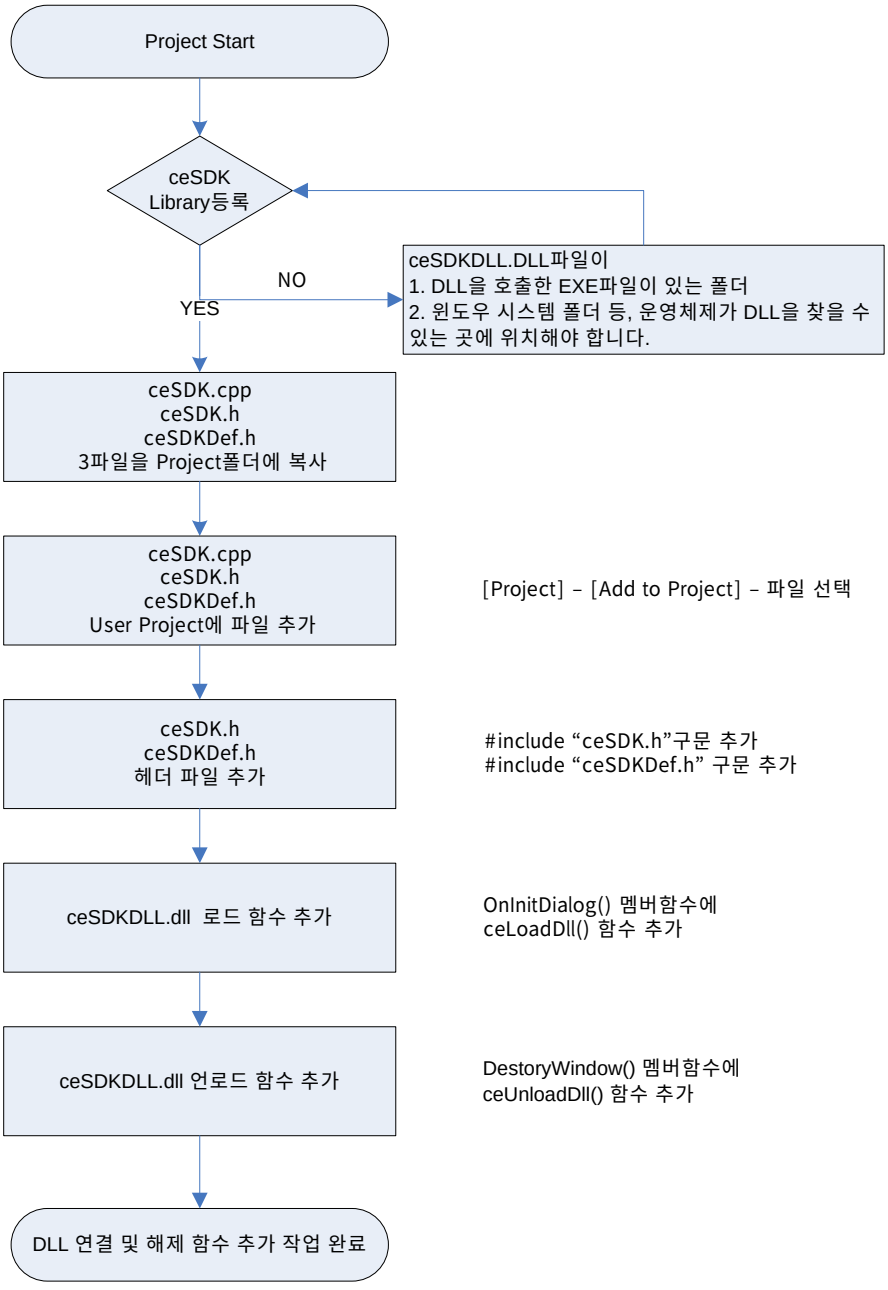


그림 2-13 Visual Studio 7.x 에서의 ceSDK 사용 순서도

Microsoft Visual Studio 2003 을 실행합니다.

메뉴에서 [파일]->[새로 만들기]->[프로젝트] 명령을 통해 새로운 프로젝트를 생성합니다.

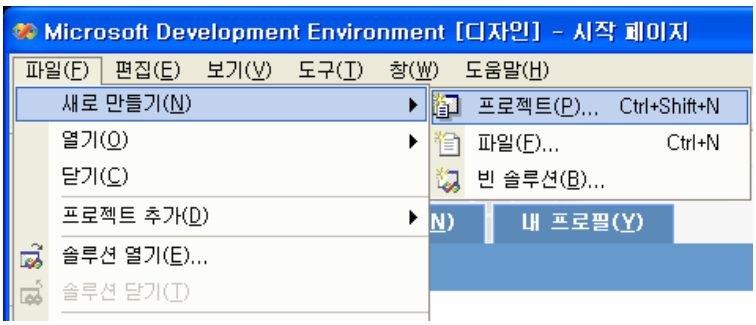


그림 2-14 프로젝트 생성의 시작

'새 프로젝트' 창의 '프로젝트 형식' 리스트에서 [Visual C++프로젝트]를 선택하고, '템플릿'에서 [MFC 응용 프 로그램]을 선택합니다. 그리고 프로젝트를 생성할 위치와 프로젝트 이름을 입력한 후 [확인] 버튼을 클릭 합니다.

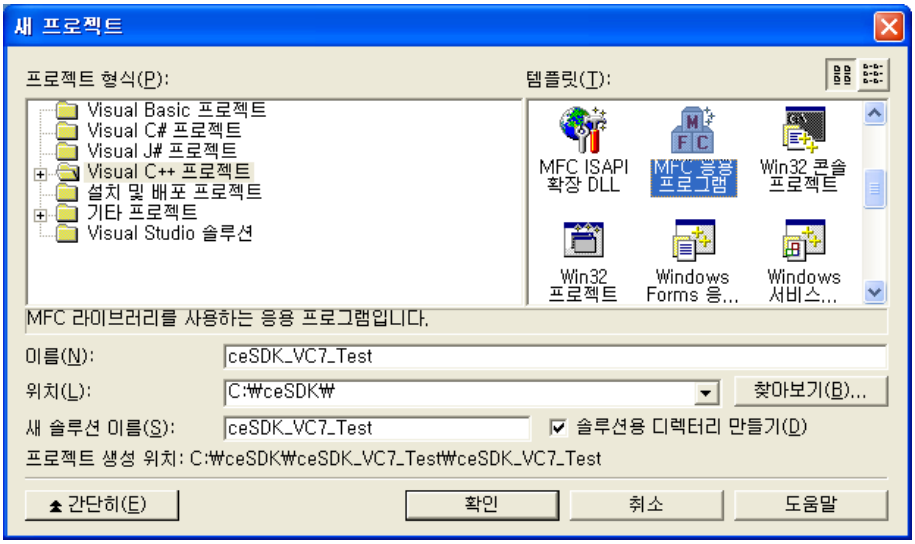


그림 2-15 프로젝트 경로 입력

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

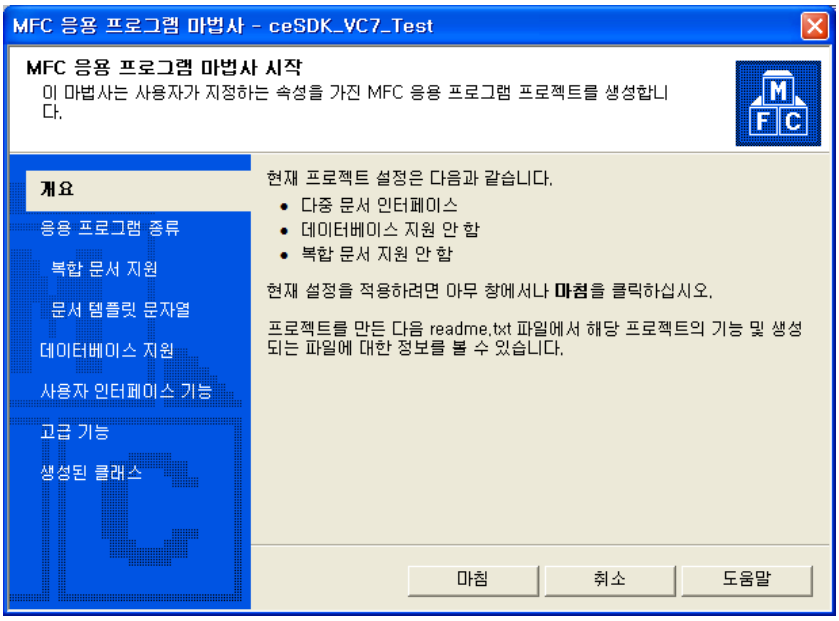


그림 2-16 응용 프로그램 종류 선택

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

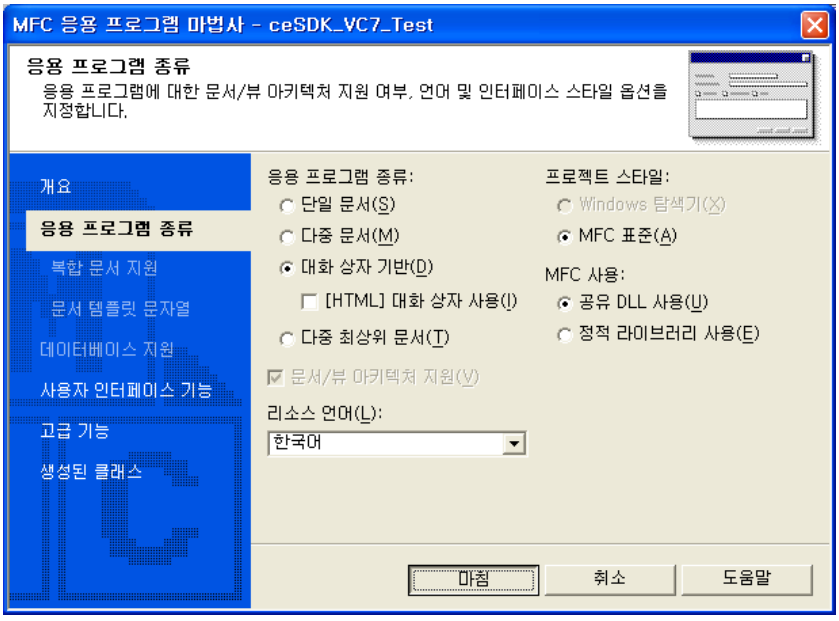


그림 2-17 대화상자 기반 프로젝트 생성

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

메뉴에서 [프로젝트]->[기존 항목 추가]를 선택합니다.

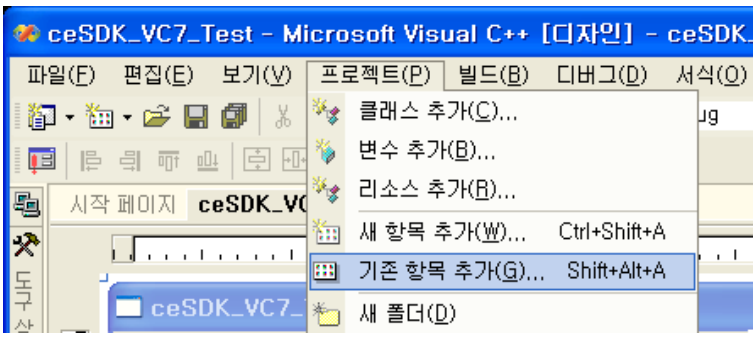


그림 2-18 기존 항목 추가 선택

추가될 파일을 선택한 후 [OK]버튼을 클릭하여 제공되는 인터페이스 파일을 프로젝트에 추가 합니다.

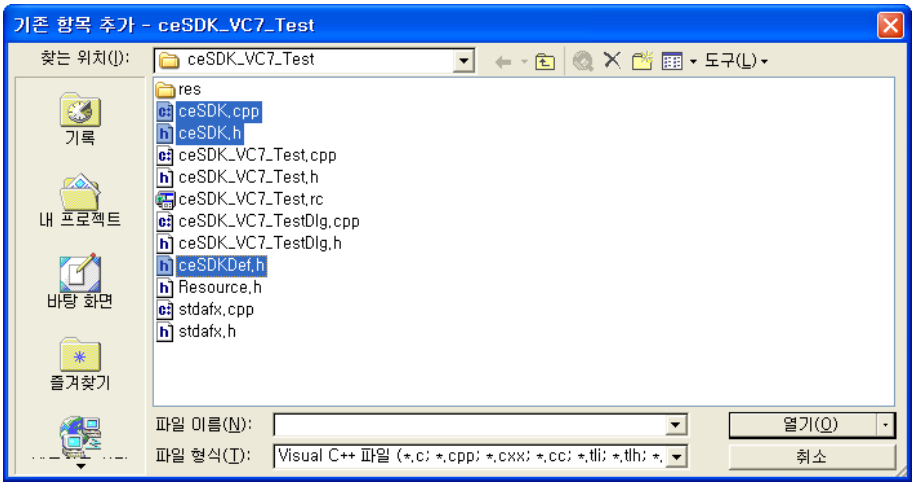


그림 2-19 헤더파일 및 소스 파일 추가

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

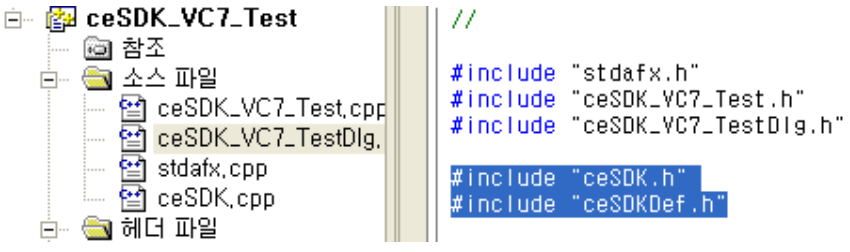


그림 2-20 인터페이스 헤더 파일 추가

(생성한 프로젝트 이름)+Dlg.cpp 파일의 OnInitDialog() 함수 내부의 “TODO” 아래에 ceLoadDll() 함수를 추가하여 ceSDK 라이브러리를 로드합니다.

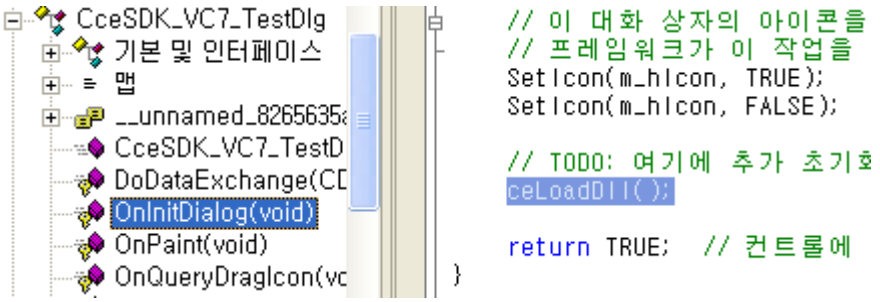


그림 2-21 Load Dll

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

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

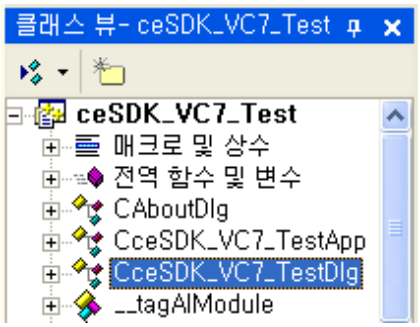


그림 2-22 사용자 생성 Dialog Class 선택

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

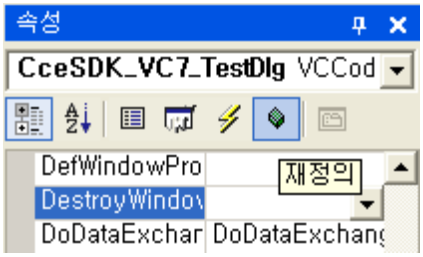


그림 2-23 Overrides 선택

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

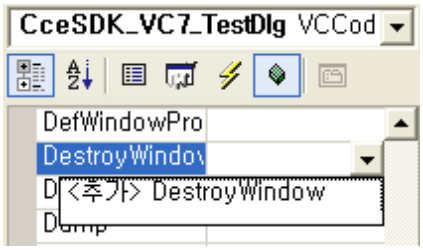


그림 2-24 Destroy Window 함수 추가

(생성한 프로젝트 이름)+Dlg 클래스의 멤버함수인 DestroyWindow()에 'ceUnloadDll()' 을 추가 합니다. 'ceUnloadDll()' 함수를 추가 하면 윈도우가 종료 될 때 자동으로 동적으로 연결된 ceSDK 가 해제 됩니다.

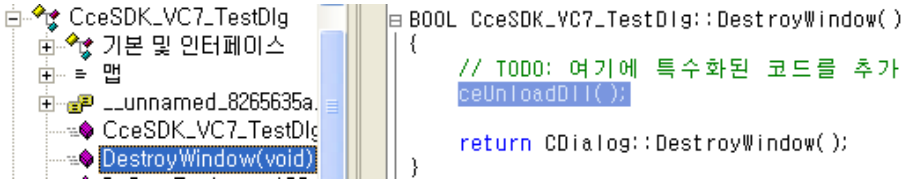


그림 2-25 UnloadDll 추가

ceSDK 의 라이브러리 방식인 DLL 의 이점은 어떤 것이 있습니까?

예를 들어 드리겠습니다. 일반적인 C 나 C++ 개발 환경에서 기본적인 Standard Input / Output Header 파일을 포함해서 사용합니다. 여기서 분명, 고객님들께서 잘 아시는 <stdio.h> 인터페이스 파일을 사용하게 됩니다.

하지만 이러한 <stdio.h> 에 정의된 함수 전체를 모두 사용하는 것은 아닙니다. 그러나 컴파일러는 <stdio.h> 에 정의된 함수의 내용들을 고객님께서 개발하시는 응용프로그램에 컴파일 시간(엄밀히 말하면 Linking 시간)에 모두 포함시킵니다. 이러한 형태는 실제 사용되지 않는 함수들 때문에 실행 파일의 크기가 커져버리며, 그만큼 메모리를 많이 차지하게 됩니다.

DLL 은 라이브러리를 실행 시간때에 연결 시키는 것을 가능하게 합니다. 즉, 필요한 기능을 실행시에 연결시킬 수 있도록 만들어 놓은 것이 DLL 이며, 각 응용 프로그램의 프로세스(Process) 코드와 데이터들은 메모리 영역의 각기 다른 별도의 영역을 갖지만 ceSDK 부분과 공유 데이터 영역은 단지 한번만 올라왔을 뿐이고 그 이후부터는 공유해서 사용합니다.

DLL 라이브러리는 함수를 다른 프로세스와 공유해서 사용하기 위해 개발되었으며, ㈜ 커미조아 ceSDK 자체는 공유영역의 메모리에 올라가서 상주하게 됩니다. 인텔 80386 이상의 CPU 에서는 프로세서(Processor) 자체에서 공유 영역을 구현할 수 있는 특수한 기법을 제공합니다.

2.3.3 Visual C++ 8.x 개발자를 위한 안내

Microsoft Visual C++ 8.x (Visual Studio 2005)에서 ceSDK 를 사용하시려면 다음 절차에 따라 사용하시면 됩니다.

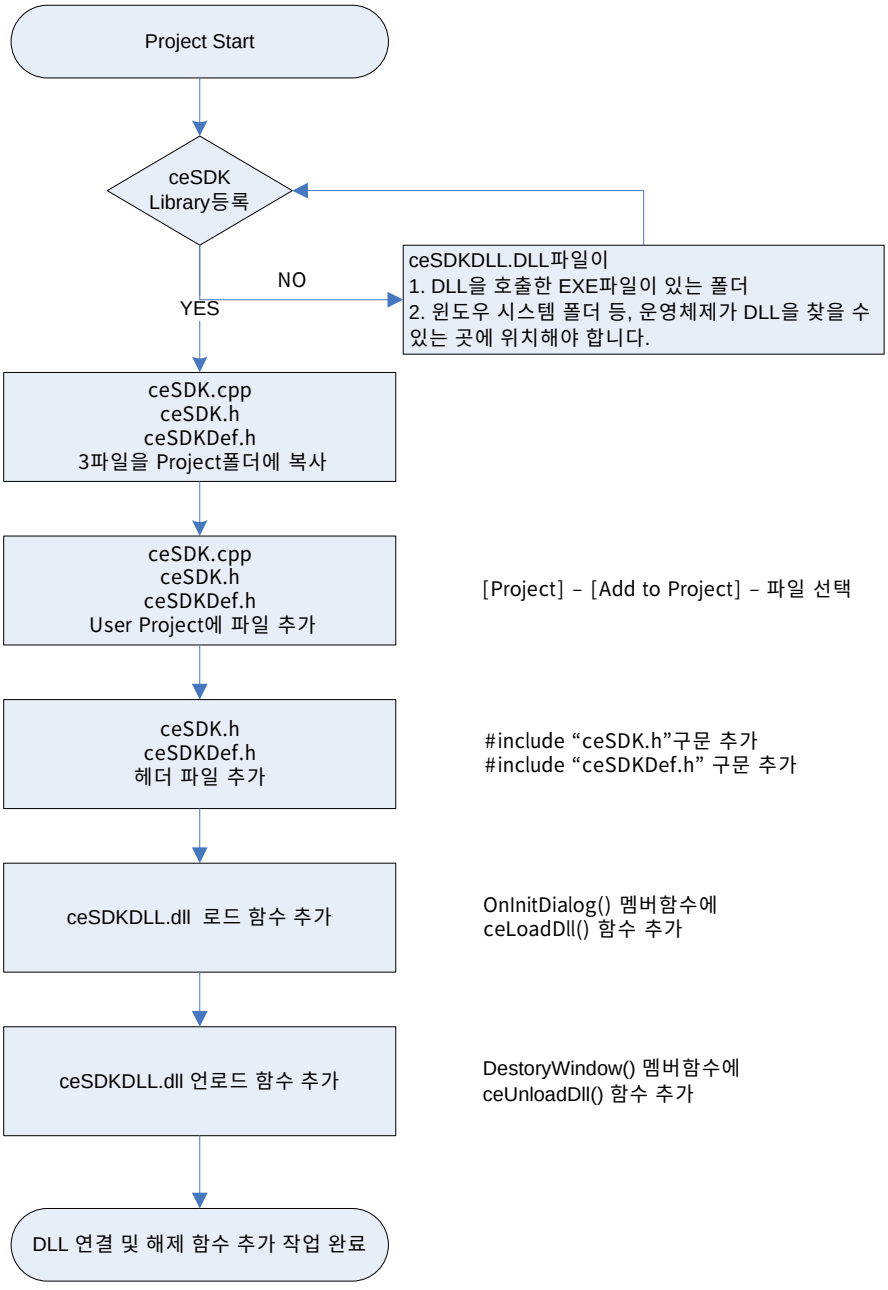


그림 2-26 Visual Studio 8.x 에서의 ceSDK 사용 순서도

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

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

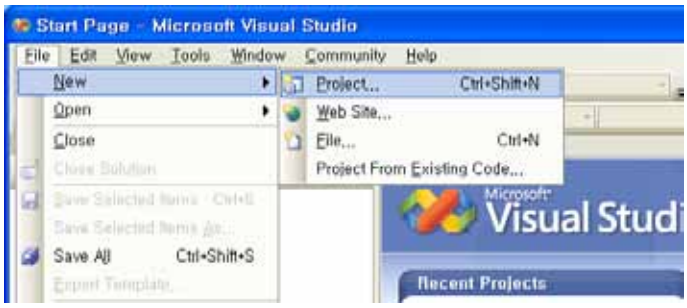


그림 2-27 새로운 프로젝트 생성 시작

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

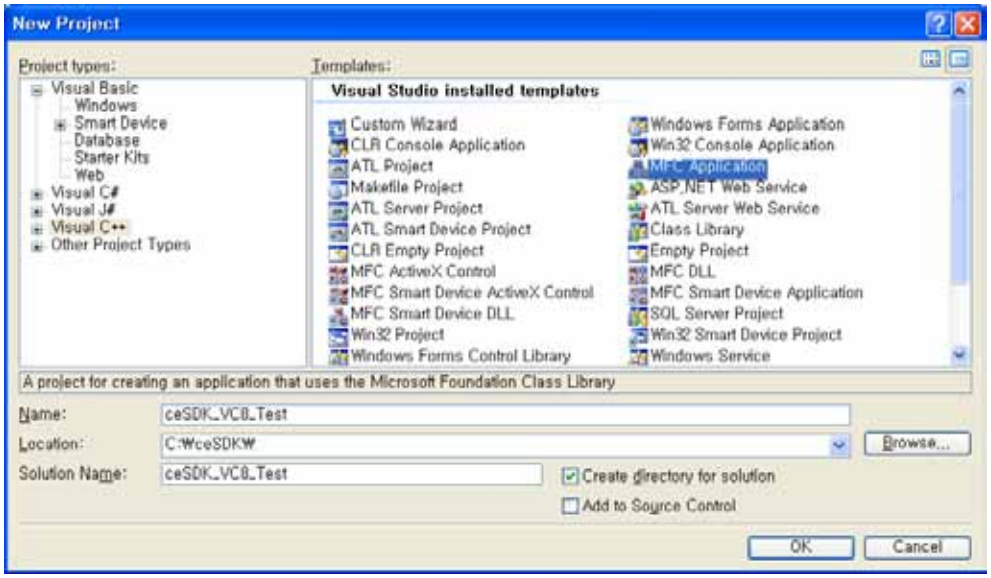


그림 2-28 새 프로젝트 옵션 선택화면

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

그림 2-29 MFC Application Wizard 의 Overview 화면

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

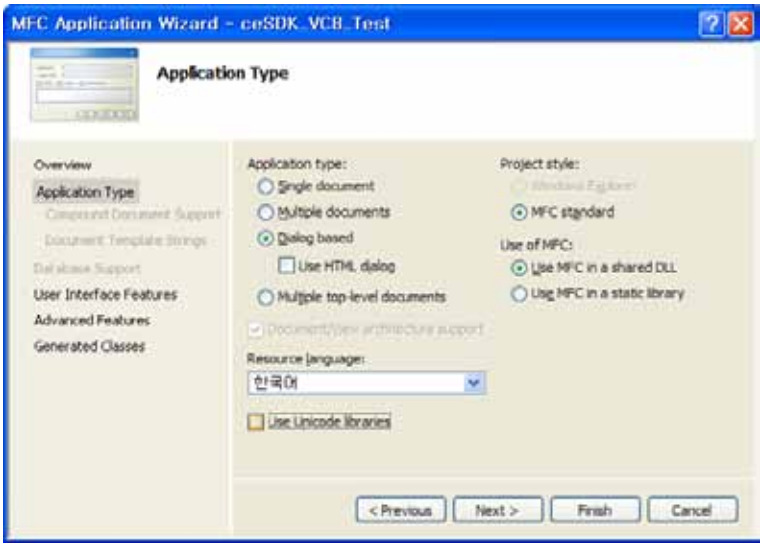


그림 2-30 MFC Application Wizard 의 Application Type 화면

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

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

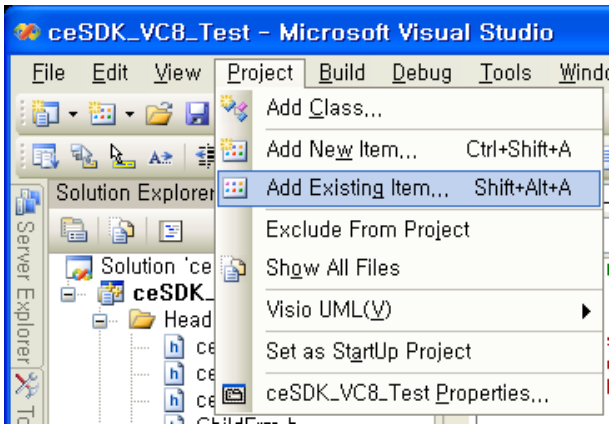


그림 2-31 메뉴에서 Add Existing Item 선택 화면

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

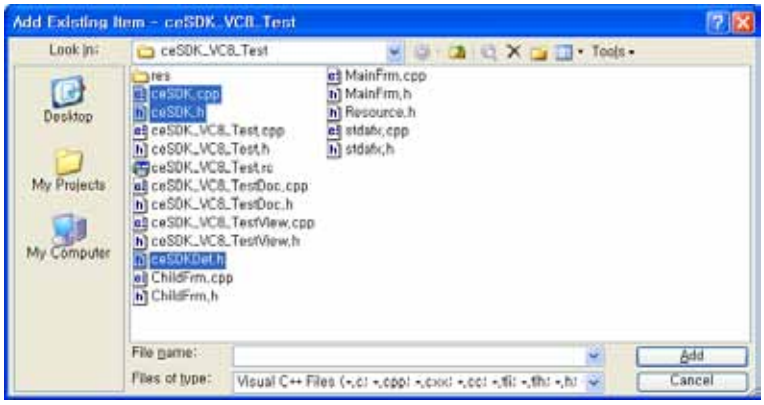


그림 2-32 새로 추가할 파일들 선택 화면

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

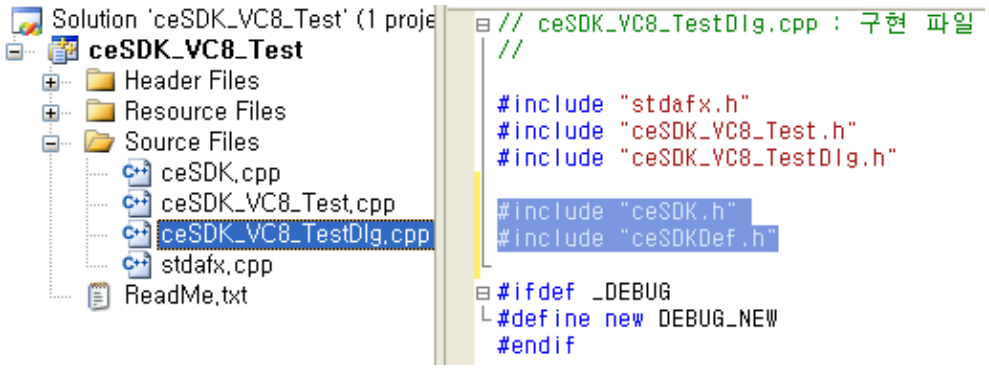


그림 2-33 사용자 생성 CPP 파일에 헤더 추가 화면

(생성한 프로젝트 이름)+Dlg.cpp 파일의 OnInitDialog() 함수 내부의 “TODO”아래에 ceLoadDll() 함수를 추가 합니다.

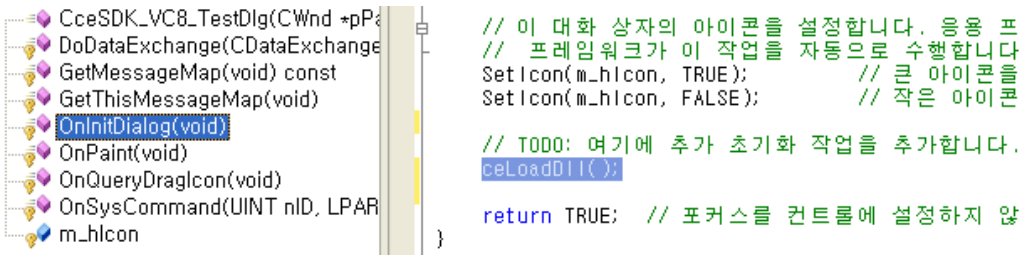


그림 2-34 LoadDll 추가

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

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

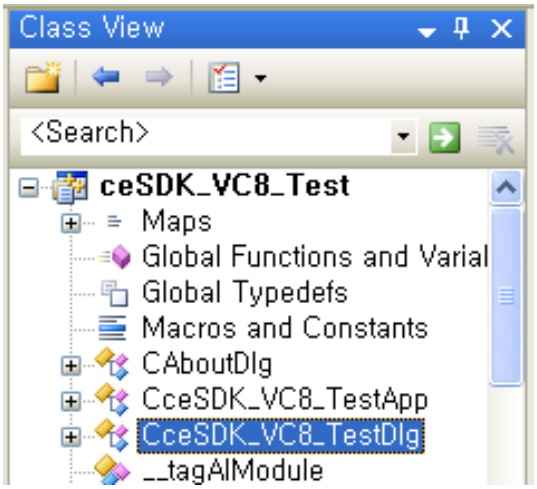


그림 2-35 사용자 생성 Dialog Class 선택

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

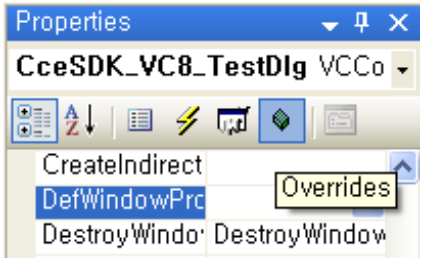


그림 2-36 Overrides 선택

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

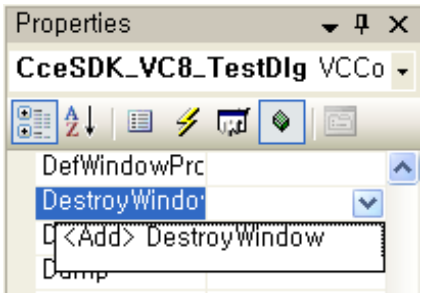


그림 2-37 Destroy Window 함수 추가

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

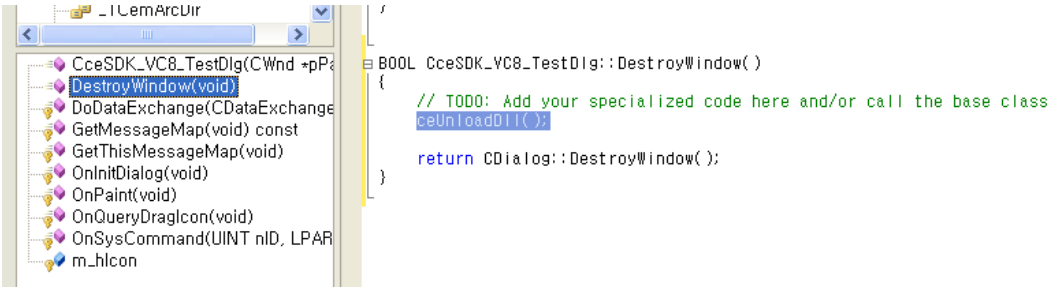


그림 2-38 UnloadDll 함수 추가

2.3.4 Borland C++ Builder 개발자를 위한 안내

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

Borland C++ Builder 에서 ceSDK 를 사용하시려면 다음 절차에 따라 사용하시면 됩니다.

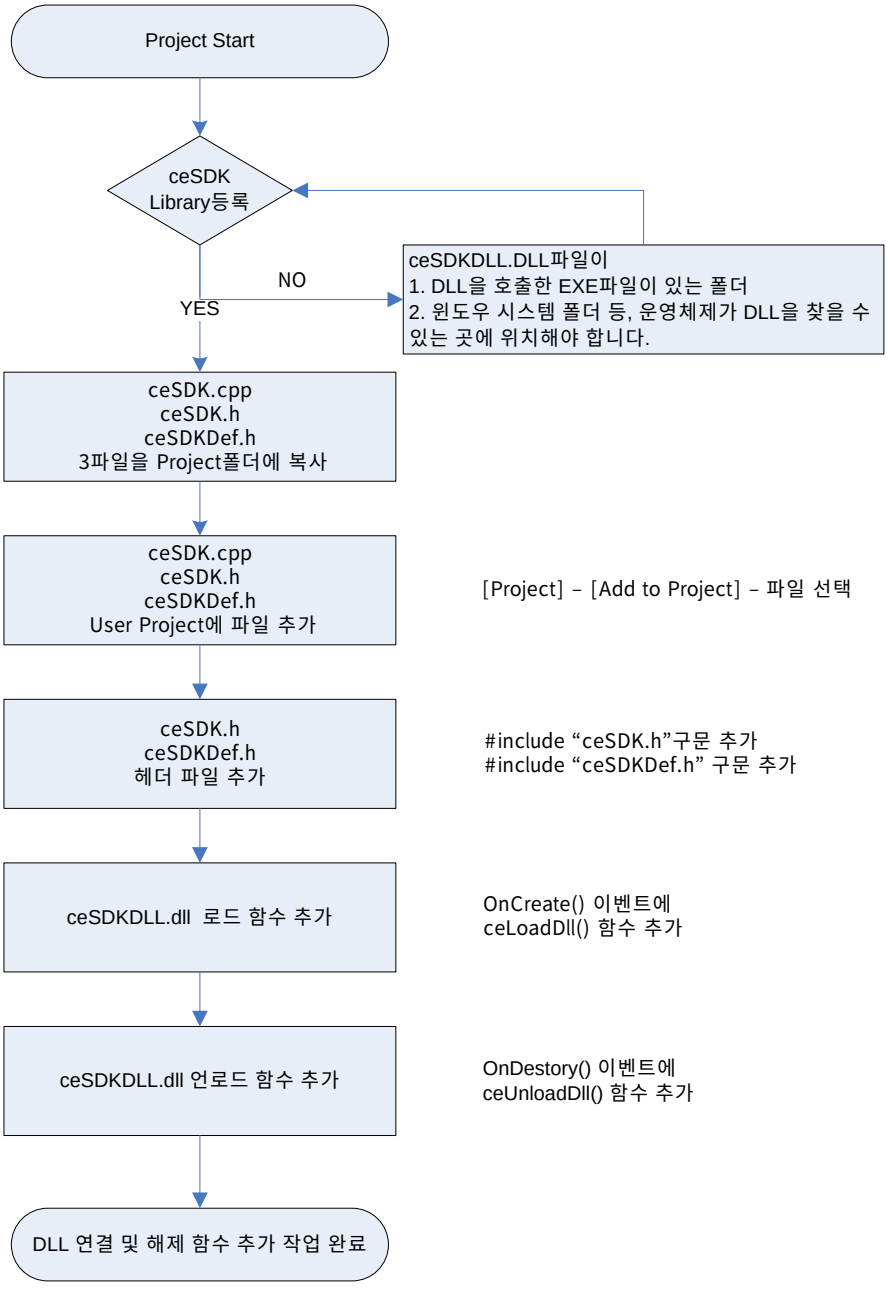


그림 2-39 Borland C++ Builder 에서 ceSDK 사용 순서도

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.

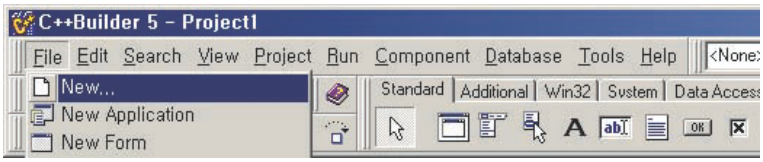


그림 2-40 BCB 5 에서 새로운 프로젝트 생성

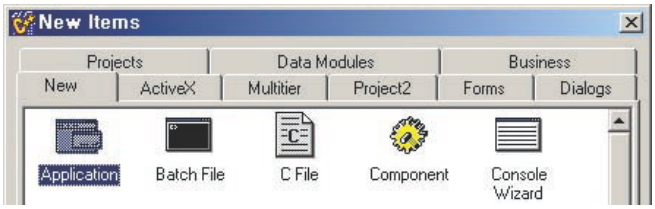


그림 2-41 BCB 5 에서 새로운 프로젝트 생성

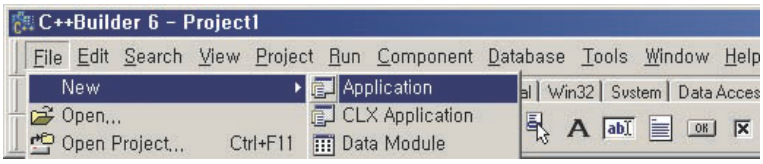


그림 2-42 BCB 6 에서 새로운 프로젝트 생성

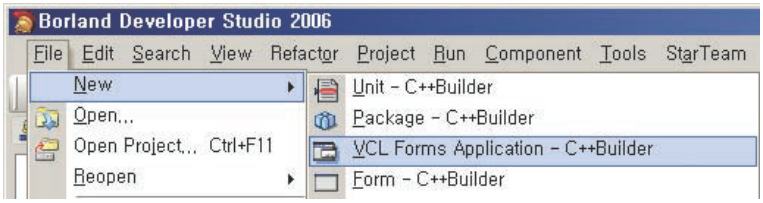


그림 2-43 BDS 2006 에서 새로운 프로젝트 생성

Borland C++ 및 VC++ 용 공용 인터페이스 정의 파일인 ceSDK.cpp, ceSDK.h, ceSDKDef.h 파일을 신규로 생성한 프로젝트 폴더로 복사 합니다.

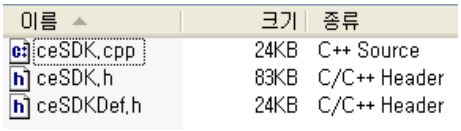


그림 2-44 ceSDK 사용시 공통으로 사용되는 파일

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

[Project] – [Add To Project] 명령을 통해 추가할 인터페이스 파일을 실제 사용자 프로젝트에 추가합니다.

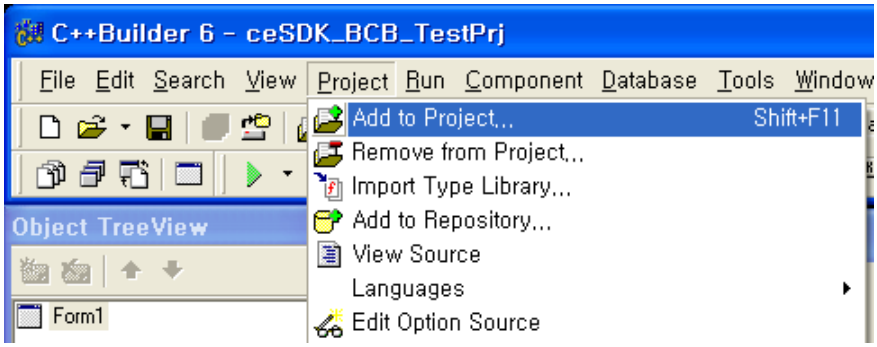


그림 2-45 C++ Builder 에서 프로젝트에 파일 추가 단계 1

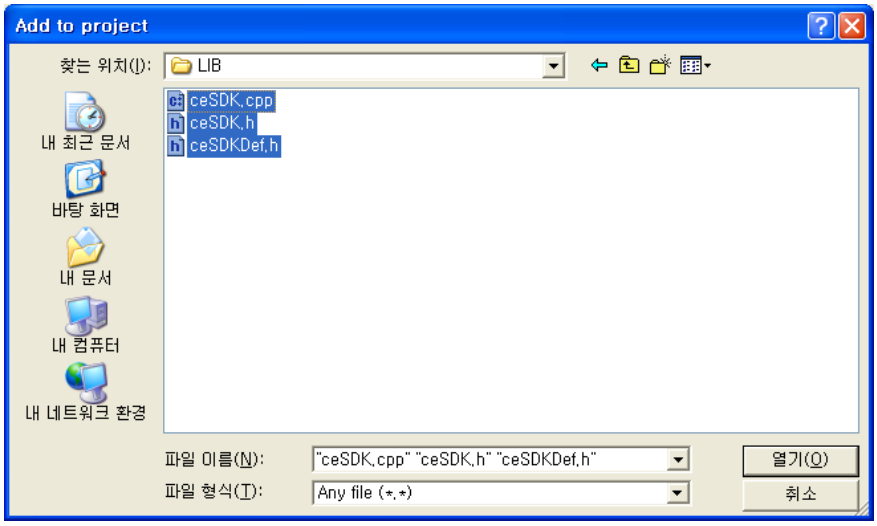


그림 2-46 C++ Builder 에서 프로젝트에 파일 추가

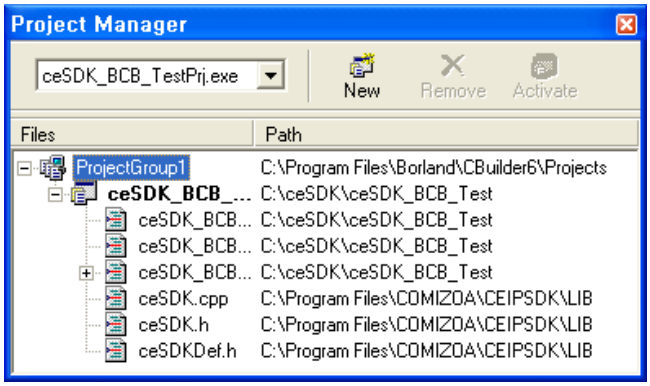


그림 2-47 Project Manager 에서 추가된 파일 확인

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

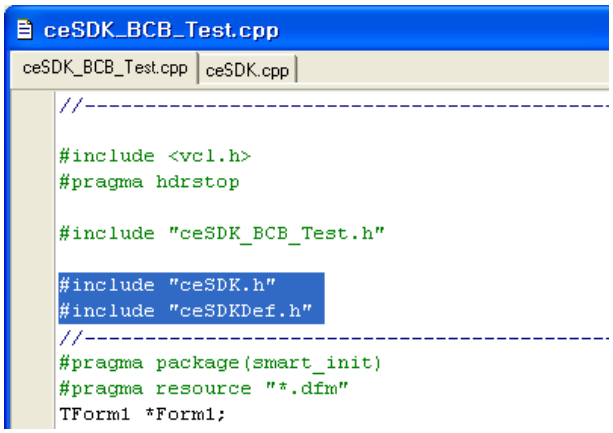


그림 2-48 라이브러리 사용시 필요한 헤더 파일 선언

ceSDK 라이브러리를 ceLoadDll() 함수를 통해 로드합니다. ceLoadDll() 함수를 추가 하는 방법은 다음과 같습니다.

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

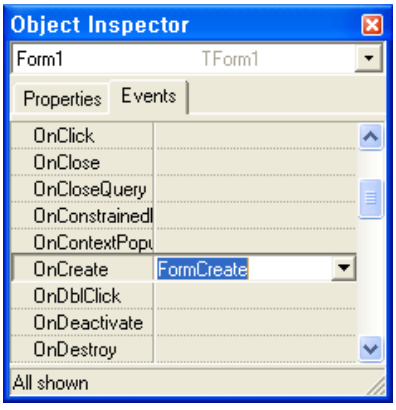


그림 2-49 OnCreate Event 추가하여 FormCreate 함수와 연결

추가된 OnCreate() 또는 응용 프로그램 시작 함수에 ceLoadDll() 함수를 추가합니다.

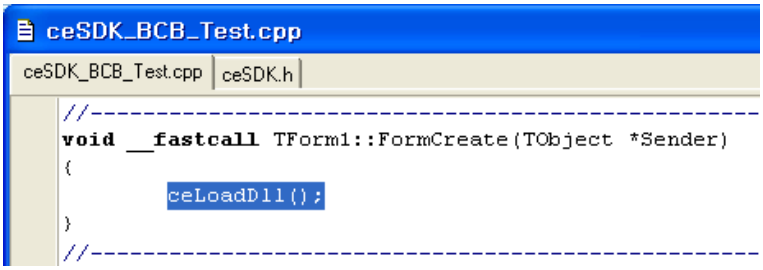


그림 2-50 OnCreate 이벤에 ceLoadDll 함수 추가

고객님이 작성하신 프로그램이 종료되면 DLL 을 명시적으로 Unload 시켜야 합니다. DLL 의 Unload 시점은 고객님께서 작성하신 응용 프로그램이 종료되는 시점에 반드시 이루어져야 하며 ceUnloadDll() 함수를 통해 이루어 집니다. ceUnloadDll() 함수를 추가 하는 방법은 다음과 같습니다.

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

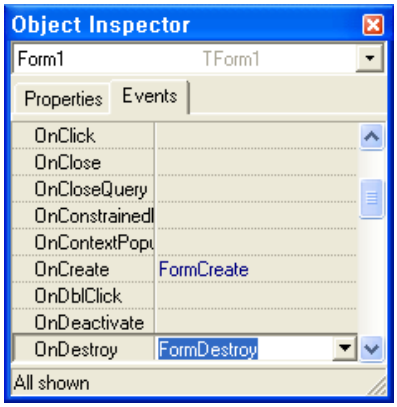


그림 2-51 응용프로그램의 종료시 DLL 이 명시적으로 UnLoad 될 수 있도록 OnDestroy Event 와 함수의 연결

추가된 FormDestroy() 또는 응용 프로그램 종료 함수에 ceUnloadDll() 함수를 추가합니다.

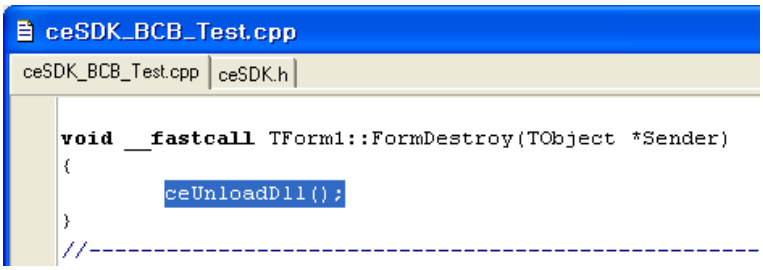


그림 2-52 FormDestroy 함수를 통하여 명시적인 UnLoadDll 함수 구현

2.3.5 Borland Delphi 개발자를 위한 안내

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

Delphi 에서 ceSDK 를 사용하시려면 다음 절차에 따라 사용하시면 됩니다.

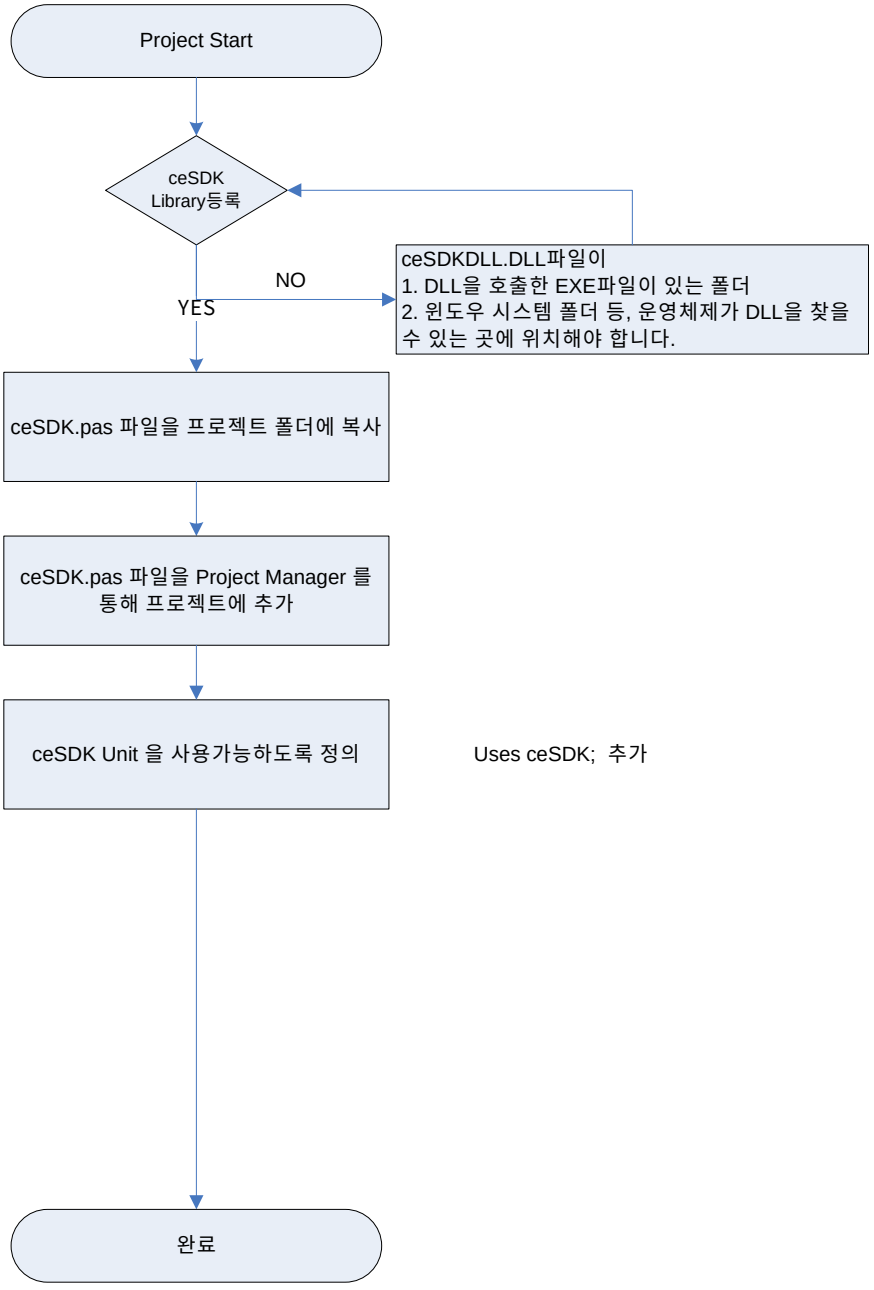


그림 2-53 Borland Delphi 에서 ceSDK 사용 순서도

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

그림 2-54 Borland Delphi 7 화면

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

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

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

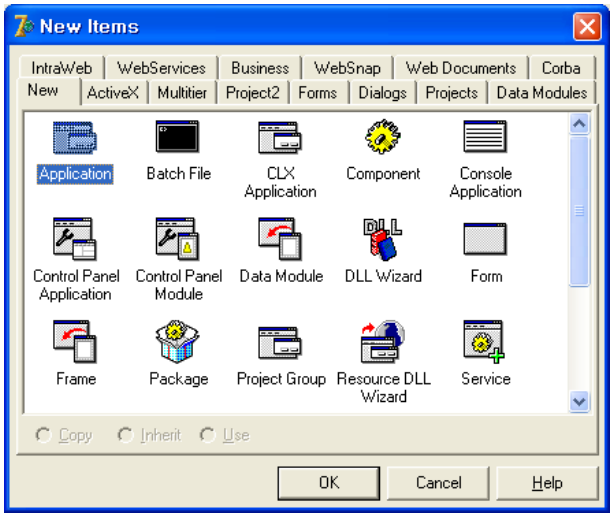


그림 2-55 Delphi 7 의 프로젝트 시작

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

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

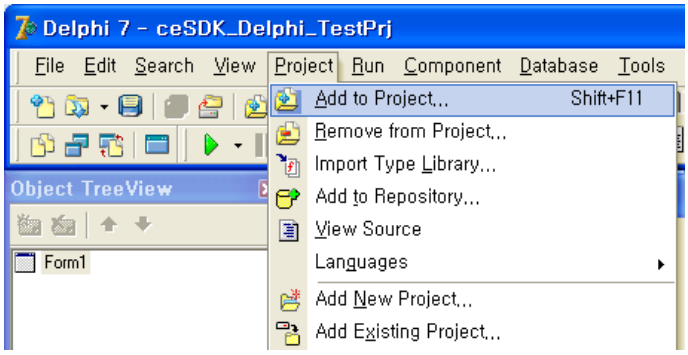


그림 2-56 Delphi 용 인터페이스 파일 추가

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

그림 2-57 Delphi 용 인터페이스 파일

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

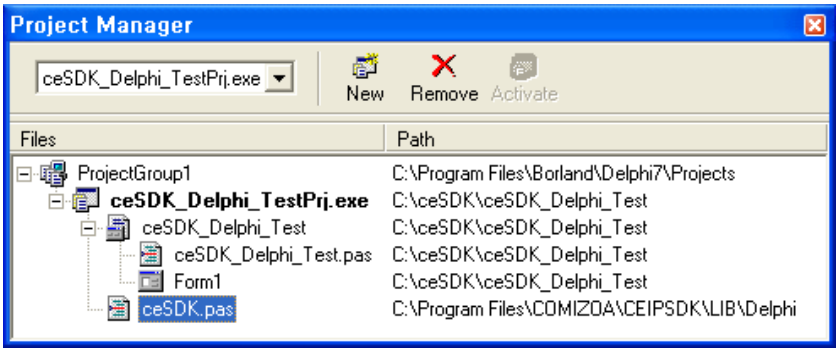


그림 2-58 Delphi 용 인터페이스 파일을 프로젝트 매니저에 추가

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

실제 구현 부의 코드를 에디터를 통해 확인합니다.

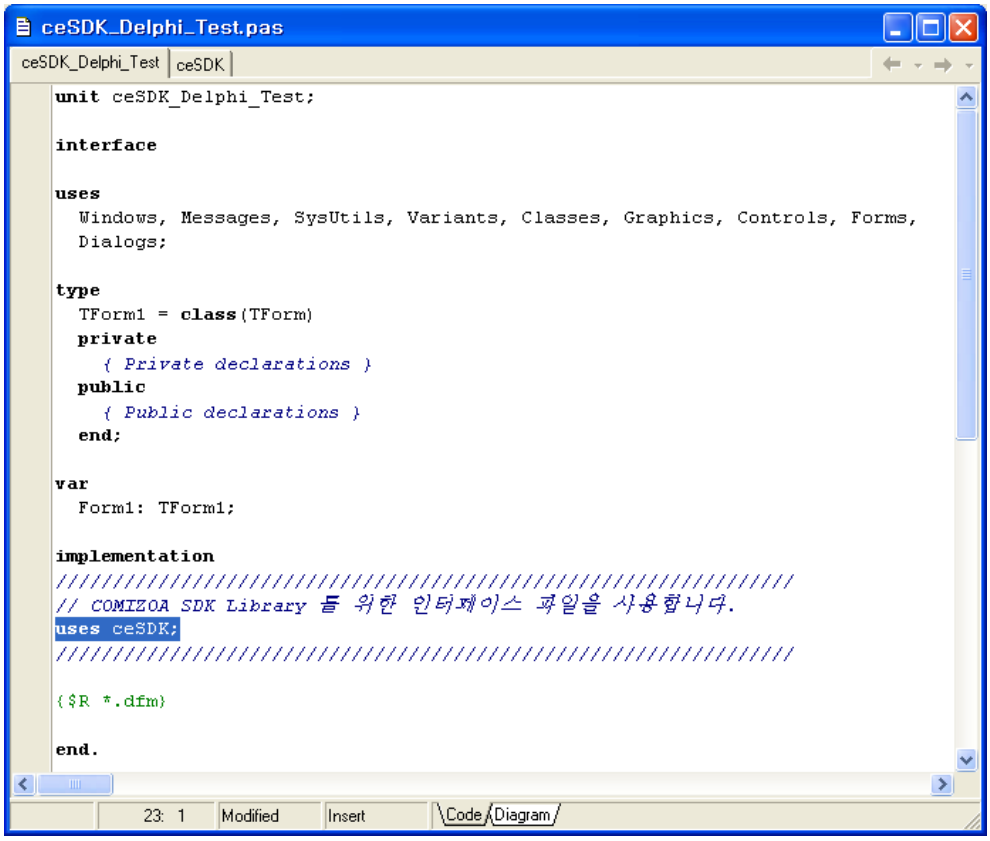


그림 2-59 uses 구문을 통해 ceSDK Unit 사용

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

2.3.6 Visual Basic 개발자를 위한 안내

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

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

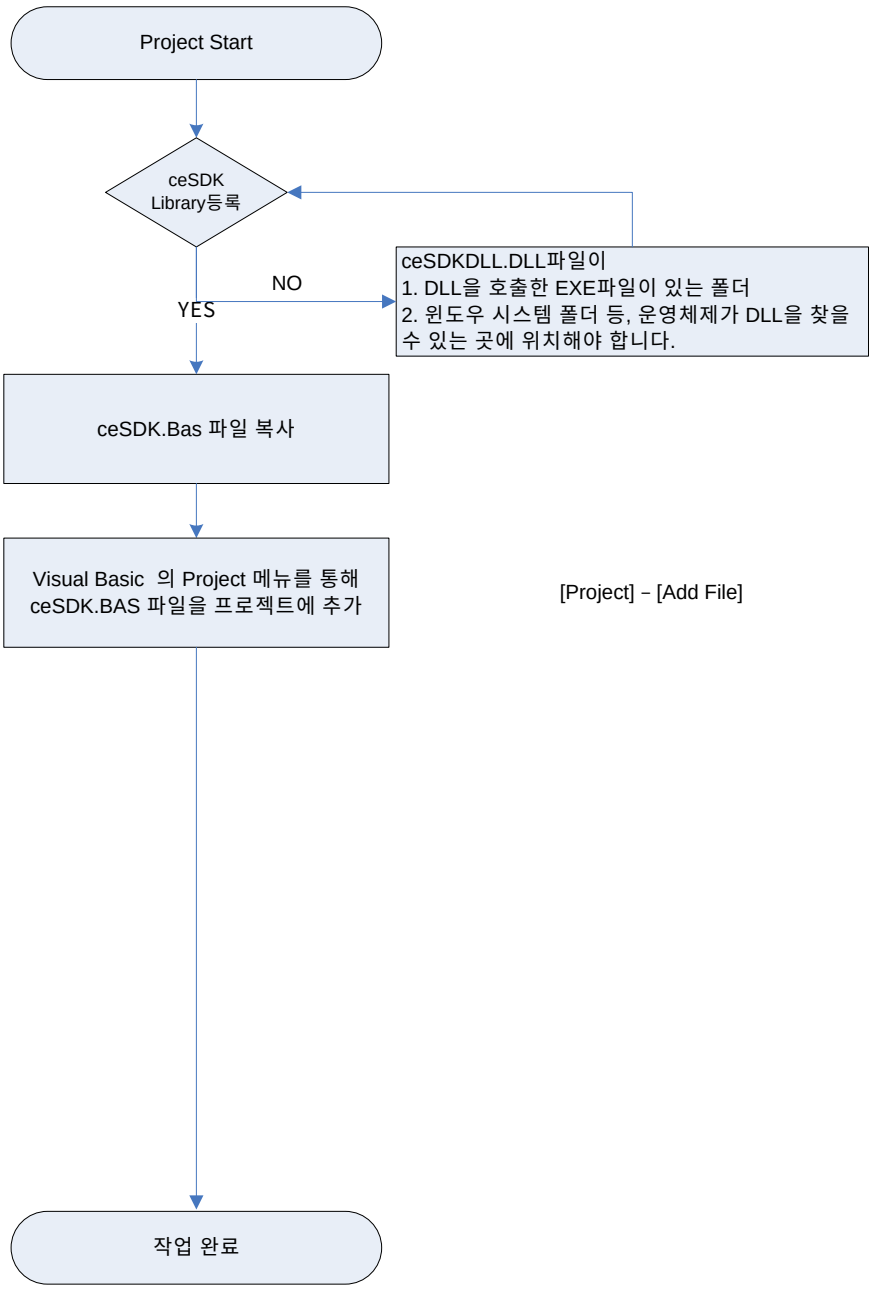


그림 2-60 Visual Basic 에서 ceSDK 사용 순서도

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

그림 2-61 새로운 프로젝트 생성

안내	Visual Basic 에 대한 ceSDK 지원 사항은 어떠합니까?
	ceSDK 는 .NET 환경의 Visual Basic 까지 지원을 하고 있습니다. 특히 미리 구성된 ceSDK 인터페이스 파일은 간단히 프로젝트에 추가만 하시면, 바로 ceSDK 를 사용하실 수 있도록 도움을 드리고 있습니다. 만약, Visual Basic 사용 고객님께서 ceSDK 사용에 어려움이 있으시다면, 언제든지 저희 ㈜커미조아 고객 지원팀으로 연락주시기 바랍니다. 최선을 다해 지원해 드릴것을 약속드립니다.

신규 프로젝트 창에서 'Standard EXE' 를 통해 표준 응용프로그램 개발을 시작합니다.

[File] 메뉴의 [New Project] 항목을 통해 신규 프로젝트를 생성합니다.

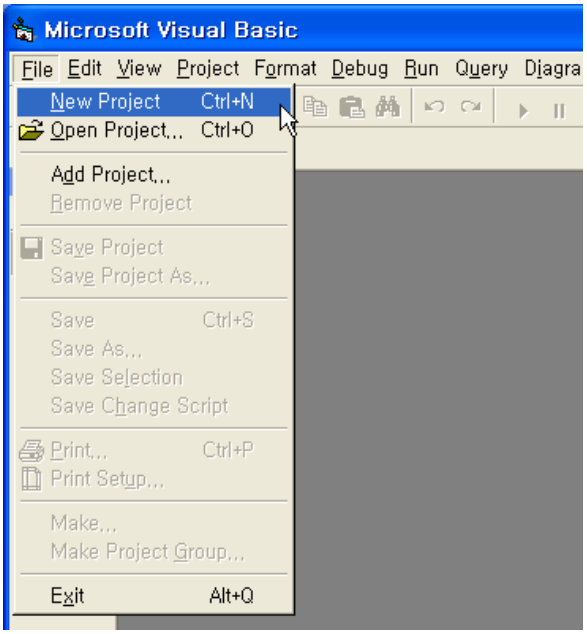


그림 2-62 신규 프로젝트의 시작

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

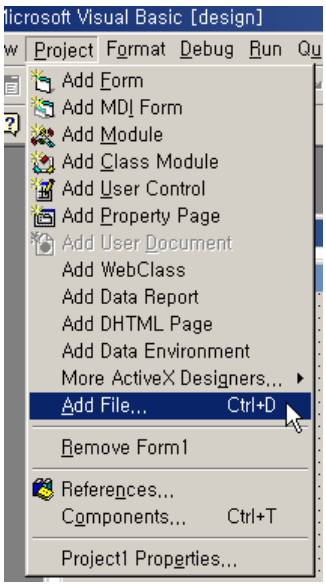


그림 2-63 프로젝트에 인터페이스 파일을 추가하기 위한 과정

그림 2-64 프로젝트에 추가 대상이 되는 ceSDK.BAS 파일

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

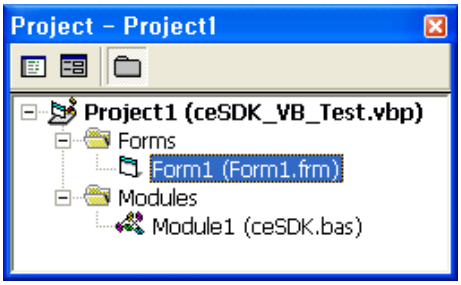


그림 2-65 프로젝트에 모듈로 추가된 ceSDK.BAS 파일

추가된 인터페이스 파일을 통해 다음과 같이 실제 응용프로그램 구현이 가능합니다.

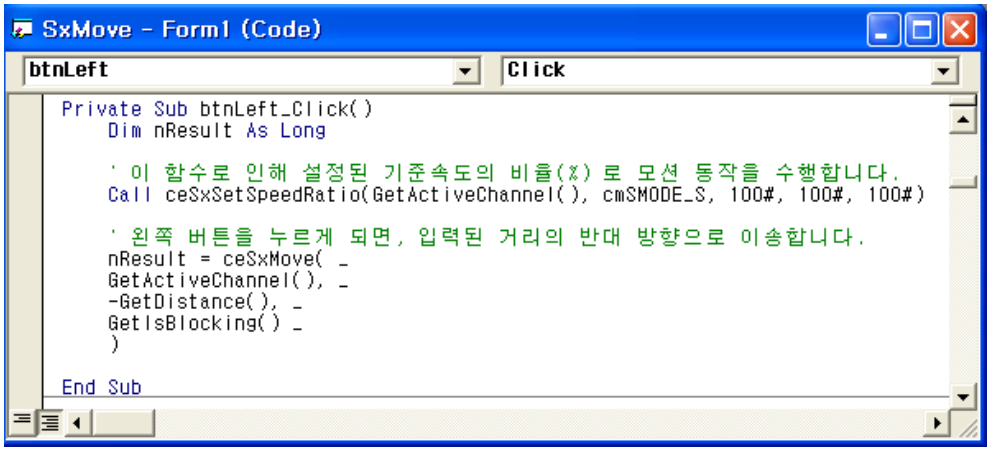


그림 2-66 ceSDK 를 통한 응용프로그램 구현

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

Chapter

3

ceSDK Introduction

다년간의 보다 강력하고 편리한 라이브러리 기술개발을 통해 자신 있게 제공하여 드리는ceSDK 는 편리한 함수명의 규칙을 통해 사용자 편의성을 극대화 하였습니다. ㈜커미조아ceSDK 의 최신 기능과 기술은 결코 흉내 낼 수 없는 커미조아의 기술입니다. 지금 확인 하십시오..

본 장에서는 ceSDK 가 제공하는 라이브러리 인터페이스에 대하여 자세한 설명 및 예를 수록하였습니다. ceSDK 는 라이브러리 기능을 보다 강력하고 효율적으로 지원할 수 있는 다양한 런타임(Run-time) 인터페이스와 라이브러리의 다양한 기능을 직관적으로 제공합니다. 본 매뉴얼에서는 ceSDK 에서 제공하는 라이브러리 함수에 대한 설명을 기능에 따라 그룹별로 수록하였습니다.

3 ceSDK Introduction

3.1 본 매뉴얼에 대해서

본 매뉴얼은 ㈜커미조아에서 제작, 배포하는 COMIZOA Ethernet/IP 통신 기반의 Motion, Digital I/O, Analog I/O, Counter 및 Serial 제어 프로그램 개발에 필요한 통신 라이브러리 (이하 ceSDK) 에 대한 인터페이스 함수에 대한 자세한 설명을 포함합니다.

3.2 함수의 명명 규칙

ceSDK 에서 제공하는 모든 함수는 다른 API 함수와 이름이 중복되는 것을 피하기 위하여 아래의 예와 같이 'ce-' 이라는 첨두어가 함수의 앞부분에 붙습니다.

ceGnLoad(), cemCfgMioProperty_Set(), cemSxMove(), cemHomeConfig_Set(), cediMulti_Get(), cedoMulti_Put(), ...

그리고 'ce-' 첨두어 바로 다음에는 해당 함수가 속하는 기능의 그룹을 대표하는 첨두어가 이어집니다. 이렇게 한 이유는 동일한 기능 그룹에 속한 함수들을 쉽게 구분할 수 있고, 함수가 리스트 될 때에 일반적으로 알파벳순으로 정렬되므로 동일 기능 그룹에 속한 함수들이 함께 리스트 될 수 있도록 하기 위함입니다.

다음은 몇 가지 기능 그룹을 대표하는 첨두어가 적용된 함수들의 예입니다.

◯ General Functions (Gn): ceGnLoad(), ceGnDebugMode(), ...

◯ Motion Cofiguration Functions (Cfg): cemCfgMioProperty_Set(), cemCfgSpeedPattern_Get(), ...

◯ Return to Home Functions (Home): cemHomeMove(), cemHomeSpeedPattern_Set(), ...

◯ Single Axis Move Functions (Sx): cemSxMoveTo(), cemSxStopEmg(), ...

◯ Interpolation Functions (Ix): cemIxLine(), cemIxSpeedPattern_Set(), ...

◯ Overriding Functions (Override): cemOverrideSpeedSet(), cemOverrideMove(), ...

◯ Master/Slave Functions (Ms): cemMsRegisterSlave(), cemMsCheckSlaveState(), ...

◯ Manual Pulsar Functions (Plsr): cemPlsrInMode_Set(), cemPlsrMove(), ...

◯ Motion States Functions (St): cemStSpeed_Get(), cemStPosition_Get(), ...

◯ Latch Functions (Ltc): cemLtcIsLatched(), cemLtcReadLatch(), ...

◯ Universal Digital I/O Functions (dio): cedioOne_Get(), cedioOne_Put(), ...

◯ Analog Input Functions (ai): ceaiVolt_Get(), ceaiCurrent_Get(), ...

◯ Analog Output Functions (ao): ceaoVolt_Out(), ceaoCurrent_Out(), ...

◯ Counter Functions (c): cec_Get(), cec_EnableOne_Set(), ...

◯ Serial Functions (s): ces_OpenPort(), ces_ReadByte(), ...

◯ Interlock Functions (s): ceil_Set(), ceil_ActionModeOne_Set(), ...

◯ Advanced Functions (Adv): cemAdvGetNodeInformation(), cemAdvManualPacket(), ...

◯ Utility Functions (utl): ceutlSyncWait(), ceutlPumpMultiMessage(), ...

3.3 데이터 형 표기

당사의 ceSDK 인터페이스는 매뉴얼에서 명시한 윈도우 표준 Dynamic Link Library 를 지원하는 어떠한 개발 환경에서도 사용 가능합니다. 하지만 데이터 형에 대한 이름은 개발 환경에 따라서 서로 다릅니다. 따라서 본 매뉴얼에서는 데이터 형의 형 표기를 아래와 같이 표기합니다. 이에 대한 각 컴파일러의 대응되는 데이터 형 표기는 아래 표를 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 값 을 전달 받는다는 것을 의미합니다.

Data type	Description	C/C++	VB 6.0	Delphi	C#
VT_EMPTY	반환 값이 없는 데이터 표현형	void	-	-	void
VT_HANDLE	운영 체제가 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 실수 변수의 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

3.4 함수 반환 값

ceSDK 에서는 일반적으로 함수 처리가 성공적이면 0 (ceERR_NONE) 을 반환합니다. 따라서 함수를 if 문의 조건 식으로 사용해서 오류를 범하는 실수는 주의하셔야 합니다.

◇ 잘못된 사용 예:

```
If ( ceGnLoad () )
{
        return error;       /* 이 경우 ceGnLoad() 함수 결과는 성공하더라도 습관적으로
                            if 문의 조건 식에 적용될 경우 원하지 않은 반대의 결과를 얻게 됩니다. */
}
```

◆ 올바른 사용 예:

```
If ( ceGnLoad () != ceERR_NONE )
{
        return error;
}
```

함수에 대한 에러 코드 리스트는 부록으로 제공되는 “에러 코드”를 참고하십시오.

3.5 사용자 개발 환경에 대한 안내

ceSDK Library 를 이용하여 사용자 어플리케이션 개발 시 일반적인 순서도

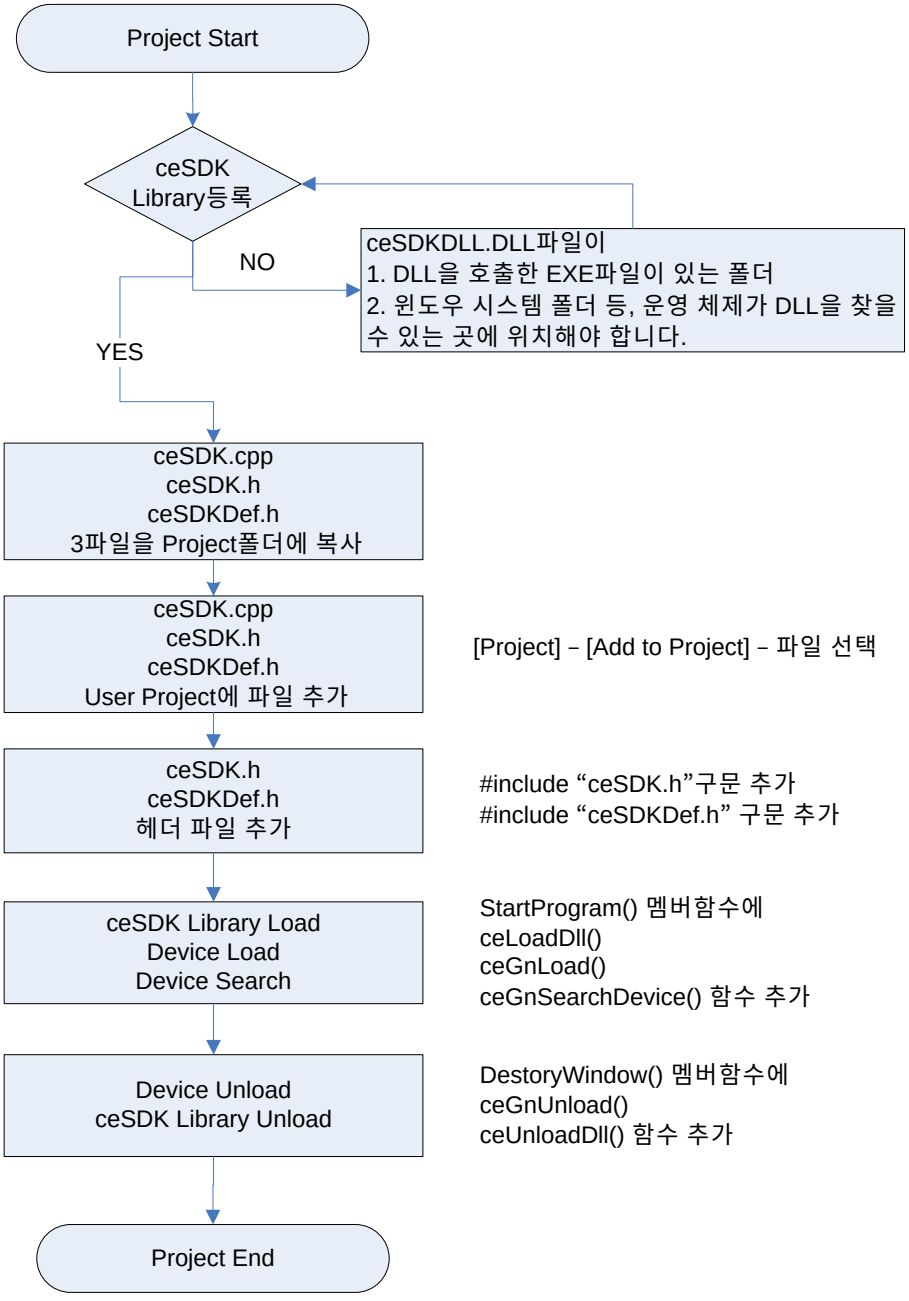


그림 3-1 ceSDK Library 를 이용한 사용자 어플리케이션 개발 시 일반적인 순서도

Chapter

4

General Functions

표준 함수와 관계된 ceSDK 라이브러리의 기능을 통해 가장 강력하고 편리한 통신 기반 제어를 하실 수 있습니다. 뛰어난 성능을 기반으로 한 ceSDK 라이브러리의 즐거움을 이제 함께 하십시오..

이 단원에서는 cEIP 장치의 초기화와 관련된 가장 일반적인 함수들을 소개합니다. 고객 여러분들께서는 라이브러리의 초기화와 사용을 위해 반드시 본 장을 읽어주시기 바랍니다. 구성되는 함수에는 ceSDK 라이브러리 로드 및 언로드, 원격 노드 탐색, 탐색 정보 반환 등에 대한 내용들로 구성되어 있습니다.

4 General Functions

4.1 함수 요약

"General Functions" 에 관련된 함수들은 다음과 같습니다.

Summary of Functions
r BOOL ceLoadDll ([none] VT_EMPTY) 라이브러리를 사용자 응용 프로그램의 사용을 위해 로드(Load) 합니다.
r VT_EMPTY ceUnloadDll ([none] VT_EMPTY) 라이브러리를 사용자 응용 프로그램의 사용 종료를 위해 언로드(Unload) 합니다.
r VT_I4 ceGnLoad ([none] VT_EMPTY) 라이브러리가 로드 된 상태에서 라이브러리 사용을 위한 구동 초기화 역할을 합니다.
r VT_I4 ceGnUnload ([none] VT_EMPTY) 라이브러리가 로드 된 상태에서 라이브러리 구동을 해제하는 역할을 합니다.
r VT_I4 ceGnSearchDevice ([in] VT_I4 RealNode, [in] DWORD nTimeout, [in] VT_I4 IsBlocking, [out] VT_PI4 pResultNode) 원격 노드를 탐색(Search)합니다.
r VT_I4 ceGnUnSearchDevice ([none] VT_EMPTY) 원격 노드 탐색 결과를 초기화 하며, 이 함수가 수행된 이후 ceGnSearchDevice 를 통해 원격 노드 정보를 탐색할 수 있습니다.
r VT_I4 ceGnReSearchDevice ([in] VT_I4 RealNode, [in] DWORD nTimeout, [in] VT_I4 IsBlocking, [out] VT_PI4 pResultNode) 원격 노드의 정보가 탐색된 상태에서, 원격 노드의 정보를 재 탐색(Research) 합니다.
r VT_I4 ceGnIsSearchedDevice ([out] VT_PI4 IsSearchedDevice) 원격 노드 탐색(Search)이 완료되어 있는지 확인합니다.
r VT_I4 ceGnResetNode ([in] VT_I4 NodeID, [in] VT_I4 ResetMode) 원격 노드의 모듈을 초기화(Reset)하는 역할을 수행합니다.
r VT_I4 ceGnCtrlBoost_Set ([in] VT_I4 BoostLevel, [in] VT_I4 BoostMode) ceSDK 서비스 프로세스 및 사용자 응용 프로그램의 프로세스 우선 순위를 제어합니다.
r VT_I4 ceGnCtrlBoost_Get ([out] VT_PI4 BoostLevel, [out] VT_PI4 BoostMode) ceSDK 서비스 프로세스 및 사용자 응용 프로그램의 프로세스 우선 순위를 반환합니다.
r VT_I4 ceGnNodeIsActive ([in] VT_I4 NodeID, [out] VT_PI4 IsActive) 원격 노드와 제어시스템간의 통신 연결 상태를 확인합니다.
r VT_I4 ceGnDebugMode ([in] VT_I4 DebugMode, [in] VT_PSTR szDebugFileName) 라이브러리의 기능을 사용하는 응용 프로그램에서 디버깅을 위한 환경을 설정합니다. 디버그 모드와 로그 파일명을 지정합니다.
r VT_I4 ceGnTotalNode ([out] VT_PI4 Node) 탐색된 전체 원격 노드의 개수를 반환합니다.
r VT_I4 ceGnTotalMotionChannel ([out] VT_PI4 Channel) 탐색된 전체 모션 제어 축(Axis) 개수를 반환합니다.

r VT_I4 ceGnTotalDIOChannel ([out] VT_PI4 Channel) 탐색된 전체 디지털 I/O(Digital Input/Output) 채널 개수를 반환합니다.
r VT_I4 ceGnTotalAIChannel ([out] VT_PI4 Channel) 탐색된 전체 아날로그 입력(Analog Input) 채널 개수를 반환합니다.
r VT_I4 ceGnTotalAOChannel ([out] VT_PI4 Channel) 탐색된 전체 아날로그 출력(Analog Ooutput) 채널 개수를 반환합니다.
r VT_I4 ceGnTotalMDIOChannel ([out] VT_PI4 Channel) 탐색된 전체 모션 디지털 I/O(Motion Digital I/O) 채널 개수를 반환합니다.
r VT_I4 ceGnTotalCNTChannel ([out] VT_PI4 Channel) 탐색된 전체 카운터(Counter) 채널 개수를 반환합니다.
r VT_I4 ceGnTotalSERChannel ([out] VT_PI4 Channel) 탐색된 전체 시리얼(Serial) 통신 채널 개수를 반환합니다.
r VT_I4 ceGnModuleCount_Motion ([in] VT_I4 NodeID, [out] VT_PI4 ModuleCount) 해당 원격 노드에 대해 탐색된 모션 모듈(Motion Module)의 개수를 반환합니다.
r VT_I4 ceGnModuleCount_Dio ([in] VT_I4 NodeID, [out] VT_PI4 ModuleCount) 해당 원격 노드에 대해 탐색된 디지털 I/O 모듈(Digital Input/Output Module)의 개수를 반환합니다.
r VT_I4 ceGnModuleCount_Ai ([in] VT_I4 NodeID, [out] VT_PI4 ModuleCount) 해당 원격 노드에 대해 탐색된 아날로그 입력 모듈(Analog Input Module)의 개수를 반환합니다.
r VT_I4 ceGnModuleCount_Ao ([in] VT_I4 NodeID, [out] VT_PI4 ModuleCount) 해당 원격 노드에 대해 탐색된 아날로그 출력 모듈(Analog Output Module)의 개수를 반환합니다.
r VT_I4 ceGnModuleCount_Mdio ([in] VT_I4 NodeID, [out] VT_PI4 ModuleCount) 해당 원격 노드에 대해 탐색된 모션 디지털 I/O 채널(Motion Digital Input/Output Channel)이 속한 모션 모듈의 개수를 반환합니다.
r VT_I4 ceGnModuleCount_Cnt ([in] VT_I4 NodeID, [out] VT_PI4 ModuleCount) 해당 원격 노드에 대해 탐색된 카운터 모듈(Counter Module)의 개수를 반환합니다.
r VT_I4 ceGnModuleCount_Ser ([in] VT_I4 NodeID, [out] VT_PI4 ModuleCount) 해당 원격 노드에 대해 탐색된 시리얼 통신 모듈(Serial Communication Module)의 개수를 반환합니다.
r VT_I4 ceGnChannelCount_Motion ([in] VT_I4 NodeID, [in] VT_I4 ModuleIdx, [out] VT_PI4 ChannelCount) 해당 원격 노드의 모션 모듈에 대해 탐색된 모션 제어 축(Axis) 개수를 반환합니다.
r VT_I4 ceGnChannelCount_Dio ([in] VT_I4 NodeID, [in] VT_I4 ModuleIdx, [out] VT_PI4 ChannelCount) 해당 원격 노드의 디지털 I/O 모듈에 대해 탐색된 DIO 채널(DIO Channel) 개수를 반환합니다.
r VT_I4 ceGnChannelCount_Ai ([in] VT_I4 NodeID, [in] VT_I4 ModuleIdx, [out] VT_PI4 ChannelCount) 해당 원격 노드의 아날로그 입력 모듈에 대해 탐색된 AI 채널(AI Channel) 개수를 반환합니다.
r VT_I4 ceGnChannelCount_Ao ([in] VT_I4 NodeID, [in] VT_I4 ModuleIdx, [out] VT_PI4 ChannelCount) 해당 원격 노드의 아날로그 출력 모듈에 대해 AO 채널(AO Channel) 개수를 반환합니다.
r VT_I4 ceGnChannelCount_Mdio ([in] VT_I4 NodeID, [in] VT_I4 ModuleIdx, [out] VT_PI4 ChannelCount) 해당 원격 노드의 모션 모듈에 대해 탐색된 모션 디지털 I/O 채널(MDIO Channel) 개수를 반환합니다.

r VT_I4 ceGnChannelCount_Cnt ([in] VT_I4 NodeID, [in] VT_I4 ModuleIdx, [out] VT_PI4 ChannelCount) 해당 원격 노드의 카운터 모듈에 대해 탐색된 카운터 채널(Counter Channel) 개수를 반환합니다.
r VT_I4 ceGnChannelCount_Ser ([in] VT_I4 NodeID, [in] VT_I4 ModuleIdx, [out] VT_PI4 ChannelCount) 해당 원격 노드의 시리얼 통신 모듈에 대해 탐색된 시리얼 통신 채널 개수를 반환합니다.
r VT_I4 ceGnLocalAxis_Get ([in] VT_I4 Axis, [out] VT_PI4 NodeIP, [out] VT_PI4 NodeID, [out] VT_PI4 NodeInGlobal, [out] VT_PI4 ModuleIdx, [out] PI4 ModuleInCh) 통합 모션 제어 축(Axis)에 대한 원격 노드 및 모션 모듈 정보를 반환합니다.
r VT_I4 ceGnLocalDIO_Get ([in] VT_I4 Channel, [out] VT_PI4 NodeIP, [out] VT_PI4 NodeID, [out] VT_PI4 NodeInGlobal, [out] VT_PI4 ModuleIdx, [out] PI4 ModuleInCh) 통합 디지털 I/O(DIO) 채널에 대한 원격 노드 및 DIO 모듈 정보를 반환합니다.
r VT_I4 ceGnLocalAI_Get ([in] VT_I4 Channel, [out] VT_PI4 NodeIP, [out] VT_PI4 NodeID, [out] VT_PI4 NodeInGlobal, [out] VT_PI4 ModuleIdx, [out] PI4 ModuleInCh) 통합 아날로그 입력(AI) 채널에 대한 원격 노드 및 AI 모듈 정보를 반환합니다.
r VT_I4 ceGnLocalAO_Get ([in] VT_I4 Channel, [out] VT_PI4 NodeIP, [out] VT_PI4 NodeID, [out] VT_PI4 NodeInGlobal, [out] VT_PI4 ModuleIdx, [out] PI4 ModuleInCh) 통합 아날로그 출력(AO) 채널에 대한 원격 노드 및 AO 모듈 정보를 반환합니다.
r VT_I4 ceGnLocaMlDIO_Get ([in] VT_I4 Channel, [out] VT_PI4 NodeIP, [out] VT_PI4 NodeID, [out] VT_PI4 NodeInGlobal, [out] VT_PI4 ModuleIdx, [out] PI4 ModuleInCh) 통합 모션 디지털 I/O(MDIO) 채널에 대한 원격 노드 및 모션 모듈 정보를 반환합니다.
r VT_I4 ceGnLocalCNT_Get ([in] VT_I4 Channel, [out] VT_PI4 NodeIP, [out] VT_PI4 NodeID, [out] VT_PI4 NodeInGlobal, [out] VT_PI4 ModuleIdx, [out] PI4 ModuleInCh) 통합 카운터(Counter) 채널에 대한 원격 노드 및 카운터 모듈 정보를 반환합니다.
r VT_I4 ceGnLocalSER_Get ([in] VT_I4 Channel, [out] VT_PI4 NodeIP, [out] VT_PI4 NodeID, [out] VT_PI4 NodeInGlobal, [out] VT_PI4 ModuleIdx, [out] PI4 ModuleInCh) 통합 시리얼 통신 채널에 대한 원격 노드 및 시리얼 통신 모듈 정보를 반환합니다.
r VT_I4 ceGnGlobalAxis_Get ([in] VT_I4 NodeID, [in] VT_I4 ModuleIdx, [in] VT_I4 ModuleInCh, [out] VT_PI4 GlobalAxis) 원격 노드 및 모션 모듈, 모듈 내의 축 번호를 통하여 전체 원격 노드의 통합 모션 제어 축 번호(Global Axis Number)를 반환합니다.
r VT_I4 ceGnGlobalDIO_Get ([in] VT_I4 NodeID, [in] VT_I4 ModuleIdx, [in] VT_I4 ModuleInCh, [out] VT_PI4 GlobalDIO) 원격 노드 및 디지털 I/O 모듈, 모듈 내의 DIO 채널 번호를 통하여 전체 원격 노드의 통합 DIO 채널 번호(Global DIO Channel Number)를 반환합니다.
r VT_I4 ceGnGlobalAI_Get ([in] VT_I4 NodeID, [in] VT_I4 ModuleIdx, [in] VT_I4 ModuleInCh, [out] VT_PI4 GlobalAI) 원격 노드 및 아날로그 입력 모듈, 모듈 내의 AI 채널 번호를 통하여 전체 원격 노드의 통합 AI 채널 번호(Global AI Channel Number)를 반환합니다.
r VT_I4 ceGnGlobalAO_Get ([in] VT_I4 NodeID, [in] VT_I4 ModuleIdx, [in] VT_I4 ModuleInCh, [out] VT_PI4 GlobalAO) 원격 노드 및 아날로그 출력 모듈, 모듈 내의 AO 채널 번호를 통하여 전체 원격 노드의 통합 AO 채널 번호(Global AO Channel Number)를 반환합니다.

r VT_I4 ceGnGlobalMDIO_Get ([in] VT_I4 NodeID, [in] VT_I4 ModuleIdx, [in] VT_I4 ModuleInCh, [out] VT_PI4 GlobalMDIO) 원격 노드 및 모션 모듈, 모듈 내의 MDIO 채널 번호를 통하여 전체 원격 노드의 통합 MDIO 채널 번호(Global MDIO Channel Number)를 반환합니다.
r VT_I4 ceGnGlobalCNT_Get ([in] VT_I4 NodeID, [in] VT_I4 ModuleIdx, [in] VT_I4 ModuleInCh, [out] VT_PI4 GlobalCNT) 원격 노드 및 카운터 모듈, 모듈 내의 카운터 채널 번호를 통하여 전체 원격 노드의 통합 카운터 채널 번호(Global Counter Channel Number)를 반환합니다.
r VT_I4 ceGnGlobalSER_Get ([in] VT_I4 NodeID, [in] VT_I4 ModuleIdx, [in] VT_I4 ModuleInCh, [out] VT_PI4 GlobalSER) 원격 노드 및 시리얼 통신 모듈, 모듈 내의 시리얼 통신 채널 번호를 통하여 전체 원격 노드의 통합 시리얼 통신 채널 번호(Global Serial Communication Channel Number)를 반환합니다.
r VT_I4 ceGnEmergency_Set ([in] VT_I4 NodeID, [in] VT_I4 State) 원격 노드의 소프트웨어 Emergency 상황을 설정하거나 해제 합니다.
r VT_I4 ceGnEmergency_Get ([in] VT_I4 NodeID, [out] VT_PI4 State) 원격 노드의 소프트웨어 Emergency 설정 상태를 반환합니다.

4.2 함수 설명

NAME

ceLoadDll

- 라이브러리(Library) 로드

INFORMATION	
1	General Function
!	VC++ (6, 7, 8)/VB BCB/Delphi
:	Level 1
J	위험 요소 없음

SYNOPSIS

r BOOL ceLoadDll ([none] VT_EMPTY)

DESCRIPTION

ceSDK 를 고객님의 응용 프로그램의 메모리 공간으로 호출합니다. 이 의미는 함수가 호출되는 순간 라이브러리는 고객님의 프로그램 내부 함수처럼 호출할 수 있게 됩니다.

이 함수는 고객님의 전체 프로그램에서 ceSDK 를 사용하기 위한 수순으로서는 가장 먼저 호출되어야 합니다.

이 함수의 사용과 호출에 있어, Boland Delphi 나 Microsoft Visual Basic 에서는 묵시적으로 이 동작이 이루어지기 때문에 함수의 호출이 필요하지 않습니다.

RETURN VALUE

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

Value	Meaning
0 (CE_FALSE)	DLL 을 로드 하는데 실패하였음을 의미합니다.
1 (CE_TRUE)	DLL 을 성공적으로 로드 하였음을 의미합니다.

SEE ALSO

ceUnloadDll

EXAMPLE

C/C++

```
#include “ceSDK.h”
#include “ceSDKDef.h”

/* 라이브러리를 응용 프로그램의 메모리 공간으로 호출합니다. */
void StartProgram ( void )
{
    // 이 함수의 반환값은 DLL 로드의 성공 여부를 반환합니다.
    BOOL nIsLoaded = ceLoadDll ();

    if ( nIsLoaded == CE_FALSE )
    {
            OutputDebugString ( "Dll Load Fail" );
            /*OutputDebugString API 는 GUI 프로그램에서 문자열을 디버거에 보낼 수 있습니다.
            Borland C++ Builder 에서는 Debug Window 에서 Event Log 를 확인 할 수 있으며,
            MS VC++ (6, 7, 8)에서는 Debug Window 에서 확인 할 수 있습니다.*/
    }
    else
    {
            //Dll Load Success
    }

/* 라이브러리를 응용 프로그램의 메모리 공간에서 해제 합니다. */
void EndProgram ( void )
{
    // 이 함수의 반환값은 없습니다. 따라서 별도의 에러 처리는 필요하지 않습니다.
    ceUnloadDll ();
}
```

NAME

ceUnloadDll

- 라이브러리(Library) 로드 해제

INFORMATION	
1	General Function
!	VC++ (6, 7, 8)/VB
	BCB/Delphi
:	Level 1
J	위험 요소 없음

SYNOPSIS

r VT_EMPTY ceUnloadDll ([none] VT_EMPTY)

DESCRIPTION

ceSDK 를 고객님의 응용 프로그램의 메모리 공간에서 해제합니다. 이 의미는 함수가 호출되는 순간 라이브러리는 고객님의 응용 프로그램에서 제거됩니다.

이 함수는 고객님의 전체 프로그램에서 ceSDK 사용을 종료 하기 위한 수순으로서는 가장 나중에 호출되어야 합니다.

이 함수의 사용과 호출에 있어, Boland Delphi 나 Microsoft Visual Basic 에서는 묵시적으로 이 동작이 이루어질 수 있도록 구성되어 있기 때문에 함수의 호출이 필요하지 않습니다.

SEE ALSO

ceLoadDll

EXAMPLE

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

NAME

ceGnLoad

- 라이브러리 구동 초기화

INFORMATION	
1	General Function
!	VC++ (6, 7, 8)/VB
	BCB/Delphi
:	Level 1
J	위험 요소 없음

SYNOPSIS

r VT_I4 ceGnLoad ([none] VT_EMPTY)

DESCRIPTION

로드 된 ceSDK 라이브러리 구동을 초기화 합니다.

이 함수는 ceSDK 의 다른 함수가 호출되기 전에 반 드시 한번은 수행되어야 합니다. 일반적으로 프로그램의 시작 부분에서 ceLoadDll() 함수 호출 후 수행해 주시면 됩니다.

RETURN VALUE

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

SEE ALSO

ceGnUnload

EXAMPLE

C/C++

```
#include “ceSDK.h”
#include “ceSDKDef.h”

void ProgramInitial ( void )
{
    if ( ceLoadDll () != CE_TRUE )
    {
            OutputDebugString ( “Can’t load ceSDK Library(DLL)” );
    }
    else
    {
            // 로드된 ceSDK 라이브러리 구동을 초기화 합니다.
            if ( ceGnLoad () != ceERR_NONE )
            {
                    OutputDebugString ( “ceGnLoad has been failed” );
            }
    }
}


void ProgramEnd ( void )
{
    // 로드된 라이브러리 구동을 해제합니다.
    ceGnUnload ();

    // 라이브러리를 응용 프로그램 메모리 공간에서 해제합니다.
    ceUnloadDll ();
}
```

Visual Basic

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

Private Sub Form_Load ()

    Dim nRetVal As Long

    ‘ 로드된 ceSDK 라이브러리 구동을 초기화 합니다.
    nRetVal = ceGnLoad ()
    If nRetVal <> ceERR_NONE Then
            MsgBox ( “ceGnLoad has been failed” )
    End If

End Sub

Private Sub Form_Unload ( Cancel As Integer )

    ‘ 로드된 라이브러리 구동을 해제합니다.
    Call ceGnUnload ()

End Sub
```

Delphi

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

procedure TForm1.OnCreate ( Sender: TObject );
begin

    // 로드된 ceSDK 라이브러리 구동을 초기화 합니다.
    if ( ceGnLoad () <> ceERR_NONE ) then
    begin
            ShowMessage ( 'ceGnLoad has been failed' );
    end;
end;

procedure TForm1.OnDestroy ( Sender: TObject );
begin

    // 로드된 라이브러리 구동을 해제합니다.
    ceGnUnload ();

end;
```

NAME

ceGnUnload

- 라이브러리 구동 해제

INFORMATION	
1	General Function
!	VC++ (6, 7, 8)/VB
	BCB/Delphi
:	Level 1
J	위험 요소 없음

SYNOPSIS

r VT_I4 ceGnUnload ([none] VT_EMPTY)

DESCRIPTION

로드(Load) 된 라이브러리 구동을 해제합니다.

이 함수는 ceSDK 의 함수 사용을 종료하기 위해 명시 적으로 호출되어야 하며, 일반적으로 프로그램의 종료 부분에서 수행해 주면 됩니다.

RETURN VALUE

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

SEE ALSO

ceGnLoad

EXAMPLE

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

NAME

ceGnSearchDevice

- 원격 노드 탐색 (Search)

INFORMATION	
1	General Function
!	VC++ (6, 7, 8)/VB
	BCB/Delphi
:	Level 1
J	위험 요소 없음

SYNOPSIS

r VT_I4 ceGnSearchDevice ([in] VT_I4 RealNode, [in] DWORD nTimeout,
[in] VT_I4 IsBlocking, [out] VT_PI4 pResultNode)

DESCRIPTION

라이브러리 구동 초기화 상태에서 원격 노드의 정보를 탐색합니다. 일반적인 응용 프로그램의 메인 프로그램에서 이 함수를 통해 원격 노드를 탐색하고, 다른 프로그램에서는 이미 검색된 정보를 참조하게 됩니다.

PARAMETER

▶ ***RealNode*** : 실제 탐색(Search)할 원격 노드의 수를 설정합니다. RealNode 의 개수는 탐색할 원격 노드의 개수보다 같거나 커야 합니다.

▶ ***nTimeout*** : 원격 노드 탐색(Search) 시간을 밀리초(ms) 단위로 설정합니다. 1 개의 원격 노드를 탐색하는데 걸리는 최소 시간은 30ms 입니다. 최소 탐색 시간은 다음과 같습니다.

원격 노드 개수	탐색 시간
1 개	30ms * 1 = 30ms
2 개	30ms * 2 = 60ms
3 개	30ms * 3 = 90ms

실제 탐색 시간이 너무 짧을 경우 요구되는 모든 원격 노드를 검색할 수 없으니, 탐색 시간을 충분히 설정하여 주시기 바랍니다.

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

Value	Meaning
0 (CE_FALSE)	블록(Blocking)을 하지 않습니다. 따라서 원격 노드 정보 탐색이 완료되는 동안에도 윈도우 이벤트를 처리합니다.
1 (CE_TRUE)	블록(Blocking)을 합니다. 따라서 원격 노드 정보 탐색이 완료되는 동안에는 윈도우 이벤트가 처리되지 않습니다.

▶ *pResultNode* : 이 매개변수를 통하여 탐색된 원격 노드의 개수를 반환합니다.

RETURN VALUE

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

㈜ 커미조아의 디바이스 초기값은 어떤 값으로 설정됩니까?

ceGnSearchDevice() 함수를 통해 탐색한 원격 노드 초기화 시 각 속성별로 사용되는 기본값은 'Appendix::C Motion Default Parameter'를 참조하시기 바랍니다.

SEE ALSO

ceGnReSearchDevice, ceGnUnSearchDevice, ceGnIsSearchedDevice

EXAMPLE

C/C++

```
#include “ceSDK.h”
#include “ceSDKDef.h”

#define REALNODE            10
#define TIMEOUT             1000

void ProgramInitial ( void )
{
    if ( ceLoadDll () != CE_TRUE )
    {
            OutputDebugString ( “Can’t load ceSDK Library(DLL)” );
            return;
    }

    // 로드된 ceSDK 라이브러리 구동을 초기화 합니다.
    if ( ceGnLoad () != ceERR_NONE )
    {
            OutputDebugString ( " ceGnLoad has been failed" );
            return;
    }


    long nIsSearchedDevice;             // 탐색된 원격 노드 정보
    long nNodeCount;                    // 원격 노드 탐색 여부 정보
    long nRetVal;

    // 원격 노드 탐색 여부를 확인 합니다.
    ceGnIsSearchedDevice ( &nIsSearchedDevice );

    // 원격 노드 탐색 여부를 확인하여 원격 노드의 정보를 탐색합니다.
    if ( nIsSearchedDevice == CE_FALSE )        // 원격 노드 탐색을 수행하지 않았습니다.
    {
            nRetVal = ceGnSearchDevice ( REALNODE,          // 탐색할 원격 노드 개수
                                         TIMEOUT,           // 노드 탐색 시간 (ms)
                                         CE_FALSE,          // 메시지 블록 여부
                                         &nNodeCount        // 탐색된 원격 노드 수 반환
                                        );

            if ( nRetVal != ceERR_NONE )
            {
                    // 적절한 에러 처리를 수행하여 주시기 바랍니다.
            }
    }
    else    // 원격 노드 탐색이 수행되었습니다.
    {
            if (ceGnReSearchDevice ( REALNODE, TIMEOUT, CE_FALE, &nNodeCount ) != ceERR_NONE )
            {
                    // 적절한 에러 처리를 수행하여 주시기 바랍니다.
            }
    }
}
```

Visual Basic

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

Private Sub Form_Load ()

    Dim nNodeCount As Long              ' 탐색된 원격 노드 정보
    Dim nIsSearchedDevice As Long       ' 원격 노드 탐색 여부 정보
    Dim nRetVal As Long

    ' 로드된 ceSDK 라이브러리 구동을 초기화 합니다.
    nRetVal = ceGnLoad ()

    If nRetVal <> ceERR_NONE Then
            MsgBox ( "ceGnLoad has been failed" )
    End If


    ' 원격 노드 탐색 여부를 확인합니다.
    Call ceGnIsSearchedDevice ( nIsSearchedDevice )

    ' 원격 노드 탐색 여부를 확인하여 원격 노드의 정보를 탐색합니다.
    If nIsSearchedDevice = CE_FALSE Then     ' 원격 노드 탐색을 수행하지 않았습니다.
            nRetVal = ceGnSearchDevice ( 10, 1000, CE_FALSE, nNodeCount )

            If nRetVal <> ceERR_NONE Then
                    ' 적절한 에러 처리를 수행하여 주시기 바랍니다.
            End If

    Else    ' 원격 노드 탐색을 수행하였습니다.
            nRetVal = ceGnReSearchDevice ( 10, 1000, CE_FALSE, nNodeCount )

            If nRetVal <> ceERR_NONE Then
                    ' 적절한 에러 처리를 수행하여 주시기 바랍니다.
            End If
    End If

End Sub
```

Delphi

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

procedure TForm1.OnCreate ( Sender: TObject );
var
    nNodeCount : LongInt;               // 탐색된 원격 노드 정보
    nIsSearchedDevice : LongInt;        // 원격 노드 탐색 여부 정보

begin
```

```
    // 로드된 ceSDK 라이브러리 구동을 초기화 합니다.
    if ( ceGnLoad () <> ceERR_NONE ) then
    begin
            ShowMessage ( 'ceGnLoad has been failed' );
    end;

    // 원격 노드 탐색 여부를 확인합니다.
    ceGnIsSearchedDevice ( @nIsSearchedDevice );

    // 원격 노드 탐색 여부를 확인하여 원격 노드의 정보를 탐색합니다.
    if nIsSearchedDevice = CE_FALSE then
    begin
            ceGnSearchDevice ( 10, 1000, CE_FALSE, @nNodeCount );
    else
            ceGnReSearchDevice ( 10, 1000, CE_FALSE, @nNodeCount );
    end;

end;
```

NAME

ceGnUnSearchDevice

- 원격 노드 탐색 결과 초기화

INFORMATION	
1	General Function
!	VC++ (6, 7, 8)/VB
	BCB/Delphi
:	Level 1
J	위험 요소 없음

SYNOPSIS

r VT_I4 ceGnUnSearchDevice ([none] VT_EMPTY)

DESCRIPTION

원격 노드 탐색 결과를 초기화 하며, 이 함수가 수행된 이후 ceGnSearchDevice 함수를 통해 원격 노드 탐색을 수행할 수 있습니다.

REFERENCE

원격 노드를 재 탐색할 필요가 있을 경우 명시적으로 본 함수를 호출하지 않고, ceGnReSearchDevice 함수를 통해 재 탐색 하시면 됩니다.

ceGnIsSearched 함수를 통해 원격 노드가 탐색된 상태인지 확인할 수 있습니다. 원격 노드 탐색 전이면 ceGnSearchDevice 함수를, 원격 노드 탐색 후라면 ceGnReSearchDevice 함수를 사용하시면 됩니다.

이 함수는 원격 노드의 탐색 결과를 초기화 하는 함수입니다. 원격 노드 정보 초기화는 다른 cEIP 응용 프로그램에 영향을 주므로 ceGnReSearchDevice 함수를 통해 탐색 된 정보를 얻어오시기 바랍니다.

RETURN VALUE

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

SEE ALSO

ceGnSearchDevice, ceGnReSearchDevice, ceGnIsSearchedDevice

EXAMPLE

C/C++

```
#include “ceSDK.h”
#include “ceSDKDef.h”

#define REALNODE        10
#define TIMEOUT         1000

long nIsSearchedDevice;     // 탐색된 원격 노드 정보
long nNodeCount;            // 원격 노드 탐색 여부 정보
long nRetVal;


// 원격 노드 탐색 여부를 확인 합니다.
ceGnIsSearchedDevice ( &nIsSearchedDevice );

// 원격 노드 탐색 여부를 확인하여 원격 노드의 정보를 탐색합니다.
if ( nIsSearchedDevice == CE_TRUE ) // 원격 노드 탐색이 수행 되었습니다.
{
    ceGnUnSearchDevice ();   // 원격 노드 탐색 결과를 초기화 합니다.
    /* 이 함수는 원격 노드의 탐색 결과를 초기화 하는 함수입니다. 이 함수 호출 시 다른 cEIP
    응용 프로그램에 영향을 주므로, 원격 노드를 재 탐색할 필요가 있을 경우 명시적으로 본 함수를
    호출하지 않고, ceGnReSearchDevice 함수를 통해 재 탐색 하여 주시기 바랍니다. */

    if (ceGnSearchDevice ( REALNODE, TIMEOUT, CE_FALE, &nNodeCount ) != ceERR_NONE )
    {
            // 적절한 에러 처리를 수행하여 주시기 바랍니다.
    }
}
```

Visual Basic

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

Dim nNodeCount As Long          ‘ 탐색된 원격 노드 정보
Dim nIsSearchedDevice As Long   ‘ 원격 노드 탐색 여부 정보
Dim nRetVal As Long

‘ 원격 노드 탐색 여부를 확인합니다.
Call ceGnIsSearchedDevice ( nIsSearchedDevice )

‘ 원격 노드 탐색 여부를 확인하여 원격 노드의 정보를 탐색합니다.
If nIsSearchedDevice = CE_TRUE Then     ‘ 원격 노드 탐색을 수행하였습니다.
    Call ceGnUnSearchDevice ()          ‘ 원격 노드 탐색 결과를 초기화 합니다.

    nRetVal = ceGnSearchDevice ( 10, 1000, CE_FALSE, nNodeCount )

    If nRetVal <> ceERR_NONE Then
```

```
            ' 적절한 에러 처리를 수행하여 주시기 바랍니다.
    End If

End If
```

Delphi

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

var
    nNodeCount : LongInt;                // 탐색된 원격 노드 정보
    nIsSearchedDevice : LongInt;         // 원격 노드 탐색 여부 정보

begin

    // 원격 노드 탐색 여부를 확인합니다.
    ceGnIsSearchedDevice ( @nIsSearchedDevice );

    // 원격 노드 탐색 여부를 확인하여 원격 노드의 정보를 탐색합니다.
    if nIsSearchedDevice = CE_TRUE then        // 원격 노드 탐색이 수행되었습니다.
    begin
            ceGnUnSearchDevice ();   // 원격 노드 탐색 결과를 초기화 합니다.
            ceGnSearchDevice ( 10, 1000, CE_FALSE, @nNodeCount );
    end;

end;
```

NAME

ceGnReSearchDevice

- 원격 노드 재 탐색 (Research)

INFORMATION	
1	General Function
!	VC++ (6, 7, 8)/VB
	BCB/Delphi
:	Level 1
J	위험 요소 없음

SYNOPSIS

r VT_I4 ceGnReSearchDevice ([in] VT_I4 RealNode, [in] DWORD nTimeout,

[in] VT_I4 IsBlocking, [out] VT_PI4 pResultNode)

DESCRIPTION

원격 노드의 정보가 탐색된 상태에서 원격 노드의 정보를 재 탐색 합니다. ceGnIsSearchedDevice 함수를 통해 원격 노드 탐색 여부를 확인할 수 있습니다.

PARAMETER

▶ ***RealNode*** : 실제 탐색(Search)할 원격 노드의 수를 설정합니다. RealNode 의 개수는 탐색할 원격 노드의 개수보다 같거나 커야 합니다.

▶ ***nTimeout*** : 원격 노드 탐색(Search) 시간을 밀리초(ms) 단위로 설정합니다. 1 개의 원격 노드를 탐색하는데 걸리는 최소 시간은 30ms 입니다. 최소 탐색 시간은 다음과 같습니다.

원격 노드 개수	탐색 시간
1 개	30ms * 1 = 30ms
2 개	30ms * 2 = 60ms
3 개	30ms * 3 = 90ms

실제 탐색 시간이 너무 짧을 경우 요구되는 모든 원격 노드를 검색할 수 없으니, 탐색 시간을 충분히 설정해 주시기 바랍니다.

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

Value	Meaning
0 (CE_FALSE)	블록(Blocking)을 하지 않습니다. 따라서 원격 노드 정보 탐색이 완료되는 동안에도 윈도우 이벤트를 처리합니다.
1 (CE_TRUE)	블록(Blocking)을 합니다. 따라서 원격 노드 정보 탐색이 완료되는 동안에는 윈도우 이벤트가 처리되지 않습니다.

▶ ***pResultNode*** : 이 매개변수를 통하여 탐색된 원격 노드의 수를 반환합니다.

RETURN VALUE

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

㈜ 커미조아의 디바이스 초기값은 어떤 값으로 설정됩니까?

ceGnSearchDevice() 함수를 통해 탐색한 원격 노드 초기화 시 각 속성별로 사용되는 기본값은 "Appendix::C Motion Default Parameter"를 참조하시기 바랍니다.

SEE ALSO

ceGnSearchDevice, ceGnUnSearchDevice, ceGnIsSearchedDevice

EXAMPLE

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

NAME

ceGnIsSearchedDevice

- 원격 노드 정보 탐색(Search) 완료 여부 확인

INFORMATION	
1	General Function
!	VC++ (6, 7, 8)/VB
	BCB/Delphi
:	Level 1
J	위험 요소 없음

SYNOPSIS

r VT_I4 ceGnIsSearchedDevice ([out] VT_PI4 IsSearchedDevice)

DESCRIPTION

ceGnIsSearchedDevice 함수를 통하여 원격 노드 탐색 완료 여부를 IsSearchedDevice 매개변수를 통해 확인합니다. 다른 프로세스에서 이미 원격 노드 정보가 탐색되었다면, 이 함수를 통해 이미 탐색된 원격 노드를 대상으로 재 탐색을 하거나 하지 않도록 할 수 있습니다.

원격 노드 정보 탐색 완료 여부에 따라, 원격 노드 탐색 전이라면 ceGnSearchDevice 함수를, 원격 노드 탐색 후라면 ceGnReSearchDevice 함수를 통해 원격 노드 정보를 탐색하여 주시기 바랍니다.

PARAMETER

▶ *IsSearchedDevice* : 이 매개변수를 통하여 원격 노드 정보 탐색 여부를 확인합니다.

Value	Meaning
0 (CE_FALSE)	원격 노드 탐색을 수행하지 않았습니다. ceGnSearchDevice 함수를 통하여 원격 노드 정보 탐색을 수행하시면 됩니다.
1 (CE_TRUE)	원격 노드 탐색을 수행하였습니다. ceGnReSearchDevice 함수를 통하여 원격 노드 정보 재 탐색을 수행하시면 됩니다.

RETURN VALUE

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

SEE ALSO

ceGnSearchDevice, ceGnReSearchDevice, ceGnUnSearchDevice

EXAMPLE

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

NAME

ceGnResetNode

- 원격 노드 모듈 상태 초기화

INFORMATION	
1	General Function
!	VC++ (6, 7, 8)/VB
	BCB/Delphi
:	Level 1
J	위험 요소 없음

SYNOPSIS

r VT_I4 ceGnResetNode ([in] VT_I4 NodeID, [in] VT_I4 ResetMode)

DESCRIPTION

해당 원격 노드 내의 모듈을 초기화하는 함수입니다. 지정된 초기화 모드를 통하여 모듈을 선택적으로, 또는 전체 모듈을 초기화 할 수 있습니다.

REFERENCE

원격 노드 모듈 상태 초기화 명령이 수행된 후 설정되는 각 속성별로 사용되는 기본값은 'Appendix::C Default Parameter' 를 참조하여 주시기 바랍니다.

PARAMETER

▶ ***NodeID*** : 원격 노드 ID. 초기화할 모듈이 속한 원격 노드 ID 를 설정합니다.

▶ ***ResetMode*** : 초기화 모드. 초기화 모드의 의미는 다음과 같습니다.

Value	Meaning
1 (CE_RESET_DIO)	Digital Input/Output Module Reset.
2 (CE_RESET_MOTION)	Motion Module Reset.
4 (CE_RESET_AIO)	Analog Input, Analog Output Module Reset.
7 (CE_RESET_ALL)	Digital I/O, Motion, AI, AO Module Reset.

RETURN VALUE

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

EXAMPLE

C/C++

```
#include “ceSDK.h”
#include “ceSDKDef.h”

void OnResetNode ()
{
    long nNodeID = 1;            // 초기화할 모듈이 속한 원격 노드 ID 를 지정합니다.

    /* Reset Mode */
    // CE_RESET_DIO : Digital Input / Output Modue Reset
    // CE_RESET_MOTION : Motion Module Reset
    // CE_RESET_AIO : Analog Input / Analog Output Module Reset
    // CE_RESET_ALL : DIO, Motion, AI, AO Module Reset

    // 대상 원격 노드의 Motion, DIO, AI, AO 모듈 모두를 리셋 합니다.
    ceGnResetNode( nNodeID,  CE_RESET_ALL );
}
```

Visual Basic

```
Private Sub OnResetNode ()

    Dim nNodeID As Long       ‘초기화할 모듈이 속한 원격 노드 ID 를 지정합니다.

    nNodeID = 1

    ‘ 대상 원격 노드의 Motion, DIO, AI, AO 모듈 모두를 리셋 합니다.
    Call ceGnResetNode ( nNodeID, CE_RESET_ALL )

End Sub
```

Delphi

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

procedure OnResetNode ();
var
    nNodeID : LongInt;                    // 탐색된 원격 노드 정보

begin
    nNodeID := 1;

    // 대상 원격 노드의 Motion, DIO, AI, AO 모듈 모두를 리셋 합니다.
```

```
    ceGnResetNode ( nNodeID, CE_RESET_ALL );
end;
```

NAME

ceGnCtrlBoost_Set / ceGnCtrlBoost_Set
- ceSDK 서비스 프로세스 및 사용자 응용 프로그램의 프로세스 우선 순위 제어 및 반환

INFORMATION	
1	General Function
!	VC++ (6, 7, 8)/VB
	BCB/Delphi
:	Level 9
L	주 의

SYNOPSIS

r VT_I4 ceGnCtrlBoost_Set ([in] VT_I4 BoostLevel, [in] VT_I4 BoostMode)

r VT_I4 ceGnCtrlBoost_Get ([out] VT_PI4 BoostLevel, [out] VT_PI4 BoostMode)

DESCRIPTION

윈도우 기반 응용 프로그램에서 프로세스는 시스템의 CPU 실행 시간을 라운드 로빈(Round Robin) 방식으로 나누어 실행하게 됩니다. ceGnCtrlBoost_Set 함수는 이 시간에서 우선 순위를 설정하여, 좀 더 많은 CPU 의 시간을 필요로 하고, 빠른 응답 성을 보이기 위한 라이브러리 기능입니다.

PARAMETER

▶ ***BoostLevel*** : 프로세스 우선순위를 설정 혹은 설정되어 있는 프로세스 우선순위를 반환합니다. BoostLevel 의 우선 순위는 다음과 같습니다.

1. CE_ABOVE_NORMAL_PRIORITY_CLASS
2. CE_BELOW_NORMAL_PRIORITY_CLASS
3. CE_HIGH_PRIORITY_CLASS[Default]
4. CE_IDLE_PRIORITY_CLASS
5. CE_NORMAL_PRIORITY_CLASS
6. CE_PROCESS_MODE_BACKGROUND_BEGIN
7. CE_PROCESS_MODE_BACKGROUND_END
8. CE_REALTIME_PRIORITY_CLASS

위 우선 순위는 시스템의 기반 우선 순위를 지칭합니다. 일반적인 시스템 프로세스가 가질 수 있는 우선 순위를 LEVEL 로 지정할 수 있습니다.

시스템은 프로세스의 반응 성을 높이면서도 다른 프로세스의 실행 시간을 지나치게 뺏지 않도록 가장 합리적이고 효율적인 방법으로 프로세스의 우선 순위를 높이거나 낮추는데 이를 우선 순위 부스트(Priority Boost)라고 합니다. 프로세스 우선 순위를 CE_REALTIME_PRIORITY_CLASS LEVEL 로 지정하면 고속의 디지털 입·출력이나, 실시간 성이 요구되는 동작에서 다른 윈도우 프로세스의 우선 순위에 영향을 받지 않을 수 있게 되며, CPU 의 시간을 독점할 수 있게 됩니다.

다만, 이러한 설정은 비전(Vision) 시스템이나 시스템 프로세스의 고속 응답성이 요구되는 다른 응용 프로그램을 동시에 실행시키게 될 경우 전체 시스템에서 심각한 성능 저하를 가져올 수 있으므로 상황에 맞는 프로세스 우선 순위를 설정하여야 합니다.

사용자 응용 프로그램의 프로세스 우선 순위는 다음과 같이 CE_NORMAL_PRIORITY_CLASS 이며, ceSDK 서비스 프로세스는 동작의 효율성과 빠른 응답성을 위해 CE_HIGH_PRIORITY_CLASS 입니다. 이 우선 순위는 본 API 호출을 통해 제어할 수 있으며, 동작에 이상이 없는 한 기본 모드를 유지를 권장합니다.

▶ ***BoostMode*** : 이 매개변수를 통하여 프로세스 우선순위를 적용 할 대상을 설정 혹은 반환합니다. BoostMode 의 의미는 다음과 같습니다.

Value	Meaning
0 (CE_PROCESS_ONLY_BOOST)	사용자 응용 프로그램의 프로세스 우선 순위를 설정.
1 (CE_SERVICE_ONLY_BOOST)	서비스 프로세스를 대상으로 우선 순위를 설정.
2 (CE_ALL_BOOST)	사용자 응용 프로그램과 서비스 프로세스를 대상으로 우선 순위를 설정.

RETURN VALUE

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

NAME

ceGnNodeIsActive

- 원격 노드와 제어시스템간의 통신 연결 상태 확인

INFORMATION	
1	General Function
!	VC++ (6, 7, 8)/VB
	BCB/Delphi
:	Level 1
J	위험 요소 없음

SYNOPSIS

r VT_I4 ceGnNodeIsActive ([in] VT_I4 NodeID, [out] VT_PI4 IsActive)

DESCRIPTION

원격 노드 탐색이 완료된 상태에서 해당 원격 노드와 제어 시스템과의 통신 연결 상태를 확인합니다.

PARAMETER

▶ *NodeID* : 원격 노드 ID. 통신 연결 상태를 확인 할 원격 노드의 ID 를 설정합니다.

▶ *IsActive* : 이 매개변수를 통하여 원격 노드와 제어 시스템간의 통신 연결 상태를 반환합니다.

Value	Meaning
0 (CE_FALSE)	원격 노드와 제어 시스템간의 통신이 단절되었음을 의미합니다.
1 (CE_TRUE)	원격 노드와 제어 시스템간의 통신이 정상적으로 연결되어 있음을 의미합니다.

RETURN VALUE

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

EXAMPLE

C/C++

```
#include “ceSDK.h”
#include “ceSDKDef.h”

void OnNodeIsActive ()
{
    long nNodeID = 1;          // 해당 원격 노드 ID
    long nIsActive;            // 통신 연결 상태 반환 변수

    // 대상 원격 노드와 제어 시스템과의 통신 연결 상태를 확인합니다.
    ceGnNodeIsActive ( nNodeID, &nIsActive );

    if ( nIsActive == CE_FALSE )
    {
            // 대상 원격 노드와 통신이 단절되었을 경우에 대한 처리
    }
}
```

Visual Basic

```
Private Sub OnNodeIsActive ()

    Dim nNodeID As Long      ‘ 해당 원격 노드 ID
    Dim nIsActive As Long    ‘ 통신 연결 상태 반환 변수

    ‘ 대상 원격 노드와 제어 시스템과의 통신 연결 상태를 확인합니다.
    ceGnNodeIsActive ( nNodeID, nIsActive )

    If nIsActive = CE_FALSE Then
            ‘ 대상 원격 노드와 통신이 단절되었을 경우에 대한 처리
    End If

End Sub
```

Delphi

```
procedure OnNodeIsActive ();
var
    nNodeID : LongInt;         // 해당 원격 노드 ID
    nIsActive : LongInt;       // 통신 연결 상태 반환 변수

begin
    nNodeID := 1;

    // 대상 원격 노드와 제어 시스템과의 통신 연결 상태를 확인합니다.
    ceGnNodeIsActive ( nNodeID, @nIsActive );
```

```
    if nIsActive = CE_FALSE then
    begin
            // 대상 원격 노드와 통신이 단절되었을 경우에 대한 처리
    end;
end;
```

NAME

ceGnDebugMode

- 디버그 모드 설정

INFORMATION	
1	General Function
!	VC++ (6, 7, 8)/VB
	BCB/Delphi
:	Level 1
J	위험 요소 없음

SYNOPSIS

r VT_I4 ceGnDebugMode ([in] VT_I4 DebugMode, [in] VT_PSTR szDebugFileName)

DESCRIPTION

구성 함수의 입력 매개변수(Parameter)와 출력 매개변수를 모니터링 하거나 라이브러리의 기능을 사용하는 응용 프로그램에서 고 수준의 디버깅(Debugging)을 위한 환경을 설정합니다. 디버그 모드 및 로깅(Logging)할 파일명을 포함한 파일 경로를 설정합니다.

PARAMETER

▶ ***DebugMode*** : 디버그 모드를 설정합니다. 디버그 모드의 종류 및 의미는 다음과 같습니다.

Value	Meaning
0 (DEBUG_OUT_WINDOW)	자체 윈도우를 생성하여 메시지를 출력합니다.
1 (DEBUG_OUT_LOCALFILE)	파일로 로그를 기록합니다.
2 (DEBUG_OUT_CONSOLE)	파일로 로그를 기록하며, 콘솔 윈도우를 생성하여 메시지를 출력합니다.

▶ ***szDebugFileName*** : 로깅(Logging)할 파일명을 포함한 경로를 지정합니다.
단, DebugMode 를 0 (DEBUG_OUT_WINDOW) 으로 설정하였을 경우에는 szDebugFileName 을 NULL 로 설정하셔야 합니다.

RETURN VALUE

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

EXAMPLE

C/C++

```
#include “ceSDK.h”
#include “ceSDKDef.h”

void OnDebegModeSet ()
{
    // 디버그 모드를 설정합니다.
    ceGnDebugMode ( DEBUG_OUT_WINDOW, // 디버그 모드 선택
                    NULL     // 0 번 모드로 사용할 경우 파일 경로에 NULL 값을 입력하시면 됩니다.
                    );
}
```

NAME

ceGnTotalNode

- 탐색된 전체 원격 노드 개수 반환

INFORMATION	
1	General Function
!	VC++ (6, 7, 8)/VB
	BCB/Delphi
:	Level 1
J	위험 요소 없음

SYNOPSIS

r VT_I4 ceGnTotalNode ([out] VT_PI4 Node)

DESCRIPTION

탐색(Search)된 전체 원격 노드의 개수를 반환합니다.

PARAMETER

▶ ***Node*** : 이 매개변수를 통하여 탐색된 전체 원격 노드의 개수를 반환합니다.

RETURN VALUE

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

EXAMPLE

C/C++

```
#include “ceSDK.h”
#include “ceSDKDef.h”

void OnGetTotalNode ()
{
    long nTotalNode;              // 탐색된 원격 노드 정보

    /* 탐색된 원격 노드 개수를 반환합니다.*/
    if ( ceGnTotalNode( &nTotalNode ) != ceERR_NONE )
    {
            OutputDebugString ( " ceGnTotalNode has been failed" );
    }
}
```

Visual Basic

```
Private Sub OnGetTotalNode ()

    Dim nTotalNode As Long              ‘ 탐색된 원격 노드 정보

    ‘ 탐색된 원격 노드 개수를 반환합니다.
    If ceGnTotalNode ( nTotalNode ) <> ceERR_NONE Then
            MsgBox ( “ceGnTotalNode has been failed” )
    End If

End Sub
```

Delphi

```
procedure OnGetTotalNode ();
var
    nTotalNode : LongInt;                  // 탐색된 원격 노드 정보

begin

    // 탐색된 원격 노드 개수를 반환합니다.
    if ceGnTotalNode ( @nTotalNode ) <> ceERR_NONE then
    begin
            ShowMessage ( ‘ceGnTotalNode has been failed’ );
    end;
end;
```

NAME

ceGnTotalMotionChannel

- 탐색된 전체 모션 제어 축(Axis) 개수 반환

INFORMATION	
1	General Function
!	VC++ (6, 7, 8)/VB
	BCB/Delphi
:	Level 1
J	위험 요소 없음

SYNOPSIS

r VT_I4 ceGnTotalMotionChannel ([out] VT_PI4 Channel)

DESCRIPTION

탐색(Search)된 전체 원격 노드의 모션 모듈을 대상으로 모션 제어 축 개수를 반환합니다.

PARAMETER

▶ *Channel* : 이 매개변수를 통하여 탐색된 전체 모션 제어 축 개수를 반환합니다.

RETURN VALUE

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

SEE ALSO

ceGnTotalDIOChannel, ceGnTotalAIChannel, ceGnTotalAOChannel, ceGnTotalMDIOChannel, ceGnTotalCNTChannel, ceGnTotalSERChannel

EXAMPLE

C/C++

```
#include “ceSDK.h”
#include “ceSDKDef.h”

void OnGetTotalChannel ()
{
    long nTotalAxes, nTotalDioCh, nTotalAiCh, nTotalAoCh, nTotalMdioCh, nTotalCntCh, nTotalSerCh;

    // 탐색된 전체 모션 제어 축 수를 반환합니다.
    ceGnTotalMotionChannel ( &nTotalAxes );

    // 탐색된 전체 DIO 채널 개수를 반환합니다.
    ceGnTotalDIOChannel ( &nTotalDioCh );

    // 탐색된 전체 AI 채널 개수를 반환합니다.
    ceGnTotalAIChannel ( &nTotalAiCh );

    // 탐색된 전체 AO 채널 개수를 반환합니다.
    ceGnTotalAOChannel ( &nTotalAoCh );

    // 탐색된 전체 MDIO 채널 개수를 반환합니다.
    ceGnTotalMDIOChannel ( &nTotalMdioCh );

    // 탐색된 전체 카운터 채널 개수를 반환합니다.
    ceGnTotalCNTChannel ( &nTotalCntCh );

    // 탐색된 전체 시리얼 통신 채널 개수를 반환합니다.
    ceGnTotalSERChannel ( &nTotalSerCh );
}
```

Visual Basic

```
Private Sub OnGetTotalChannel ()

    Dim nTotalAxes As Long, nTotalDioCh As Long, nTotalAiCh As Long, nTotalAoCh As Long
    Dim nTotalMdioCh As Long, nTotalCntCh As Long, nTotalSerCh As Long

    ‘ 탐색된 전체 모션 제어 축 수를 반환합니다.
    Call ceGnTotalMotionChannel ( nTotalAxes )

    ‘ 탐색된 전체 DIO 채널 개수를 반환합니다.
    Call ceGnTotalDIOChannel ( nTotalDioCh )

    ‘ 탐색된 전체 AI 채널 개수를 반환합니다.
    Call ceGnTotalAIChannel ( nTotalAiCh )

    ‘ 탐색된 전체 AO 채널 개수를 반환합니다.
    Call ceGnTotalAOChannel ( nTotalAoCh )

    ‘ 탐색된 전체 MDIO 채널 개수를 반환합니다.
```

```
    Call ceGnTotalMDIOChannel ( nTotalMdioCh )

    ‘ 탐색된 전체 카운터 채널 개수를 반환합니다.
    Call ceGnTotalCNTChannel ( nTotalCntCh )

    ‘ 탐색된 전체 시리얼 통신 채널 개수를 반환합니다.
    Call ceGnTotalSERChannel ( nTotalSerCh )

End Sub
```

Delphi

```
procedure OnGetTotalChannel ();
var
    nTotalAxes, nTotalDioCh, nTotalAiCh, nTotalAoCh, nTotalMdioCh, nTotalCntCh, nTotalSerCh : LongInt;

begin
    ‘ 탐색된 전체 모션 제어 축 수를 반환합니다.
     ceGnTotalMotionChannel ( @nTotalAxes );

    ‘ 탐색된 전체 DIO 채널 개수를 반환합니다.
    ceGnTotalDIOChannel ( @nTotalDioCh );

    ‘ 탐색된 전체 AI 채널 개수를 반환합니다.
    ceGnTotalAIChannel ( @nTotalAiCh );

    ‘ 탐색된 전체 AO 채널 개수를 반환합니다.
    ceGnTotalAOChannel ( @nTotalAoCh );

    ‘ 탐색된 전체 MDIO 채널 개수를 반환합니다.
    ceGnTotalMDIOChannel ( @nTotalMdioCh );

    ‘ 탐색된 전체 카운터 채널 개수를 반환합니다.
    ceGnTotalCNTChannel ( @nTotalCntCh );

    ‘ 탐색된 전체 시리얼 통신 채널 개수를 반환합니다.
    ceGnTotalSERChannel ( @nTotalSerCh );

end;
```

NAME

ceGnTotalDIOChannel

- 탐색된 전체 디지털 I/O 채널 개수 반환

INFORMATION
1 General Function
! VC++ (6, 7, 8)/VB
BCB/Delphi
: Level 1
J 위험 요소 없음

SYNOPSIS

r VT_I4 ceGnTotalDIOChannel ([out] VT_PI4 Channel)

DESCRIPTION

탐색(Search)된 전체 원격 노드의 디지털 I/O 모듈을 대상으로 DIO 채널(Digital Input/Output Channel) 개수를 반환합니다.

PARAMETER

▶ *Channel* : 이 매개변수를 통하여 탐색된 전체 디지털 I/O 채널의 개수를 반환합니다.

RETURN VALUE

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

SEE ALSO

ceGnTotalMotionChannel, ceGnTotalAIChannel, ceGnTotalAOChannel, ceGnTotalMDIOChannel, ceGnTotalCNTChannel, ceGnTotalSERChannel

EXAMPLE

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

NAME

ceGnTotalAIChannel

- 탐색된 전체 AI(Analog Input) 채널 개수 반환

INFORMATION	
1	General Function
!	VC++ (6, 7, 8)/VB
	BCB/Delphi
:	Level 1
J	위험 요소 없음

SYNOPSIS

r VT_I4 ceGnTotalAIChannel ([out] VT_PI4 Channel)

DESCRIPTION

탐색(Search)된 전체 원격 노드의 아날로그 입력 모듈을 대상으로 아날로그 입력 채널(Analog Input Channel)의 개수를 반환합니다.

PARAMETER

▶ *Channel* : 이 매개변수를 통하여 탐색된 전체 아날로그 입력 채널의 개수를 반환합니다.

RETURN VALUE

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

SEE ALSO

ceGnTotalMotionChannel, ceGnTotalDIOChannel, ceGnTotalAOChannel, ceGnTotalMDIOChannel, ceGnTotalCNTChannel, ceGnTotalSERChannel

EXAMPLE

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

NAME

ceGnTotalAOChannel

- 탐색된 전체 AO(Analog Output) 채널 개수 반환

INFORMATION	
1	General Function
!	VC++ (6, 7, 8)/VB
	BCB/Delphi
:	Level 1
J	위험 요소 없음

SYNOPSIS

r VT_I4 ceGnTotalAOChannel ([out] VT_PI4 Channel)

DESCRIPTION

탐색(Search)된 전체 원격 노드의 아날로그 출력 모듈을 대상으로 아날로그 출력 채널(Analog Output Channel)의 개수를 반환합니다.

PARAMETER

▶ *Channel* : 이 매개변수를 통하여 탐색된 전체 아날로그 출력 채널의 개수를 반환합니다.

RETURN VALUE

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

SEE ALSO

ceGnTotalMotionChannel, ceGnTotalDIOChannel, ceGnTotalAIChannel, ceGnTotalMDIOChannel, ceGnTotalCNTChannel, ceGnTotalSERChannel

EXAMPLE

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

NAME

ceGnTotalMDIOChannel

- 탐색된 전체 모션 디지털 I/O(Motion Digital I/O) 채널 개수 반환

INFORMATION	
1	General Function
!	VC++ (6, 7, 8)/VB
	BCB/Delphi
:	Level 1
J	위험 요소 없음

SYNOPSIS

r VT_I4 ceGnTotalMDIOChannel ([out] VT_PI4 Channel)

DESCRIPTION

탐색(Search)된 전체 원격 노드의 모션 모듈을 대상으로 모션 디지털 I/O 채널(Motion Digital Input /Output Channel)의 개수를 반환합니다.

PARAMETER

▶ *Channel* : 이 매개변수를 통하여 탐색된 전체 모션 디지털 I/O 채널의 개수를 반환합니다.

RETURN VALUE

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

SEE ALSO

ceGnTotalMotionChannel, ceGnTotalDIOChannel, ceGnTotalAIChannel, ceGnTotalAOChannel, ceGnTotalCNTChannel, ceGnTotalSERChannel

EXAMPLE

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

NAME

ceGnTotalCNTChannel

- 탐색된 전체 카운터(Counter) 채널 개수 반환

INFORMATION	
1	General Function
!	VC++ (6, 7, 8)/VB
	BCB/Delphi
:	Level 1
J	위험 요소 없음

SYNOPSIS

r VT_I4 ceGnTotalCNTChannel ([out] VT_PI4 Channel)

DESCRIPTION

탐색(Search)된 전체 원격 노드의 카운터 모듈을 대상으로 카운터 채널(Counter Channel)의 개수를 반환합니다.

PARAMETER

▶ *Channel* : 이 매개변수를 통하여 탐색된 전체 카운터 채널의 개수를 반환합니다.

RETURN VALUE

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

SEE ALSO

ceGnTotalMotionChannel, ceGnTotalDIOChannel, ceGnTotalAIChannel, ceGnTotalAOChannel, ceGnTotalMDIOChannel, ceGnTotalSERChannel

EXAMPLE

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

NAME

ceGnTotalSERChannel

- 탐색된 전체 시리얼 통신 채널 개수 반환

INFORMATION	
1	General Function
!	VC++ (6, 7, 8)/VB
	BCB/Delphi
:	Level 1
J	위험 요소 없음

SYNOPSIS

r VT_I4 ceGnTotalCNTChannel ([out] VT_PI4 Channel)

DESCRIPTION

탐색(Search)된 전체 원격 노드의 시리얼 통신 모듈을 대상으로 시리얼 통신 채널의 개수를 반환합니다.

PARAMETER

▶ *Channel* : 이 매개변수를 통하여 탐색된 전체 시리얼 통신 채널의 개수를 반환합니다.

RETURN VALUE

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

SEE ALSO

ceGnTotalMotionChannel, ceGnTotalDIOChannel, ceGnTotalAIChannel, ceGnTotalAOChannel, ceGnTotalMDIOChannel, ceGnTotalCNTChannel

EXAMPLE

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

NAME

ceGnModuleCount_Motion

- 해당 원격 노드의 탐색된 모션 모듈 (Motion Module) 개수 반환

INFORMATION	
1	General Function
!	VC++ (6, 7, 8)/VB
	BCB/Delphi
:	Level 1
J	위험 요소 없음

SYNOPSIS

r VT_I4 ceGnModuleCount_Motion ([in] VT_I4 NodeID, [out] VT_PI4 ModuleCount)

DESCRIPTION

해당 원격 노드에 대해 탐색된 모션 모듈(Motion Module)의 개수를 반환합니다.

PARAMETER

▶ ***NodeID*** : 원격 노드 ID. 탐색된 모션 모듈의 개수를 얻어올 원격 노드 ID 를 설정합니다.

▶ ***ModuleCount*** : 이 매개변수를 통하여 해당 원격 노드 내의 탐색된 모션 모듈 개수를 반환합니다.

모션 모듈(Motion Module) 에 대한 안내.
cEIP 슬레이브 모듈 중 ceMC02P, ceMC04P 모듈을 지칭합니다. 향후 다른 사양의 모션 모듈이 추가될 수 있습니다.

RETURN VALUE

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

SEE ALSO

ceGnModuleCount_Dio, ceGnModuleCount_Ai, ceGnModuleCount_Ao, ceGnModuleCount_Mdio, ceGnModuleCount_Cnt, ceGnModuleCount_Ser

EXAMPLE

C/C++

```
#include “ceSDK.h”
#include “ceSDKDef.h”

void OnGetModuleCount ()
{
    long nNodeID = 1;                       // 해당 원격 노드 ID
    long nMotModuleCnt, nDioModuleCnt, nAiModuleCnt, nAoModuleCnt, nCountModuleCnt, nSerModuleCnt;


    // 해당 원격 노드에 대해 탐색된 모션 모듈 개수를 반환합니다.
    ceGnModuleCount_Motion ( nNodeID, &nMotModuleCnt );

    // 해당 원격 노드에 대해 탐색된 DIO 모듈 개수를 반환합니다.
    ceGnModuleCount_Dio ( nNodeID, &nDioModuleCnt );

    // 해당 원격 노드에 대해 탐색된 AI 모듈 개수를 반환합니다.
    ceGnModuleCount_Ai ( nNodeID, &nAiModuleCnt );

    // 해당 원격 노드에 대해 탐색된 AO 모듈 개수를 반환합니다.
    ceGnModuleCount_Ao ( nNodeID, &nAoModuleCnt );

    // 해당 원격 노드에 대해 탐색된 MDIO 채널이 속한 모션 모듈 개수를 반환합니다.
    ceGnModuleCount_Mdio ( nNodeID, &nMotModuleCnt );

    // 해당 원격 노드에 대해 탐색된 카운터 모듈 개수를 반환합니다.
    ceGnModuleCount_Cnt ( nNodeID, &nCountModuleCnt );

    // 해당 원격 노드에 대해 탐색된 시리얼 통신 모듈 개수를 반환합니다.
    ceGnModuleCount_Ser ( nNodeID, &nSerModuleCnt );
}
```

Visual Basic

```
Private Sub OnGetModuleCount ()

    Dim nNodeID As Long                     ‘ 해당 원격 노드 ID
    Dim nMotModuleCnt As Long, nDioModuleCnt As Long, nAiModuleCnt As Long
    Dim nAoModuleCnt As Long, nCountModuleCnt As Long, nSerModuleCnt As Long


    ‘ 해당 원격 노드에 대해 탐색된 모션 모듈 개수를 반환합니다.
    Call ceGnModuleCount_Motion ( nNodeID, nMotModuleCnt )

    ‘ 해당 원격 노드에 대해 탐색된 DIO 모듈 개수를 반환합니다.
    Call ceGnModuleCount_Dio ( nNodeID, nDioModuleCnt )

    ‘ 해당 원격 노드에 대해 탐색된 AI 모듈 개수를 반환합니다.
    Call ceGnModuleCount_Ai ( nNodeID, nAiModuleCnt )
```

```
    ‘ 해당 원격 노드에 대해 탐색된 AO 모듈 개수를 반환합니다.
    Call ceGnModuleCount_Ao ( nNodeID, nAoModuleCnt )

    ‘ 해당 원격 노드에 대해 탐색된 MDIO 채널이 속한 모션 모듈 개수를 반환합니다.
    Call ceGnModuleCount_Mdio ( nNodeID, nMotModuleCnt )

    ‘ 해당 원격 노드에 대해 탐색된 카운터 모듈 개수를 반환합니다.
    Call ceGnModuleCount_Cnt ( nNodeID, nCountModuleCnt )

    ‘ 해당 원격 노드에 대해 탐색된 시리얼 통신 모듈 개수를 반환합니다.
    Call ceGnModuleCount_Ser ( nNodeID, nSerModuleCnt )

End Sub
```

Delphi

```
procedure OnGetModuleCount ();
var
    nNodeID : LongInt;                      // 해당 원격 노드 ID
    nMotModuleCnt, nDioModuleCnt, nAiModuleCnt, nAoModuleCnt, nCountModuleCnt,
    nSerModuleCnt : LongInt;

begin
    nNodeID := 1;

    // 해당 원격 노드에 대해 탐색된 모션 모듈 개수를 반환합니다.
    ceGnModuleCount_Motion ( nNodeID, @nMotModuleCnt );

    // 해당 원격 노드에 대해 탐색된 DIO 모듈 개수를 반환합니다.
    ceGnModuleCount_Dio ( nNodeID, @nDioModuleCnt );

    // 해당 원격 노드에 대해 탐색된 AI 모듈 개수를 반환합니다.
    ceGnModuleCount_Ai ( nNodeID, @nAiModuleCnt );

    // 해당 원격 노드에 대해 탐색된 AO 모듈 개수를 반환합니다.
    ceGnModuleCount_Ao ( nNodeID, @nAoModuleCnt );

    // 해당 원격 노드에 대해 탐색된 MDIO 채널이 속한 모션 모듈 개수를 반환합니다.
    ceGnModuleCount_Mdio ( nNodeID, @nMotModuleCnt );

    // 해당 원격 노드에 대해 탐색된 카운터 모듈 개수를 반환합니다.
    ceGnModuleCount_Cnt ( nNodeID, @nCountModuleCnt );

    // 해당 원격 노드에 대해 탐색된 시리얼 통신 모듈 개수를 반환합니다.
    ceGnModuleCount_Ser ( nNodeID, @nSerModuleCnt );

end;
```

NAME

ceGnModuleCount_Dio

- 해당 원격 노드의 탐색된 디지털 I/O 모듈 (Digital Input/Output Module) 개수 반환

INFORMATION	
1	General Function
!	VC++ (6, 7, 8)/VB
	BCB/Delphi
:	Level 1
J	위험 요소 없음

SYNOPSIS

r VT_I4 ceGnModuleCount_Dio ([in] VT_I4 NodeID, [out] VT_PI4 ModuleCount)

DESCRIPTION

해당 원격 노드에 대해 탐색된 디지털 I/O 모듈(Digital Input/Output Module)의 개수를 반환합니다.

PARAMETER

▶ ***NodeID*** : 원격 노드 ID. 탐색된 디지털 I/O 모듈의 개수를 얻어올 원격 노드 ID 를 설정합니다.

▶ ***ModuleCount*** : 이 매개변수를 통하여 해당 원격 노드 내의 탐색된 디지털 I/O 모듈 개수를 반환합니다.

디지털 I/O 모듈(Digital I/O Module) 에 대한 안내.

cEIP 슬레이브 모듈 중 ceD16CM, ceDI32N, ceDO32N 모듈을 지칭합니다.
향후 다른 사양의 디지털 I/O 모듈이 추가될 수 있습니다.

RETURN VALUE

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

SEE ALSO

ceGnModuleCount_Motion, ceGnModuleCount_Ai, ceGnModuleCount_Ao,
ceGnModuleCount_Mdio, ceGnModuleCount_Cnt, ceGnModuleCount_Ser

EXAMPLE

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

NAME

ceGnModuleCount_Ai

- 해당 원격 노드의 탐색된 아날로그 입력 모듈 (Analog Input Module) 개수 반환

INFORMATION	
1	General Function
!	VC++ (6, 7, 8)/VB
	BCB/Delphi
:	Level 1
J	위험 요소 없음

SYNOPSIS

r VT_I4 ceGnModuleCount_Ai ([in] VT_I4 NodeID, [out] VT_PI4 ModuleCount)

DESCRIPTION

해당 원격 노드에 대해 탐색된 아날로그 입력 모듈(Analog Input Module)의 개수를 반환합니다.

PARAMETER

▶ ***NodeID*** : 원격 노드 ID. 탐색된 아날로그 입력 모듈의 개수를 얻어올 원격 노드 ID 를 설정합니다.

▶ ***ModuleCount*** : 이 매개변수를 통하여 해당 원격 노드 내의 탐색된 아날로그 입력 모듈 개수를 반환합니다.

아날로그 입력 모듈(Analog Input Module) 에 대한 안내.
cEIP 슬레이브 모듈 중 ceAI08A 모듈을 지칭합니다. 향후 다른 사양의 아날로그 입력 모듈이 추가될 수 있습니다.

RETURN VALUE

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

SEE ALSO

ceGnModuleCount_Motion, ceGnModuleCount_Dio, ceGnModuleCount_Ao, ceGnModuleCount_Mdio, ceGnModuleCount_Cnt, ceGnModuleCount_Ser

EXAMPLE

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

NAME

ceGnModuleCount_Ao

- 해당 원격 노드의 탐색된 아날로그 출력 모듈 (Analog Output Module) 개수 반환

INFORMATION	
1	General Function
!	VC++ (6, 7, 8)/VB
	BCB/Delphi
:	Level 1
J	위험 요소 없음

SYNOPSIS

r VT_I4 ceGnModuleCount_Ao ([in] VT_I4 NodeID, [out] VT_PI4 ModuleCount)

DESCRIPTION

해당 원격 노드에 대해 탐색된 아날로그 출력 모듈(Analog Output Module)의 개수를 반환합니다.

PARAMETER

▶ ***NodeID*** : 원격 노드 ID. 탐색된 아날로그 출력 모듈의 개수를 얻어올 노드 ID 를 설정합니다.

▶ ***ModuleCount*** : 이 매개변수를 통하여 해당 원격 노드 내의 탐색된 아날로그 출력 모듈 개수를 반환합니다.

아날로그 출력 모듈(Analog Output Module) 에 대한 안내.

cEIP 슬레이브 모듈 중 ceAO02A, ceAO04A 모듈을 지칭합니다.
향후 다른 사양의 아날로그 출력 모듈이 추가될 수 있습니다.

RETURN VALUE

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

SEE ALSO

ceGnModuleCount_Motion, ceGnModuleCount_Dio, ceGnModuleCount_Ai, ceGnModuleCount_Mdio, ceGnModuleCount_Cnt, ceGnModuleCount_Ser

EXAMPLE

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

NAME

ceGnModuleCount_Mdio

- 해당 원격 노드의 탐색된 모션 디지털 I/O (MDIO) 채널이 속한 모션 모듈 개수 반환

INFORMATION	
1	General Function
!	VC++ (6, 7, 8)/VB
	BCB/Delphi
:	Level 1
J	위험 요소 없음

SYNOPSIS

r VT_I4 ceGnModuleCount_Mdio ([in] VT_I4 NodeID, [out] VT_PI4 ModuleCount)

DESCRIPTION

해당 원격 노드에 대해 탐색된 모션 디지털 I/O 채널(Motion Digital Input/Output Channel)이 속한 모션 모듈의 개수를 반환합니다.

PARAMETER

▶ *NodeID* : 원격 노드 ID. 탐색된 MDIO 채널이 속한 모션 모듈의 개수를 얻어올 원격 노드 ID 를 설정합니다.

▶ *ModuleCount* : 해당 원격 노드 내의 MDIO 채널이 속한 모션 모듈 개수를 반환합니다.

모션 디지털 I/O 채널에 대한 안내.

모션 디지털 I/O 채널은 cEIP 슬레이브 모듈 중 ceMC02P 모듈에서 제공하는 디지털 I/O 채널을 지칭합니다.
향후 다른 사양의 모션 모듈이 추가될 수 있습니다.

RETURN VALUE

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

SEE ALSO

ceGnModuleCount_Motion, ceGnModuleCount_Dio, ceGnModuleCount_Ai, ceGnModuleCount_Ao, ceGnModuleCount_Cnt, ceGnModuleCount_Ser

EXAMPLE

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

NAME

ceGnModuleCount_Cnt

- 해당 원격 노드의 탐색된 카운터 모듈 (Counter Module) 개수 반환

INFORMATION	
1	General Function
!	VC++ (6, 7, 8)/VB
	BCB/Delphi
:	Level 1
J	위험 요소 없음

SYNOPSIS

r VT_I4 ceGnModuleCount_Cnt ([in] VT_I4 NodeID, [out] VT_PI4 ModuleCount)

DESCRIPTION

해당 원격 노드에 대해 탐색된 카운터 모듈(Counter Module)의 개수를 반환합니다.

PARAMETER

▶ ***NodeID*** : 원격 노드 ID. 탐색된 카운터 모듈의 개수를 얻어올 노드 ID 를 설정합니다.

▶ ***ModuleCount*** : 이 매개변수를 통하여 해당 원격 노드 내의 탐색된 카운터 모듈 개수를 반환합니다.

카운터 모듈(Counter Module) 에 대한 안내.
cEIP 슬레이브 모듈 중 ceCN08A 모듈을 지칭합니다. 향후 다른 사양의 카운터 모듈이 추가될 수 있습니다.

RETURN VALUE

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

SEE ALSO

ceGnModuleCount_Motion, ceGnModuleCount_Dio, ceGnModuleCount_Ai, ceGnModuleCount_Ao, ceGnModuleCount_Mdio, ceGnModuleCount_Ser

EXAMPLE

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

NAME

ceGnModuleCount_Ser

- 해당 원격 노드의 탐색된 시리얼 통신 모듈 (Serial Communication Module) 개수 반환

INFORMATION	
1	General Function
!	VC++ (6, 7, 8)/VB
	BCB/Delphi
:	Level 1
J	위험 요소 없음

SYNOPSIS

r VT_I4 ceGnModuleCount_Ser ([in] VT_I4 NodeID, [out] VT_PI4 ModuleCount)

DESCRIPTION

해당 원격 노드에 대해 탐색된 시리얼 통신 모듈(Serial Communication Module)의 개수를 반환합니다.

PARAMETER

▶ *NodeID* : 원격 노드 ID. 탐색된 아날로그 출력 모듈의 개수를 얻어올 원격 노드 ID 를 설정합니다.

▶ *ModuleCount* : 이 매개변수를 통하여 해당 원격 노드 내의 탐색된 아날로그 출력 모듈 개수를 반환합니다.

아날로그 출력 모듈(Analog Output Module) 에 대한 안내.

cEIP 슬레이브 모듈 중 ceAO02A, ceAO04A 모듈을 지칭합니다.
향후 다른 사양의 아날로그 출력 모듈이 추가될 수 있습니다.

RETURN VALUE

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

SEE ALSO

ceGnModuleCount_Motion, ceGnModuleCount_Dio, ceGnModuleCount_Ai, ceGnModuleCount_Mdio, ceGnModuleCount_Cnt, ceGnModuleCount_Ser

EXAMPLE

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

NAME

ceGnChannelCount_Motion

- 해당 모션 모듈에 대해 탐색된 모션 제어 축(Axis) 개수 반환

INFORMATION
1 General Function
! VC++ (6, 7, 8)/VB
BCB/Delphi
: Level 1
J 위험 요소 없음

SYNOPSIS

r VT_I4 ceGnChannelCount_Motion ([in] VT_I4 NodeID, [in] VT_I4 ModuleIdx, [out] VT_PI4 ChannelCount)

DESCRIPTION

해당 원격 노드의 모션 모듈에 대해 탐색된 모션 제어 축(Axis) 개수를 반환합니다.

PARAMETER

▶ ***NodeID*** : 원격 노드 ID. 탐색된 모션 제어 축 개수를 얻어올 모션 모듈이 속한 원격 노드 ID 를 설정합니다.

▶ ***ModuleIdx*** : 모션 모듈 ID. 탐색된 모션 제어 축 개수를 얻어올 모션 모듈 ID 를 설정합니다.

▶ ***ChannelCount*** : 이 매개변수를 통하여 해당 모션 모듈에 대해 탐색된 모션 제어 축 개수를 반환합니다.

RETURN VALUE

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

SEE ALSO

ceGnChannelCount_Dio, ceGnChannelCount_Ai, ceGnChannelCount_Ao, ceGnChannelCount_Mdio, ceGnChannelCount_Cnt, ceGnChannelCount_Ser

EXAMPLE

C/C++

```
#include “ceSDK.h”
#include “ceSDKDef.h”

void OnGetChannelCount ()
{
    long nNodeID = 1                    // 해당 원격 노드 ID

    long nMotModuleIdx = 1;             // 해당 모션 모듈 ID
    long nDioModuleIdx = 2;             // 해당 DIO 모듈 ID
    long nAiModuleIdx = 3;              // 해당 AI 모듈 ID
    long nAoModuleIdx = 4;              // 해당 AO 모듈 ID
    long nCountModuleIdx = 5;           // 해당 카운터 모듈 ID
    long nSerModuleIdx = 6;             // 해당 시리얼 통신 모듈 ID

    long nAxesCnt, nDioChCnt, nAiChCnt, nAoChCnt, nMdioChCnt, nCountChCnt, nSerChCnt;


    // 해당 원격 노드 모션 모듈에 대해 탐색된 모션 제어 축 개수를 반환합니다.
    ceGnChannelCount_Motion ( nNodeID, nMotModuleIdx, &nAxesCnt );

    // 해당 원격 노드 DIO 모듈에 대해 탐색된 DIO 채널 개수를 반환합니다.
    ceGnChannelCount_Dio ( nNodeID, nDioModuleIdx, &nDioChCnt );

    // 해당 원격 노드 AI 모듈에 대해 탐색된 AI 채널 개수를 반환합니다.
    ceGnChannelCount_Ai ( nNodeID, nAiModuleIdx, &nAiChCnt );

    // 해당 원격 노드 AO 모듈에 대해 탐색된 AO 채널 개수를 반환합니다.
    ceGnChannelCount_Ao ( nNodeID, nAoModuleIdx, &nAoChCnt );

    // 해당 원격 노드 모션 모듈에 대해 탐색된 MDIO 채널 개수를 반환합니다.
    ceGnChannelCount_Mdio ( nNodeID, nMotModuleIdx, &nMdioChCnt );

    // 해당 원격 노드 카운터 모듈에 대해 탐색된 카운터 채널 개수를 반환합니다.
    ceGnChannelCount_Cnt ( nNodeID, nCountModuleIdx, &nCountChCnt );

    // 해당 원격 노드 시리얼 통신 모듈에 대해 탐색된 카운터 채널 개수를 반환합니다.
    ceGnChannelCount_Cnt ( nNodeID, nSerModuleIdx, &nSerChCnt );
}
```

Visual Basic

```
Private Sub OnGetChannelCount ()

    Dim nNodeID As Long

    Dim nMotModuleIdx As Long, nDioModuleIdx As Long, nAiModuleIdx As Long
    Dim nAoModuleIdx As Long, nCountModuleIdx As Long, nSerModuleIdx As Long
```

```
    Dim nAxesCnt As Long, nDioChCnt As Long, nAiChCnt As Long, nAoChCnt As Long
    Dim nMdioChCnt As Long, nCountChCnt As Long, nSerChCnt As Long

    nNodeID = 1                 ‘ 해당 원격 노드 ID

    nMotModuleIdx = 1           ‘ 해당 모션 모듈 ID
    nDioModuleIdx = 2           ‘ 해당 DIO 모듈 ID
    nAiModuleIdx = 3            ‘ 해당 AI 모듈 ID
    nAoModuleIdx = 4            ‘ 해당 AO 모듈 ID
    nCountModuleIdx = 5         ‘ 해당 카운터 모듈 ID
    nSerModuleIdx = 6           ‘ 해당 시리얼 통신 모듈 ID

    ‘ 해당 원격 노드 모션 모듈에 대해 탐색된 모션 제어 축 개수를 반환합니다.
    Call ceGnChannelCount_Motion ( nNodeID, nMotModuleIdx, nAxesCnt )

    ‘ 해당 원격 노드 DIO 모듈에 대해 탐색된 DIO 채널 개수를 반환합니다.
    Call ceGnChannelCount_Dio ( nNodeID, nDioModuleIdx, nDioChCnt )

    ‘ 해당 원격 노드 AI 모듈에 대해 탐색된 AI 채널 개수를 반환합니다.
    Call ceGnChannelCount_Ai ( nNodeID, nAiModuleIdx, nAiChCnt )

    ‘ 해당 원격 노드 AO 모듈에 대해 탐색된 AO 채널 개수를 반환합니다.
    Call ceGnChannelCount_Ao ( nNodeID, nAoModuleIdx, nAoChCnt )

    ‘ 해당 원격 노드 모션 모듈에 대해 탐색된 MDIO 채널 개수를 반환합니다.
    Call ceGnChannelCount_Mdio ( nNodeID, nMotModuleIdx, nMdioChCnt )

    ‘ 해당 원격 노드 카운터 모듈에 대해 탐색된 카운터 채널 개수를 반환합니다.
    Call ceGnChannelCount_Cnt ( nNodeID, nCountModuleIdx, nCountChCnt )

    ‘ 해당 원격 노드 시리얼 통신 모듈에 대해 탐색된 카운터 채널 개수를 반환합니다.
    Call ceGnChannelSer_Cnt ( nNodeID, nSerModuleIdx, nSerChCnt )

End Sub
```

Delphi

```
procedure OnGetChannelCount ()
var
    nNodeID : LongInt;

    nMotModuleIdx, nDioModuleIdx, nAiModuleIdx, nAoModuleIdx, nCountModuleIdx, nSerModuleIdx : LongInt;

    nAxesCnt, nDioChCnt, nAiChCnt, nAoChCnt, nMdioChCnt, nCountChCnt, nSerChCnt : LongInt;

begin
    nNodeID := 1;               // 해당 원격 노드 ID

    nMotModuleIdx := 1;         // 해당 모션 모듈 ID
    nDioModuleIdx := 2;         // 해당 DIO 모듈 ID
```

```
    nAiModuleIdx := 3;          // 해당 AI 모듈 ID
    nAoModuleIdx := 4;          // 해당 AO 모듈 ID
    nCountModuleIdx := 5;       // 해당 카운터 모듈 ID
    nCountModuleIdx := 6;       // 해당 시리얼 통신 모듈 ID


    // 해당 원격 노드 모션 모듈에 대해 탐색된 모션 제어 축 개수를 반환합니다.
    ceGnChannelCount_Motion ( nNodeID, nMotModuleIdx, @nAxesCnt );

    // 해당 원격 노드 DIO 모듈에 대해 탐색된 DIO 채널 개수를 반환합니다.
    ceGnChannelCount_Dio ( nNodeID, nDioModuleIdx, @nDioChCnt );

    // 해당 원격 노드 AI 모듈에 대해 탐색된 AI 채널 개수를 반환합니다.
    ceGnChannelCount_Ai ( nNodeID, nAiModuleIdx, @nAiChCnt );

    // 해당 원격 노드 AO 모듈에 대해 탐색된 AO 채널 개수를 반환합니다.
    ceGnChannelCount_Ao ( nNodeID, nAoModuleIdx, @nAoChCnt );

    // 해당 원격 노드 모션 모듈에 대해 탐색된 MDIO 채널 개수를 반환합니다.
    ceGnChannelCount_Mdio ( nNodeID, nMotModuleIdx, @nMdioChCnt );

    // 해당 원격 노드 카운터 모듈에 대해 탐색된 카운터 채널 개수를 반환합니다.
    ceGnChannelCount_Cnt ( nNodeID, nCountModuleIdx, @nCountChCnt );

    // 해당 원격 노드 시리얼 통신 모듈에 대해 탐색된 카운터 채널 개수를 반환합니다.
    ceGnChannelCount_Ser ( nNodeID, nSerModuleIdx, @nSerChCnt );

end;
```

NAME

ceGnChannelCount_Dio

- 해당 디지털 I/O 모듈에 대해 탐색된 DIO 채널 (DIO Channel) 개수 반환

INFORMATION	
1	General Function
!	VC++ (6, 7, 8)/VB
	BCB/Delphi
:	Level 1
J	위험 요소 없음

SYNOPSIS

r VT_I4 ceGnChannelCount_Dio ([in] VT_I4 NodeID, [in] VT_I4 ModuleIdx, [out] VT_PI4 ChannelCount)

DESCRIPTION

해당 원격 노드의 디지털 I/O 모듈에 대해 탐색된 DIO 채널(DIO Channel) 개수를 반환합니다.

PARAMETER

▶ *NodeID* : 원격 노드 ID. 탐색된 DIO 채널 개수를 얻어올 디지털 I/O 모듈이 속한 원격 노드 ID 를 설정합니다.

▶ *ModuleIdx* : 디지털 I/O 모듈 ID. 탐색된 DIO 채널 개수를 얻어올 디지털 I/O 모듈 ID 를 설정합니다.

▶ *ChannelCount* : 이 매개변수를 통하여 해당 디지털 I/O 모듈에 대해 탐색된 DIO 채널 개수를 반환합니다.

RETURN VALUE

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

SEE ALSO

ceGnChannelCount_Motion, ceGnChannelCount_Ai, ceGnChannelCount_Ao, ceGnChannelCount_Mdio, ceGnChannelCount_Cnt, ceGnChannelCount_Ser

EXAMPLE

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

NAME

ceGnChannelCount_Ai

- 해당 아날로그 입력 모듈에 대해 탐색된 AI 채널 (AI Channel) 개수 반환

INFORMATION	
1	General Function
!	VC++ (6, 7, 8)/VB
	BCB/Delphi
:	Level 1
J	위험 요소 없음

SYNOPSIS

r VT_I4 ceGnChannelCount_Ai ([in] VT_I4 NodeID, [in] VT_I4 ModuleIdx,
[out] VT_PI4 ChannelCount)

DESCRIPTION

해당 원격 노드의 아날로그 입력 모듈에 대해 탐색된 AI 채널(AI Channel) 개수를 반환합니다.

PARAMETER

▶ *NodeID* : 원격 노드 ID. 탐색된 AI 채널 개수를 얻어올 아날로그 입력 모듈이 속한 원격 노드 ID 를 설정합니다.

▶ *ModuleIdx* : 아날로그 입력 모듈 ID. 탐색된 AI 채널 개수를 얻어올 아날로그 입력 모듈 ID 를 설정합니다.

▶ *ChannelCount* : 이 매개변수를 통하여 해당 아날로그 입력 모듈에 대해 탐색된 AI 채널 개수를 반환합니다.

RETURN VALUE

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

SEE ALSO

ceGnChannelCount_Motion, ceGnChannelCount_Dio, ceGnChannelCount_Ao, ceGnChannelCount_Mdio, ceGnChannelCount_Cnt, ceGnChannelCount_Ser

EXAMPLE

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

NAME

ceGnChannelCount_Ao

- 해당 아날로그 출력 모듈에 대해 탐색된 AO 채널 (AO Channel) 개수 반환

INFORMATION	
1	General Function
!	VC++ (6, 7, 8)/VB
	BCB/Delphi
:	Level 1
J	위험 요소 없음

SYNOPSIS

r VT_I4 ceGnChannelCount_Ao ([in] VT_I4 NodeID, [in] VT_I4 ModuleIdx,
[out] VT_PI4 ChannelCount)

DESCRIPTION

해당 원격 노드의 아날로그 출력 모듈에 대해 AO 채널(AO Channel) 개수를 반환합니다.

PARAMETER

▶ *NodeID* : 원격 노드 ID. 탐색된 AO 채널 개수를 얻어올 아날로그 출력 모듈이 속한 원격 노드 ID 를 설정합니다.

▶ *ModuleIdx* : 아날로그 출력 모듈 ID. 탐색된 AO 채널 개수를 얻어올 아날로그 출력 모듈 ID 를 설정합니다.

▶ *ChannelCount* : 이 매개변수를 통하여 해당 아날로그 출력 모듈에 대해 탐색된 AO 채널 개수를 반환합니다.

RETURN VALUE

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

SEE ALSO

ceGnChannelCount_Motion, ceGnChannelCount_Dio, ceGnChannelCount_Ai, ceGnChannelCount_Mdio, ceGnChannelCount_Cnt, ceGnChannelCount_Ser

EXAMPLE

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

NAME

ceGnChannelCount_Mdio

- 해당 모션 모듈에 대해 탐색된 모션 디지털 I/O 채널 (MDIO Channel) 개수 반환

INFORMATION	
1	General Function
!	VC++ (6, 7, 8)/VB
	BCB/Delphi
:	Level 1
J	위험 요소 없음

SYNOPSIS

r VT_I4 ceGnChannelCount_Mdio ([in] VT_I4 NodeID, [in] VT_I4 ModuleIdx, [out] VT_PI4 ChannelCount)

DESCRIPTION

해당 원격 노드의 모션 모듈에 대해 탐색된 모션 디지털 I/O 채널(MDIO Channel) 개수를 반환합니다. ChannelCount 변수를 통하여 반환되는 채널의 수는 MDI, MDO 각각의 채널 개수를 의미합니다.

PARAMETER

▶ *NodeID* : 원격 노드 ID. 탐색된 MDIO 채널 개수를 얻어올 모션 모듈이 속한 원격 노드 ID 를 설정합니다.

▶ *ModuleIdx* : 모션 모듈 ID. 탐색된 MDIO 채널 개수를 얻어올 모션 모듈 ID 를 설정합니다.

▶ *ChannelCount* : 이 매개변수를 통하여 해당 모션 모듈에 대해 탐색된 MDIO 채널의 개수를 반환합니다.

RETURN VALUE

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

SEE ALSO

ceGnChannelCount_Motion, ceGnChannelCount_Dio, ceGnChannelCount_Ai, ceGnChannelCount_Ao, ceGnChannelCount_Cnt, ceGnChannelCount_Ser

EXAMPLE

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

NAME

ceGnChannelCount_Cnt

- 해당 카운터 모듈에 대해 탐색된 카운터 채널 (Counter Channel) 개수 반환

INFORMATION	
1	General Function
!	VC++ (6, 7, 8)/VB
	BCB/Delphi
:	Level 1
J	위험 요소 없음

SYNOPSIS

r VT_I4 ceGnChannelCount_Cnt ([in] VT_I4 NodeID, [in] VT_I4 ModuleIdx, [out] VT_PI4 ChannelCount)

DESCRIPTION

해당 원격 노드의 카운터 모듈에 대해 탐색된 카운터 채널(Counter Channel) 개수를 반환합니다.

PARAMETER

▶ *NodeID* : 원격 노드 ID. 탐색된 카운터 채널 개수를 얻어올 카운터 모듈이 속한 원격 노드 ID 를 설정합니다.

▶ *ModuleIdx* : 카운터 모듈 ID. 탐색된 카운터 채널 개수를 얻어올 카운터 모듈 ID 를 설정합니다.

▶ *ChannelCount* : 이 매개변수를 통하여 해당 카운터 모듈에 대해 탐색된 카운터 채널 개수를 반환합니다.

RETURN VALUE

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

SEE ALSO

ceGnChannelCount_Motion, ceGnChannelCount_Dio, ceGnChannelCount_Ai, ceGnChannelCount_Ao, ceGnChannelCount_Mdio, ceGnChannelCount_Ser

EXAMPLE

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

NAME

ceGnChannelCount_Ser

- 해당 시리얼 통신 모듈에 대해 탐색된 시리얼 통신 채널 개수 반환

INFORMATION	
1	General Function
!	VC++ (6, 7, 8)/VB
	BCB/Delphi
:	Level 1
J	위험 요소 없음

SYNOPSIS

r VT_I4 ceGnChannelCount_Ser ([in] VT_I4 NodeID, [in] VT_I4 ModuleIdx, [out] VT_PI4 ChannelCount)

DESCRIPTION

해당 원격 노드의 시리얼 통신 모듈에 대해 탐색된 시리얼 통신 채널 개수를 반환합니다.

PARAMETER

▶ *NodeID* : 원격 노드 ID. 탐색된 시리얼 통신 채널 개수를 얻어올 시리얼 통신 모듈이 속한 원격 노드 ID 를 설정합니다.

▶ *ModuleIdx* : 시리얼 통신 모듈 ID. 탐색된 시리얼 통신 채널 개수를 얻어올 시리얼 통신 모듈 ID 를 설정합니다.

▶ *ChannelCount* : 이 매개변수를 통하여 해당 시리얼 통신 모듈에 대해 탐색된 시리얼 통신 채널 개수를 반환합니다.

RETURN VALUE

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

SEE ALSO

ceGnChannelCount_Motion, ceGnChannelCount_Dio, ceGnChannelCount_Ai, ceGnChannelCount_Ao, ceGnChannelCount_Mdio, ceGnChannelCount_Ser

EXAMPLE

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

NAME

ceGnLocalAxis_Get

- 통합 모션 제어 축(Axis)에 대한 원격 노드 및 모션 모듈 정보 반환

INFORMATION	
1	General Function
!	VC++ (6, 7, 8)/VB
	BCB/Delphi
:	Level 1
K	위험 요소 없음

SYNOPSIS

r VT_I4 ceGnLocalAxis_Get ([in] VT_I4 Axis, [out] VT_PI4 NodeIP, [out] VT_PI4 NodeID, [out] VT_PI4 NodeInGlobal, [out] VT_PI4 ModuleIdx, [out] PI4 ModuleInCh)

DESCRIPTION

통합 모션 제어 축(Global Axis) 중 해당 모션 축에 대해 원격 노드 정보와 그 축이 속한 원격 노드 내의 상대적인 모션 축 번호, 그리고 모션 모듈 정보 및 모듈 내의 축 번호에 대한 정보를 반환합니다.

PARAMETER

▶ ***Axis*** : 모션 제어 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.

▶ ***NodeIP*** : 이 매개변수를 통하여 해당 모션 제어 축이 속한 원격 노드의 IP 를 반환합니다.

▶ ***NodeID*** : 이 매개변수를 통하여 해당 모션 제어 축이 속한 원격 노드의 ID 를 반환합니다.

▶ ***NodeInGlobal*** : 이 매개변수를 통하여 해당 원격 노드 내에 포함된 모션 모듈들에 대해 지정한 축이 갖는 상대적인 모션 축 번호를 반환합니다.

원격 노드 내의 모션 모듈들에 대한 상대적 모션 축 번호는 왜 필요한가요?

보간 제어는 대상 원격 노드 안의 축들로 제한되며, 축 그룹에 포함할 축 번호는 원격 노드 안에서의 제한된 통합 축 번호로 설정하여야 합니다. 보간 제어에 대한 자세한 설명은 '7.2 기본 보간 제어' 를 참고하여 주시기 바랍니다.

▶ ***ModuleIdx*** : 이 매개변수를 통하여 해당 모션 제어 축이 속한 모션 모듈의 ID 를 반환합니다.

▶ ***ModuleInCh*** : 이 매개변수를 통하여 해당 축이 속한 모션 모듈 내의 축 번호를 반환합니다.

RETURN VALUE

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

SEE ALSO

ceGnLocalDIO_Get, ceGnLocalAI_Get, ceGnLocalAO_Get, ceGnLocalMDIO_Get, ceGnLocalCNT_Get, ceGnLocalSER_Get

EXAMPLE

C/C++

```
#include “ceSDK.h”
#include “ceSDKDef.h”

void OnGetLocalChInfo ()
{
    long nNodeIP;                   // 해당 채널이 속한 원격 노드 IP 정보.
    long nNodeID;                   // 해당 채널이 속한 원격 노드 ID 정보.
    long nNodeInGlobal;             // 해당 원격 노드 내에서의 제한된 통합 채널 번호 정보.
    long nModuleIdx;                // 해당 채널이 속한 모듈 ID 정보.
    long nModuleInCh;               // 해당 채널이 속한 모듈 내에서의 채널 번호 정보.

    long nGlobalAxisNo = 1;         // 모션 1 번축 선택. 통합 축 번호를 의미.
    long nGlobalDiChNo = 2;         // DI 2 번 채널 선택. 통합 채널 번호를 의미.
    Long nGloablAiChNo = 1;         // AI 1 번 채널 선택. 통합 채널 번호를 의미.
    Long nGlobalAoChNo = 2;         // AO 2 번 채널 선택. 통합 채널 번호를 의미.
    Long nGlobalMdoChNo = 1;        // MDO 1 번 채널 선택. 통합 채널 번호를 의미.
    Long nGlobalCountChNo = 2;      // 카운터 2 번 채널 선택. 통합 채널 번호를 의미.
    Long nGlobalSerChNo = 1;        // 시리얼 통신 1 번 채널 선택. 통합 채널 번호를 의미.

    // 모션 1 번 축에 대해 원격 노드 및 모션 모듈 정보 반환.
    ceGnLocalAxis_Get ( nGlobalAxisNo, &nNodeIP,  &nNodeID,  &nNodeInGlobal, &nModuleIdx,
    &nModuleInCh );

    // DI 2 번 채널에 대해 원격 노드 및 DIO 모듈 정보 반환.
    ceGnLocalDIO_Get ( nGlobalDiChNo, &nNodeIP,  &nNodeID,  &nNodeInGlobal, &nModuleIdx,
    &nModuleInCh );

    // AI 1 번 채널에 대해 원격 노드 및 AI 모듈 정보 반환.
    ceGnLocalAI_Get ( nGlobalAiChNo, &nNodeIP,  &nNodeID,  &nNodeInGlobal, &nModuleIdx,
    &nModuleInCh );

    // AO 2 번 채널에 대해 원격 노드 및 AO 모듈 정보 반환.
    ceGnLocalAO_Get ( nGlobalAoChNo, &nNodeIP,  &nNodeID,  &nNodeInGlobal, &nModuleIdx,
    &nModuleInCh );
```

```
    // MDO 1 번 채널에 대해 원격 노드 및 모션 모듈 정보 반환.
    ceGnLocalMDIO_Get ( nGlobalMdoChNo, &nNodeIP,  &nNodeID,  &nNodeInGlobal, &nModuleIdx,
    &nModuleInCh );

    // 카운터 2 번 채널에 대해 원격 노드 및 카운터 모듈 정보 반환.
    ceGnLocalCNT_Get ( nGlobalCountChNo, &nNodeIP,  &nNodeID,  &nNodeInGlobal, &nModuleIdx,
    &nModuleInCh );

    // 시리얼 통신 1 번 채널에 대해 원격 노드 및 시리얼 통신 모듈 정보 반환.
    ceGnLocalSER_Get ( nGlobalCountChNo, &nNodeIP,  &nNodeID,  &nNodeInGlobal, &nModuleIdx,
    &nModuleInCh );

}
```

Visual Basic

```
Private Sub OnGetLocalChInfo ()

    Dim nNodeIP As Long, nNodeID As Long, nNodeInGlobal As Long
    Dim nModuleIdx As Long, nModuleInCh As Long

    Dim nGlobalAxisNo As Long, nGlobalDiChNo As Long, nGloablAiChNo As Long, nGlobalAoChNo As Long
    Dim nGlobalMdoChNo As Long, nGlobalCountChNo As Long, nGlobalSerChNo As Long

    nGlobalAxisNo = 1          ' 모션 1 번축 선택. 통합 축 번호를 의미.
    nGlobalDiChNo = 2          ' DI 2 번 채널 선택. 통합 채널 번호를 의미.
    nGloablAiChNo = 1          ' AI 1 번 채널 선택. 통합 채널 번호를 의미.
    nGlobalAoChNo = 2          ' AO 2 번 채널 선택. 통합 채널 번호를 의미.
    nGlobalMdoChNo = 1         ' MDO 1 번 채널 선택. 통합 채널 번호를 의미.
    nGlobalCountChNo = 2       ' 카운터 2 번 채널 선택. 통합 채널 번호를 의미.
    nGlobalSerialChNo = 1      ' 카운터 1 번 채널 선택. 통합 채널 번호를 의미.

    ' 모션 1 번 축에 대해 원격 노드 및 모션 모듈 정보 반환.
    Call ceGnLocalAxis_Get ( nGlobalAxisNo, nNodeIP, nNodeID, nNodeInGlobal, nModuleIdx, nModuleInCh )

    ' DI 2 번 채널에 대해 원격 노드 및 DIO 모듈 정보 반환.
    Call ceGnLocalDIO_Get ( nGlobalDiChNo, nNodeIP, nNodeID, nNodeInGlobal, nModuleIdx, nModuleInCh )

    ' AI 1 번 채널에 대해 원격 노드 및 AI 모듈 정보 반환.
    Call ceGnLocalAI_Get ( nGlobalAiChNo, nNodeIP, nNodeID, nNodeInGlobal, nModuleIdx, nModuleInCh )

    ' AO 2 번 채널에 대해 원격 노드 및 AO 모듈 정보 반환.
    Call ceGnLocalAO_Get ( nGlobalAoChNo, nNodeIP,  nNodeID,  nNodeInGlobal, nModuleIdx, nModuleInCh )

    ' MDO 1 번 채널에 대해 원격 노드 및 모션 모듈 정보 반환.
    Call ceGnLocalMDIO_Get ( nGlobalMdoChNo, nNodeIP, nNodeID, nNodeInGlobal, nModuleIdx, _
                             nModuleInCh )

    ' 카운터 2 번 채널에 대해 원격 노드 및 카운터 모듈 정보 반환.
    Call ceGnLocalCNT_Get ( nGlobalCountChNo, nNodeIP,  nNodeID,  nNodeInGlobal, nModuleIdx, _
                              nModuleInCh )
```

```
    ‘ 시리얼 통신 1 번 채널에 대해 원격 노드 및 카운터 모듈 정보 반환.
    Call ceGnLocalSER_Get ( nGlobalSerChNo, nNodeIP,  nNodeID,  nNodeInGlobal, nModuleIdx, _
                                nModuleInCh )

End Sub
```

Delphi

```
procedure OnGetLocalChInfo ();
var
    nNodeIP, nNodeID, nNodeInGlobal, nModuleIdx, nModuleInCh : LongInt;
    nGlobalAxisNo, nGlobalDiChNo, nGloablAiChNo, nGlobalAoChNo : LongInt;
    nGlobalMdoChNo, nGlobalCountChNo, nGlobalSerChNo : LongInt;

begin
    nGlobalAxisNo := 1;         ‘ 모션 1 번축 선택. 통합 축 번호를 의미.
    nGlobalDiChNo := 2;         ‘ DI 2 번 채널 선택. 통합 채널 번호를 의미.
    nGloablAiChNo := 1;         ‘ AI 1 번 채널 선택. 통합 채널 번호를 의미.
    nGlobalAoChNo := 2;         ‘ AO 2 번 채널 선택. 통합 채널 번호를 의미.
    nGlobalMdoChNo := 1;        ‘ MDO 1 번 채널 선택. 통합 채널 번호를 의미.
    nGlobalCountChNo := 2;      ‘ 카운터 2 번 채널 선택. 통합 채널 번호를 의미.
    nGlobalSerChNo := 2;        ‘ 시리얼 통신 2 번 채널 선택. 통합 채널 번호를 의미.


    // 모션 1 번 축에 대해 원격 노드 및 모션 모듈 정보 반환.
    ceGnLocalAxis_Get ( nGlobalAxisNo, @nNodeIP, @nNodeID, @nNodeInGlobal, @nModuleIdx,
                        @nModuleInCh );

    // DI 2 번 채널에 대해 원격 노드 및 DIO 모듈 정보 반환.
    ceGnLocalDIO_Get ( nGlobalDiChNo, @nNodeIP, @nNodeID, @nNodeInGlobal, @nModuleIdx,
                        @nModuleInCh );

    // AI 1 번 채널에 대해 원격 노드 및 AI 모듈 정보 반환.
    ceGnLocalAI_Get ( nGlobalAiChNo, @nNodeIP, @nNodeID, @nNodeInGlobal, @nModuleIdx,
                        @nModuleInCh );

    // AO 2 번 채널에 대해 원격 노드 및 AO 모듈 정보 반환.
    ceGnLocalAO_Get ( nGlobalAoChNo, @nNodeIP,  @nNodeID,  @nNodeInGlobal, @nModuleIdx,
                        @nModuleInCh );

    // MDO 1 번 채널에 대해 원격 노드 및 모션 모듈 정보 반환.
    ceGnLocalMDIO_Get ( nGlobalMdoChNo, @nNodeIP, @nNodeID, @nNodeInGlobal, @nModuleIdx,
                        @nModuleInCh );

    // 카운터 2 번 채널에 대해 원격 노드 및 카운터 모듈 정보 반환.
    ceGnLocalCNT_Get ( nGlobalCountChNo, @nNodeIP,  @nNodeID,  @nNodeInGlobal, @nModuleIdx,
                        @nModuleInCh );

    // 시리얼 통신 1 번 채널에 대해 원격 노드 및 카운터 모듈 정보 반환.
    ceGnLocalSER_Get ( nGlobalSerChNo, @nNodeIP,  @nNodeID,  @nNodeInGlobal, @nModuleIdx,
                        @nModuleInCh );

End;
```

NAME

ceGnLocalDIO_Get

- 통합 디지털 I/O(DIO) 채널에 대한 원격 노드 및 DIO 모듈 정보 반환

INFORMATION	
1	General Function
!	VC++ (6, 7, 8)/VB
	BCB/Delphi
:	Level 1
K	위험 요소 없음

SYNOPSIS

r VT_I4 ceGnLocalDIO_Get ([in] VT_I4 Channel, [out] VT_PI4 NodeIP, [out] VT_PI4 NodeID, [out] VT_PI4 NodeInGlobal, [out] VT_PI4 ModuleIdx, [out] PI4 ModuleInCh)

DESCRIPTION

통합 DIO 채널(Global DIO Channel) 중 해당 DIO 채널에 대해 원격 노드 정보와 그 채널이 속한 원격 노드 내의 상대적인 DIO 채널 번호, 그리고 DIO 모듈 정보 및 모듈 내의 채널 번호에 대한 정보를 반환합니다.

PARAMETER

▶ *Channel* : DIO 채널 번호. 통합 채널로 관리되는 채널 번호를 의미하며, 상수 값으로 0 (Zero Based) 이상, (최대 통합 채널 수 -1) 이하의 값을 채널 번호로 설정할 수 있습니다.

▶ *NodeIP* : 이 매개변수를 통하여 해당 DIO 채널이 속한 원격 노드의 IP 를 반환합니다.

▶ *NodeID* : 이 매개변수를 통하여 해당 DIO 채널이 속한 원격 노드의 ID 를 반환합니다.

▶ *NodeInGlobal* : 이 매개변수를 통하여 해당 원격 노드 내에 포함된 DIO 모듈들에 대해 지정한 채널이 갖는 상대적인 DIO 채널 번호를 반환합니다.

▶ *ModuleIdx* : 이 매개변수를 통하여 해당 채널이 속한 DIO 모듈의 ID 를 반환합니다.

▶ *ModuleInCh* : 이 매개변수를 통하여 해당 채널이 속한 DIO 모듈 내의 채널 번호를 반환합니다.

RETURN VALUE

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

SEE ALSO

ceGnLocalAxis_Get, ceGnLocalAI_Get, ceGnLocalAO_Get, ceGnLocalMDIO_Get, ceGnLocalCNT_Get, ceGnLocalSER_Get

EXAMPLE

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

NAME

ceGnLocalAI_Get

- 통합 아날로그 입력(AI) 채널에 대한 원격 노드 및 AI 모듈 정보 반환

INFORMATION	
1	General Function
!	VC++ (6, 7, 8)/VB
	BCB/Delphi
:	Level 1
K	위험 요소 없음

SYNOPSIS

r VT_I4 ceGnLocalAI_Get ([in] VT_I4 Channel, [out] VT_PI4 NodeIP, [out] VT_PI4 NodeID, [out] VT_PI4 NodeInGlobal, [out] VT_PI4 ModuleIdx, [out] PI4 ModuleInCh)

DESCRIPTION

통합 아날로그 입력 채널(Global Analog Input Channel) 중 해당 AI 채널에 대한 원격 노드 정보와 그 채널이 속한 원격 노드 내의 상대적인 채널 번호, 그리고 AI 모듈 정보 및 모듈 내의 채널 번호에 대한 정보를 반환합니다.

PARAMETER

▶ ***Channel*** : AI 채널 번호. 통합 채널로 관리되는 채널 번호를 의미하며, 상수 값으로 0 (Zero Based) 이상, (최대 통합 채널 수 -1) 이하의 값을 채널 번호로 설정할 수 있습니다.

▶ ***NodeIP*** : 이 매개변수를 통하여 해당 AI 채널이 속한 원격 노드의 IP 를 반환합니다.

▶ ***NodeID*** : 이 매개변수를 통하여 해당 AI 채널이 속한 원격 노드의 ID 를 반환합니다.

▶ ***NodeInGlobal*** : 이 매개변수를 통하여 해당 원격 노드 내에 포함된 AI 모듈들에 대해 지정한 채널이 갖는 상대적인 AI 채널 번호를 반환합니다.

▶ ***ModuleIdx*** : 이 매개변수를 통하여 해당 채널이 속한 AI 모듈의 ID 를 반환합니다.

▶ ***ModuleInCh*** : 이 매개변수를 통하여 해당 채널이 속한 AI 모듈 내의 채널 번호를 반환합니다.

RETURN VALUE

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

SEE ALSO

ceGnLocalAxis_Get, ceGnLocalDIO_Get, ceGnLocalAO_Get, ceGnLocalMDIO_Get, ceGnLocalCNT_Get, ceGnLocalSER_Get

EXAMPLE

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

NAME

ceGnLocalAO_Get

- 통합 아날로그 출력(AO) 채널에 대한 원격 노드 및 AO 모듈 정보 반환

INFORMATION	
1	General Function
!	VC++ (6, 7, 8)/VB
	BCB/Delphi
:	Level 1
K	위험 요소 없음

SYNOPSIS

r VT_I4 ceGnLocalAO_Get ([in] VT_I4 Channel, [out] VT_PI4 NodeIP, [out] VT_PI4 NodeID, [out] VT_PI4 NodeInGlobal, [out] VT_PI4 ModuleIdx, [out] PI4 ModuleInCh)

DESCRIPTION

통합 아날로그 출력 채널(Global Analog Output Channel) 중 해당 AO 채널에 대한 원격 노드 정보와 그 채널이 속한 원격 노드 내의 상대적인 채널 번호, 그리고 AO 모듈 정보 및 모듈 내의 채널 번호에 대한 정보를 반환합니다.

PARAMETER

▶ ***Channel*** : AO 채널 번호. 통합 채널로 관리되는 채널 번호를 의미하며, 상수 값으로 0 (Zero Based) 이상, (최대 통합 채널 수 -1) 이하의 값을 채널 번호로 설정할 수 있습니다.

▶ ***NodeIP*** : 이 매개변수를 통하여 해당 AO 채널이 속한 원격 노드의 IP 를 반환합니다.

▶ ***NodeID*** : 이 매개변수를 통하여 해당 AO 채널이 속한 원격 노드의 ID 를 반환합니다.

▶ ***NodeInGlobal*** : 이 매개변수를 통하여 해당 원격 노드 내에 포함된 AO 모듈들에 대해 지정한 채널이 갖는 상대적인 AO 채널 번호를 반환합니다.

▶ ***ModuleIdx*** : 이 매개변수를 통하여 해당 채널이 속한 AO 모듈의 ID 를 반환합니다.

▶ ***ModuleInCh*** : 이 매개변수를 통하여 해당 채널이 속한 AO 모듈 내의 채널 번호를 반환합니다.

RETURN VALUE

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

SEE ALSO

ceGnLocalAxis_Get, ceGnLocalDIO_Get, ceGnLocalAI_Get, cceGnLocalMDIO_Get, eGnLocalCNT_Get, ceGnLocalSER_Get

EXAMPLE

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

NAME

ceGnLocalMDIO_Get

- 통합 모션 디지털 I/O(MDIO) 채널에 대한 원격 노드 및 모션 모듈 정보 반환

INFORMATION	
1	General Function
!	VC++ (6, 7, 8)/VB BCB/Delphi
:	Level 1
K	위험 요소 없음

SYNOPSIS

r VT_I4 ceGnLocalMDIO_Get ([in] VT_I4 Channel, [out] VT_PI4 NodeIP, [out] VT_PI4 NodeID, [out] VT_PI4 NodeInGlobal, [out] VT_PI4 ModuleIdx, [out] PI4 ModuleInCh)

DESCRIPTION

통합 모션 디지털 I/O 채널(Global Motion Digital I/O Channel) 중 해당 MDIO 채널에 대한 원격 노드 정보와 그 채널이 속한 원격 노드 내의 상대적인 채널 번호, 그리고 모션 모듈 정보 및 모듈 내의 채널 번호에 대한 정보를 반환합니다.

PARAMETER

▶ *Channel* : MDIO 채널 번호. 통합 채널로 관리되는 채널 번호를 의미하며, 상수 값으로 0 (Zero Based) 이상, (최대 통합 채널 수 -1) 이하의 값을 채널 번호로 설정할 수 있습니다.

▶ *NodeIP* : 이 매개변수를 통하여 해당 MDIO 채널이 속한 원격 노드의 IP 를 반환합니다.

▶ *NodeID* : 이 매개변수를 통하여 해당 MDIO 채널이 속한 원격 노드의 ID 를 반환합니다.

▶ *NodeInGlobal* : 이 매개변수를 통하여 해당 원격 노드 내에 포함된 모션 모듈들에 대해 지정한 MDIO 채널이 갖는 상대적인 채널 번호를 반환합니다.

▶ *ModuleIdx* : 이 매개변수를 통하여 해당 MDIO 채널이 속한 모션 모듈 ID 를 반환합니다.

▶ *ModuleInCh* : 이 매개변수를 통하여 해당 MDIO 채널이 속한 모션 모듈 내의 채널 번호를 반환합니다.

RETURN VALUE

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

SEE ALSO

ceGnLocalAxis_Get, ceGnLocalDIO_Get, ceGnLocalAI_Get, ceGnLocalAO_Get, ceGnLocalCNT_Get, ceGnLocalSER_Get

EXAMPLE

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

NAME

ceGnLocalCNT_Get

- 통합 카운터(Counter) 채널에 대한 원격 노드 및 카운터 모듈 정보 반환

INFORMATION	
1	General Function
!	VC++ (6, 7, 8)/VB
	BCB/Delphi
:	Level 1
K	위험 요소 없음

SYNOPSIS

r VT_I4 ceGnLocalCNT_Get ([in] VT_I4 Channel, [out] VT_PI4 NodeIP, [out] VT_PI4 NodeID, [out] VT_PI4 NodeInGlobal, [out] VT_PI4 ModuleIdx, [out] PI4 ModuleInCh)

DESCRIPTION

통합 카운터 채널(Global Counter Channel) 중 해당 카운터 채널에 대한 원격 노드 정보와 그 채널이 속한 원격 노드 내의 상대적인 채널 번호, 그리고 카운터 모듈 정보 및 모듈 내의 채널 번호에 대한 정보를 반환합니다.

PARAMETER

▶ *Channel* : 카운터 채널 번호. 통합 채널로 관리되는 채널 번호를 의미하며, 상수 값으로 0 (Zero Based) 이상, (최대 통합 채널 수 -1) 이하의 값을 채널 번호로 설정할 수 있습니다.

▶ *NodeIP* : 이 매개변수를 통하여 해당 채널이 속한 원격 노드의 IP 를 반환합니다.

▶ *NodeID* : 이 매개변수를 통하여 해당 채널이 속한 원격 노드의 ID 를 반환합니다.

▶ *NodeInGlobal* : 이 매개변수를 통하여 해당 원격 노드 내에 포함된 카운터 모듈들에 대해 지정한 채널이 갖는 상대적인 채널 번호를 반환합니다.

▶ *ModuleIdx* : 이 매개변수를 통하여 해당 채널이 속한 카운터 모듈의 ID 를 반환합니다.

▶ *ModuleInCh* : 이 매개변수를 통하여 해당 채널이 속한 카운터 모듈 내의 채널 번호를 반환합니다.

RETURN VALUE

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

SEE ALSO

ceGnLocalAxis_Get, ceGnLocalDIO_Get, ceGnLocalAI_Get, ceGnLocalAO_Get, ceGnLocalMDIO_Get, ceGnLocalSER_Get

EXAMPLE

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

NAME

ceGnLocalSER_Get

- 통합 시리얼 통신 채널에 대한 원격 노드 및 시리얼 통신 모듈 정보 반환

INFORMATION	
1	General Function
!	VC++ (6, 7, 8)/VB
	BCB/Delphi
:	Level 1
K	위험 요소 없음

SYNOPSIS

r VT_I4 ceGnLocalSER_Get ([in] VT_I4 Channel, [out] VT_PI4 NodeIP, [out] VT_PI4 NodeID, [out] VT_PI4 NodeInGlobal, [out] VT_PI4 ModuleIdx, [out] PI4 ModuleInCh)

DESCRIPTION

통합 시리얼 통신 채널(Global Serial Communication Channel) 중 해당 시리얼 통신 채널에 대한 원격 노드 정보와 그 채널이 속한 원격 노드 내의 상대적인 채널 번호, 그리고 시리얼 통신 모듈 정보 및 모듈 내의 채널 번호에 대한 정보를 반환합니다.

PARAMETER

▶ *Channel* : 시리얼 통신 채널 번호. 통합 채널로 관리되는 채널 번호를 의미하며, 상수 값으로 0 (Zero Based) 이상, (최대 통합 채널 수 -1) 이하의 값을 채널 번호로 설정할 수 있습니다.

▶ *NodeIP* : 이 매개변수를 통하여 해당 채널이 속한 원격 노드의 IP 를 반환합니다.

▶ *NodeID* : 이 매개변수를 통하여 해당 채널이 속한 원격 노드의 ID 를 반환합니다.

▶ *NodeInGlobal* : 이 매개변수를 통하여 해당 원격 노드 내에 포함된 시리얼 통신 모듈들에 대해 지정한 채널이 갖는 상대적인 채널 번호를 반환합니다.

▶ *ModuleIdx* : 이 매개변수를 통하여 해당 채널이 속한 시리얼 통신 모듈의 ID 를 반환합니다.

▶ *ModuleInCh* : 이 매개변수를 통하여 해당 채널이 속한 시리얼 통신 모듈 내의 채널 번호를 반환합니다.

RETURN VALUE

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

SEE ALSO

ceGnLocalAxis_Get, ceGnLocalDIO_Get, ceGnLocalAI_Get, ceGnLocalAO_Get, ceGnLocalMDIO_Get, ceGnLocalCNT_Get

EXAMPLE

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

NAME

ceGnGlobalAxis_Get

- 원격 노드 및 모션 모듈, 모듈 내의 축 번호를 통하여 통합(Global) 축 번호를 반환

INFORMATION	
1	General Function
!	VC++ (6, 7, 8)/VB
	BCB/Delphi
:	Level 1
K	위험 요소 없음

SYNOPSIS

r VT_I4 ceGnGlobalAxis_Get ([in] VT_I4 NodeID, [in] VT_I4 ModuleIdx, [in] VT_I4 ModuleInCh, [out] VT_PI4 GlobalAxis)

DESCRIPTION

원격 노드 및 모션 모듈, 모듈 내의 축 번호를 통하여 전체 원격 노드의 통합 모션 제어 축 번호(Global Axis Number)를 반환합니다.

PARAMETER

▶ *NodeID* : 원격 노드 ID. 통합 모션 제어 축 번호를 얻어올 모션 모듈이 속한 원격 노드 ID 를 설정합니다.

▶ *ModuleIdx* : 모션 모듈 ID. 통합 모션 제어 축 번호를 얻어올 모션 모듈의 ID 를 설정합니다.

▶ *ModuleInCh* : 모션 모듈 내의 채널 번호. 통합 모션 제어 축 번호를 얻어올 모션 모듈 내의 채널 번호를 설정합니다.

▶ *GlobalAxis* : 이 매개변수를 통하여 통합 모션 제어 축 번호를 반환합니다.

RETURN VALUE

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

SEE ALSO

ceGnGlobalDIO_Get, ceGnGlobalAI_Get, ceGnGlobalAO_Get, ceGnGlobalMDIO_Get, ceGnGlobalCNT_Get, ceGnGlobalSER_Get

EXAMPLE

C/C++

```
#include “ceSDK.h”
#include “ceSDKDef.h”

void OnGetLocalChInfo ()
{
    long nNodeID = 1;                   // 해당 채널이 속한 원격 노드 ID.

    long nMotModuleIdx = 1;             // 해당 모션 모듈 ID
    long nDioModuleIdx = 2;             // 해당 DIO 모듈 ID
    long nAiModuleIdx = 3;              // 해당 AI 모듈 ID
    long nAoModuleIdx = 4;              // 해당 AO 모듈 ID
    long nCountModuleIdx = 5;           // 해당 카운터 모듈 ID
    long nSerModuleIdx = 6;             // 해당 시리얼 통신 모듈 ID

    long nMotModuleInCh = 1;            // 모션 모듈 내에서의 모션 축 번호.
    long nDiModuleInCh = 1;             // DIO 모듈 내에서의 DI 채널 번호.
    long nAiModuleInCh = 1;             // AI 모듈 내에서의 AI 채널 번호.
    long nAoModuleInCh = 1;             // AO 듈 내에서의 AO 채널 번호.
    long nMdoModuleInCh = 1;            // 모션 모듈 내에서의 Mdo 채널 번호.
    long nCountModuleInCh = 1;          // 카운터 모듈 내에서의 카운터 채널 번호.
    long nSerModuleInCh = 1;            // 시리얼 통신 모듈 내에서의 시리얼 통신 채널 번호.

    // 통합 채널 번호 정보
    long nGlobalAxisNo, nGlobalDiChNo, nGloablAiChNo, nGlobalAoChNo,  nGlobalMdoChNo,
    nGlobalCountChNo, nGlobalSerChNo;

    // 원격 노드 및 모션 모듈, 모듈 내 축 번호를 통하여 통합 축 번호 정보 반환.
    ceGnGlobalAxis_Get ( nNodeID, nMotModuleIdx, nMotModuleInCh,  &nGlobalAxisNo );

    // 원격 노드 및 DIO 모듈, 모듈 내 채널 번호를 통하여 통합 채널 번호 정보 반환.
    ceGnGlobalDIO_Get  (nNodeID, nDiModuleIdx, nDiModuleInCh,  &nGlobalDiChNo );

    // 원격 노드 및 AI 모듈, 모듈 내 채널 번호를 통하여 통합 채널 번호 정보 반환.
    ceGnGlobalAI_Get ( nNodeID, nAiModuleIdx, nAiModuleInCh,  &nGlobalAiChNo );

    // 원격 노드 및 AO 모듈, 모듈 내 채널 번호를 통하여 통합 채널 번호 정보 반환.
    ceGnGlobalAO_Get ( nNodeID, nAoModuleIdx, nAoModuleInCh,  &nGlobalAoChNo );

    // 원격 노드 및 모션 모듈, 모듈 내 MDO 채널 번호를 통하여 통합 채널 번호 정보 반환.
    ceGnGlobalMDIO_Get ( nNodeID, nMotModuleIdx, nMdoModuleInCh,  &nGlobalMdoChNo );

    // 원격 노드 및 카운터 모듈, 모듈 내 채널 번호를 통하여 통합 채널 번호 정보 반환.
    ceGnGlobalCNT_Get ( nNodeID, nCountModuleIdx, nCountModuleInCh,  &nGlobalCountChNo );

    // 원격 노드 및 시리얼 통신 모듈, 모듈 내 채널 번호를 통하여 통합 채널 번호 정보 반환.
    ceGnGlobalSER_Get ( nNodeID, nSerModuleIdx, nSerModuleInCh,  &nGlobalSerChNo );
}
```

Visual Basic

```
Private Sub OnGetLocalChInfo ()

    Dim nNodeID As Long      ' 해당 원격 노드 ID

    ' 해당 모듈 ID
    Dim nMotModuleIdx As Long, nDioModuleIdx As Long, nAiModuleIdx As Long
    Dim nAoModuleIdx As Long, nCountModuleIdx As Long, nSerModuleIdx As Long

    ' 해당 모듈 내에서의 채널 번호 정보
    Dim nMotModuleInCh As Long, nDiModuleInCh As Long, nAiModuleInCh As Long, nAoModuleInCh As Long
    Dim nMdoModuleInCh As Long, nCountModuleInCh As Long, nSerModuleInCh As Long

    ' 통합 채널 번호 정보 반환
    Dim nGlobalAxisNo As Long, nGlobalDiChNo As Long, nGloablAiChNo As Long, nGlobalAoChNo As Long
    Dim nGlobalMdoChNo As Long, nGlobalCountChNo As Long, nGlobalSerChNo As Long


    nNodeID = 1              ' 해당 채널이 속한 원격 노드 ID.

    nMotModuleIdx = 1        ' 해당 모션 모듈 ID
    nDioModuleIdx = 2        ' 해당 DIO 모듈 ID
    nAiModuleIdx = 3         ' 해당 AI 모듈 ID
    nAoModuleIdx = 4         ' 해당 AO 모듈 ID
    nCountModuleIdx = 5      ' 해당 카운터 모듈 ID
    nSerModuleIdx = 6        ' 해당 시리얼 통신 모듈 ID

    nMotModuleInCh = 1       ' 모션 모듈 내에서의 모션 축 번호.
    nDiModuleInCh = 1        ' DIO 모듈 내에서의 DI 채널 번호.
    nAiModuleInCh = 1        ' AI 모듈 내에서의 AI 채널 번호.
    nAoModuleInCh = 1        ' AO 듈 내에서의 AO 채널 번호.
    nMdoModuleInCh = 1       ' 모션 모듈 내에서의 Mdo 채널 번호.
    nCountModuleInCh = 1     ' 카운터 모듈 내에서의 카운터 채널 번호.
    nSerModuleInCh = 1       ' 시리얼 통신 모듈 내에서의 시리얼 통신 채널 번호.


    ' 원격 노드 및 모션 모듈, 모듈 내 축 번호를 통하여 통합 축 번호 정보 반환.
    Call ceGnGlobalAxis_Get ( nNodeID, nMotModuleIdx, nMotModuleInCh, nGlobalAxisNo )

    ' 원격 노드 및 DIO 모듈, 모듈 내 채널 번호를 통하여 통합 채널 번호 정보 반환.
    Call ceGnGlobalDIO_Get  (nNodeID, nDiModuleIdx, nDiModuleInCh, nGlobalDiChNo )

    ' 원격 노드 및 AI 모듈, 모듈 내 채널 번호를 통하여 통합 채널 번호 정보 반환.
    Call ceGnGlobalAI_Get ( nNodeID, nAiModuleIdx, nAiModuleInCh, nGlobalAiChNo )

    ' 원격 노드 및 AO 모듈, 모듈 내 채널 번호를 통하여 통합 채널 번호 정보 반환.
    Call ceGnGlobalAO_Get ( nNodeID, nAoModuleIdx, nAoModuleInCh, nGlobalAoChNo )
```

```
    ‘ 원격 노드 및 모션 모듈, 모듈 내 MDO 채널 번호를 통하여 통합 채널 번호 정보 반환.
    Call ceGnGlobalMDIO_Get ( nNodeID, nMotModuleIdx, nMdoModuleInCh, nGlobalMdoChNo )

    ‘ 원격 노드 및 카운터 모듈, 모듈 내 채널 번호를 통하여 통합 채널 번호 정보 반환.
    Call ceGnGlobalCNT_Get ( nNodeID, nCountModuleIdx, nCountModuleInCh, nGlobalCountChNo )

    ‘ 원격 노드 및 시리얼 통신 모듈, 모듈 내 채널 번호를 통하여 통합 채널 번호 정보 반환.
    Call ceGnGlobalSER_Get ( nNodeID, nSerModuleIdx, nSerModuleInCh, nGlobalSerChNo )

End Sub
```

Delphi

```
procedure OnGetLocalChInfo ();
var
    nNodeID : LongInt           // 해당 원격 노드 ID

    // 해당 모듈 ID
    nMotModuleIdx, nDioModuleIdx, nAiModuleIdx, nAoModuleIdx, nCountModuleIdx, nSerModuleIdx : LongInt;

    // 해당 모듈 내에서의 채널 번호 정보
    nMotModuleInCh, nDiModuleInCh, nAiModuleInCh, nAoModuleInCh : LongInt;
    nMdoModuleInCh, nCountModuleInCh, nSerModuleInCh : LongInt;

    // 통합 채널 번호 정보 반환
    nGlobalAxisNo, nGlobalDiChNo, nGloablAiChNo, nGlobalAoChNo : LongInt;
    nGlobalMdoChNo, nGlobalCountChNo, nGlobalSerChNo : LongInt;

begin
    nNodeID := 1;               // 해당 채널이 속한 원격 노드 ID.

    nMotModuleIdx := 1;         // 해당 모션 모듈 ID
    nDioModuleIdx := 2;         // 해당 DIO 모듈 ID
    nAiModuleIdx := 3;          // 해당 AI 모듈 ID
    nAoModuleIdx := 4;          // 해당 AO 모듈 ID
    nCountModuleIdx := 5;       // 해당 카운터 모듈 ID
    nSerModuleIdx := 6;         // 해당 시리얼 통신 모듈 ID

    nMotModuleInCh := 1;        // 모션 모듈 내에서의 모션 축 번호.
    nDiModuleInCh := 1;         // DIO 모듈 내에서의 DI 채널 번호.
    nAiModuleInCh := 1;         // AI 모듈 내에서의 AI 채널 번호.
    nAoModuleInCh := 1;         // AO 듈 내에서의 AO 채널 번호.
    nMdoModuleInCh := 1;        // 모션 모듈 내에서의 Mdo 채널 번호.
    nCountModuleInCh := 1;      // 카운터 모듈 내에서의 카운터 채널 번호.
    nSerModuleInCh := 1;        // 시리얼 통신 모듈 내에서의 시리얼 통신 채널 번호.


    // 원격 노드 및 모션 모듈, 모듈 내 축 번호를 통하여 통합 축 번호 정보 반환.
    ceGnGlobalAxis_Get ( nNodeID, nMotModuleIdx, nMotModuleInCh, @nGlobalAxisNo );
```

```
    // 원격 노드 및 DIO 모듈, 모듈 내 채널 번호를 통하여 통합 채널 번호 정보 반환.
    ceGnGlobalDIO_Get  (nNodeID, nDiModuleIdx, nDiModuleInCh, @nGlobalDiChNo );

    // 원격 노드 및 AI 모듈, 모듈 내 채널 번호를 통하여 통합 채널 번호 정보 반환.
    ceGnGlobalAI_Get ( nNodeID, nAiModuleIdx, nAiModuleInCh, @nGlobalAiChNo );

    // 원격 노드 및 AO 모듈, 모듈 내 채널 번호를 통하여 통합 채널 번호 정보 반환.
    ceGnGlobalAO_Get ( nNodeID, nAoModuleIdx, nAoModuleInCh, @nGlobalAoChNo );

    // 원격 노드 및 모션 모듈, 모듈 내 MDO 채널 번호를 통하여 통합 채널 번호 정보 반환.
    ceGnGlobalMDIO_Get ( nNodeID, nMotModuleIdx, nMdoModuleInCh, @nGlobalMdoChNo );

    // 원격 노드 및 카운터 모듈, 모듈 내 채널 번호를 통하여 통합 채널 번호 정보 반환.
    ceGnGlobalCNT_Get ( nNodeID, nCountModuleIdx, nCountModuleInCh, @nGlobalCountChNo );

    // 원격 노드 및 시리얼 통신 모듈, 모듈 내 채널 번호를 통하여 통합 채널 번호 정보 반환.
    ceGnGlobalSER_Get ( nNodeID, nSerModuleIdx, nSerModuleInCh, @nGlobalSerChNo );

end
```

NAME

ceGnGlobalDIO_Get

- 원격 노드 및 디지털 I/O 모듈, 모듈 내의 채널 번호를 통하여 통합(Global) 채널 번호를 반환

INFORMATION	
1	General Function
!	VC++ (6, 7, 8)/VB
	BCB/Delphi
:	Level 1
K	위험 요소 없음

SYNOPSIS

r VT_I4 ceGnGlobalDIO_Get ([in] VT_I4 NodeID, [in] VT_I4 ModuleIdx, [in] VT_I4 ModuleInCh, [out] VT_PI4 GlobalDIO)

DESCRIPTION

원격 노드 및 디지털 I/O 모듈, 모듈 내의 DIO 채널 번호를 통하여 전체 원격 노드의 통합 DIO 채널 번호(Global DIO Channel Number)를 반환합니다.

PARAMETER

▶ ***NodeID*** : 원격 노드 ID. 통합 DIO 채널 번호를 얻어올 디지털 I/O 모듈이 속한 원격 노드 ID 를 설정합니다.

▶ ***ModuleIdx*** : 디지털 I/O 모듈 ID. 통합 DIO 채널 번호를 얻어올 디지털 I/O 모듈의 ID 를 설정합니다.

▶ ***ModuleInCh*** : 디지털 I/O 모듈 내의 채널 번호. 통합 DIO 채널 번호를 얻어올 디지털 I/O 모듈 내의 채널 번호를 설정합니다.

▶ ***GlobalDIO*** : 이 매개변수를 통하여 통합 DIO 채널 번호를 반환합니다.

RETURN VALUE

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

SEE ALSO

ceGnGlobalAxis_Get, ceGnGlobalAI_Get, ceGnGlobalAO_Get, ceGnGlobalMDIO_Get, ceGnGlobalCNT_Get, ceGnGlobalSER_Get

EXAMPLE

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

NAME

ceGnGlobalAI_Get

- 원격 노드 및 AI 모듈, 모듈 내의 채널 번호를 통하여 통합(Global) 채널 번호를 반환

INFORMATION	
1	General Function
!	VC++ (6, 7, 8)/VB
	BCB/Delphi
:	Level 1
K	위험 요소 없음

SYNOPSIS

r VT_I4 ceGnGlobalAI_Get ([in] VT_I4 NodeID, [in] VT_I4 ModuleIdx,
[in] VT_I4 ModuleInCh, [out] VT_PI4 GlobalAI)

DESCRIPTION

원격 노드 및 아날로그 입력 모듈, 모듈 내의 AI 채널 번호를 통하여 전체 원격 노드의 통합 AI 채널 번호(Global AI Channel Number)를 반환합니다.

PARAMETER

▶ *NodeID* : 원격 노드 ID. 통합 AI 채널 번호를 얻어올 아날로그 입력 모듈이 속한 원격 노드 ID 를 설정합니다.

▶ *ModuleIdx* : 아날로그 입력 모듈 ID. 통합 AI 채널 번호를 얻어올 아날로그 입력 모듈의 ID 를 설정합니다.

▶ *ModuleInCh* : 아날로그 입력 모듈 내의 채널 번호. 통합 AI 채널 번호를 얻어올 아날로그 입력 모듈 내의 채널 번호를 설정합니다.

▶ *GlobalAI* : 이 매개변수를 통하여 통합 AI 채널 번호를 반환합니다.

RETURN VALUE

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

SEE ALSO

ceGnGlobalAxis_Get, ceGnGlobalDIO_Get, ceGnGlobalAO_Get, ceGnGlobalMDIO_Get, ceGnGlobalCNT_Get, ceGnGlobalSER_Get

EXAMPLE

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

NAME

ceGnGlobalAO_Get

- 원격 노드 및 AO 모듈, 모듈 내의 채널 번호를 통하여 통합(Global) 채널 번호를 반환

INFORMATION	
1	General Function
!	VC++ (6, 7, 8)/VB
	BCB/Delphi
:	Level 1
K	위험 요소 없음

SYNOPSIS

r VT_I4 ceGnGlobalAO_Get ([in] VT_I4 NodeID, [in] VT_I4 ModuleIdx, [in] VT_I4 ModuleInCh, [out] VT_PI4 GlobalAO)

DESCRIPTION

원격 노드 및 아날로그 출력 모듈, 모듈 내의 AO 채널 번호를 통하여 전체 원격 노드의 통합 AO 채널 번호(Global AO Channel Number)를 반환합니다.

PARAMETER

▶ ***NodeID*** : 원격 노드 ID. 통합 AO 채널 번호를 얻어올 아날로그 출력 모듈이 속한 원격 노드 ID 를 설정합니다.

▶ ***ModuleIdx*** : 아날로그 출력 모듈 ID. 통합 AO 채널 번호를 얻어올 아날로그 출력 모듈의 ID 를 설정합니다.

▶ ***ModuleInCh*** : 아날로그 출력 모듈 내의 채널 번호. 통합 AO 채널 번호를 얻어올 아날로그 출력 모듈의 채널 번호를 설정합니다.

▶ ***GlobalAO*** : 이 매개변수를 통하여 통합 AO 채널 번호를 반환합니다.

RETURN VALUE

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

SEE ALSO

ceGnGlobalAxis_Get, ceGnGlobalDIO_Get, ceGnGlobalAI_Get, ceGnGlobalMDIO_Get, ceGnGlobalCNT_Get, ceGnGlobalSER_Get

EXAMPLE

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

NAME

ceGnGlobalMDIO_Get

- 원격 노드 및 모션 모듈, 모듈 내의 MDIO 채널 번호를 통하여 통합(Global) 채널 번호를 반환

INFORMATION	
1	General Function
!	VC++ (6, 7, 8)/VB
	BCB/Delphi
:	Level 1
K	위험 요소 없음

SYNOPSIS

r VT_I4 ceGnGlobalMDIO_Get ([in] VT_I4 NodeID, [in] VT_I4 ModuleIdx, [in] VT_I4 ModuleInCh, [out] VT_PI4 GlobalMDIO)

DESCRIPTION

원격 노드 및 모션 모듈, 모듈 내의 MDIO 채널 번호를 통하여 전체 원격 노드의 통합 MDIO 채널 번호(Global MDIO Channel Number)를 반환합니다.

PARAMETER

▶ *NodeID* : 원격 노드 ID. 통합 MDIO 채널 번호를 얻어올 모션 모듈이 속한 원격 노드 ID 를 설정합니다.

▶ *ModuleIdx* : 모션 모듈 ID. 통합 MDIO 채널 번호를 얻어올 모션 모듈의 ID 를 설정합니다.

▶ *ModuleInCh* : 모션 모듈내의 MDIO 채널 번호. 통합 MDIO 채널 번호를 얻어올 모션 모듈 내의 MDIO 채널 번호를 설정합니다.

▶ *GlobalMDIO* : 이 매개변수를 통하여 통합 MDIO 채널 번호를 반환합니다.

RETURN VALUE

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

SEE ALSO

ceGnGlobalAxis_Get, ceGnGlobalDIO_Get, ceGnGlobalAI_Get, ceGnGlobalAO_Get, ceGnGlobalCNT_Get, ceGnGlobalSER_Get

EXAMPLE

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

NAME

ceGnGlobalCNT_Get

- 원격 노드 및 카운터 모듈, 모듈 내의 채널 번호를 통하여 통합(Global) 채널 번호를 반환

INFORMATION	
1	General Function
!	VC++ (6, 7, 8)/VB
	BCB/Delphi
:	Level 1
K	위험 요소 없음

SYNOPSIS

r VT_I4 ceGnGlobalCNT_Get ([in] VT_I4 NodeID, [in] VT_I4 ModuleIdx, [in] VT_I4 ModuleInCh, [out] VT_PI4 GlobalCNT)

DESCRIPTION

원격 노드 및 카운터 모듈, 모듈 내의 카운터 채널 번호를 통하여 전체 원격 노드의 통합 카운터 채널 번호(Global Counter Channel Number)를 반환합니다.

PARAMETER

▶ *NodeID* : 원격 노드 ID. 통합 카운터 채널 번호를 얻어올 카운터 모듈이 속한 원격 노드 ID 를 설정합니다.

▶ *ModuleIdx* : 카운터 모듈 ID. 통합 카운터 채널 번호를 얻어올 카운터 모듈 ID 를 설정합니다.

▶ *ModuleInCh* : 카운터 모듈 내의 채널 번호. 통합 카운터 채널 번호를 얻어올 카운터 모듈 내의 채널 번호를 설정합니다.

▶ *GlobalCNT* : 이 매개변수를 통하여 통합 카운터 채널 번호를 반환합니다.

RETURN VALUE

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

SEE ALSO

ceGnGlobalAxis_Get, ceGnGlobalDIO_Get, ceGnGlobalAI_Get, ceGnGlobalAO_Get, ceGnGlobalMDIO_Get, ceGnGlobalSER_Get

EXAMPLE

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

NAME

ceGnGlobalSER_Get

- 원격 노드 및 시리얼 통신 모듈, 모듈 내의 채널 번호를 통하여 통합(Global) 채널 번호를 반환

INFORMATION	
1	General Function
!	VC++ (6, 7, 8)/VB
	BCB/Delphi
:	Level 1
K	위험 요소 없음

SYNOPSIS

r VT_I4 ceGnGlobalSER_Get ([in] VT_I4 NodeID, [in] VT_I4 ModuleIdx,
[in] VT_I4 ModuleInCh, [out] VT_PI4 GlobalSER)

DESCRIPTION

원격 노드 및 시리얼 통신 모듈, 모듈 내의 시리얼 통신 채널 번호를 통하여 전체 원격 노드의 통합 시리얼 통신 채널 번호(Global Serial Communication Channel Number)를 반환합니다.

PARAMETER

▶ ***NodeID*** : 원격 노드 ID. 통합 시리얼 통신 채널 번호를 얻어올 시리얼 통신 모듈이 속한 원격 노드 ID 를 설정합니다.

▶ ***ModuleIdx*** : 시리얼 통신 모듈 ID. 통합 시리얼 통신 채널 번호를 얻어올 시리얼 통신 모듈 ID 를 설정합니다.

▶ ***ModuleInCh*** : 시리얼 통신 모듈 내의 채널 번호. 통합 시리얼 통신 채널 번호를 얻어올 시리얼 통신 모듈 내의 채널 번호를 설정합니다.

▶ ***GlobalSER*** : 이 매개변수를 통하여 통합 시리얼 통신 채널 번호를 반환합니다.

RETURN VALUE

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

SEE ALSO

ceGnGlobalAxis_Get, ceGnGlobalDIO_Get, ceGnGlobalAI_Get, ceGnGlobalAO_Get, ceGnGlobalMDIO_Get, ceGnGlobalSER_Get

EXAMPLE

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

NAME

ceGnEmergency_Set / ceGnEmergency_Get

- 소프트웨어 Emergency 설정 및 설정 상태 반환

INFORMATION	
1	General Function
!	VC++ (6, 7, 8)/VB
	BCB/Delphi
:	Level 1
J	위험 요소 없음

SYNOPSIS

r VT_I4 ceGnEmergency_Set ([in] VT_I4 NodeID, [in] VT_I4 State)

r VT_I4 ceGnEmergency_Get ([in] VT_I4 NodeID, [out] VT_PI4 State)

DESCRIPTION

ceGnEmergency_Set 함수는 해당 원격 노드에 소프트웨어 Emergency 를 설정하거나 해제하며, ceGnEmergency_Get 함수는 설정되어있는 소프트웨어 Emergency 상태에 대해 반환합니다.

소프트웨어적인 Emergency 는 EMG 센서의 설치가 용이하지 않을 때 안전을 위하여 소프트웨어 Emergency 상태를 설정하는 것입니다. 소프트웨어 Emergency 상황을 설정하면 모션이 감속 없이 비상 정지 합니다.

PARAMETER

▶ ***NodeID*** : 원격 노드 ID. 소프트웨어 Emergency 상태를 설정할 원격 노드 ID 를 설정합니다.

▶ ***State*** : 이 매개변수를 통해 소프트웨어 Emergency 설정 혹은 설정된 상태를 반환합니다.

Value	Meaning
0 (CE_FALSE)	소프트웨어 Emergency 비활성.
1 (CE_TRUE)	소프트웨어 Emergency 활성.

RETURN VALUE

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

EXAMPLE

C/C++

```
#include “ceSDK.h”
#include “ceSDKDef.h”

void OnSetSwEmergency ()
{
    long nNodeID = 1;           // 해당 원격 노드 ID.
    long nSwEmgStatus;          // 해당 원격 노드 모션 모듈의 소프트웨어 Emergency 상태 정보.

    /* 해당 노드의 소프트웨어 Emergency 설정 상태를 확인한 후, 설정된 상태에 따라
    소프트웨어 Emergency 상태 설정 또는 해제 합니다. */

    if ( ceGnEmergency_Get ( nNodeID,  &nSwEmgStatus ) == ceERR_NONE )
    {
            switch ( nSwEmgStatus )
            {
            case CE_FALSE :         // 소프트웨어 Emergency 비활성 상태
                    ceGnEmergency_Set ( nNodeID, CE_TRUE );
                    break;

            case CE_TRUE :          // 소프트웨어 Emergency 활성 상태
                    ceGnEmergency_Set ( nNodeID, CE_FALSE );
                    break;
            }
    }
}
```

Visual Basic

```
Private Sub OnSetSwEmergency ()

    Dim nNodeID As Long             ‘ 해당 원격 노드 ID.
    Dim nSwEmgStatus As Long        ‘ 해당 원격 노드 모션 모듈의 소프트웨어 Emergency 상태 정보.

    nNodeID = 1

    ‘ 해당 노드의 소프트웨어 Emergency 설정 상태를 확인한 후, 설정된 상태에 따라
    ‘ 소프트웨어 Emergency 상태 설정 또는 해제 합니다.

    If ceGnEmergency_Get ( nNodeID, nSwEmgStatus ) = ceERR_NONE Then

            Select Case nSwEmgStatus
                    Case CE_FALSE           ‘ 소프트웨어 Emergency 비활성 상태
                            ceGnEmergency_Set ( nNodeID, CE_TRUE )
                    Case CE_TRUE            ‘ 소프트웨어 Emergency 활성 상태
                            ceGnEmergency_Set ( nNodeID, CE_FALSE )
            End Select
    End If
End Sub
```

Delphi

```
procedure OnSetSwEmergency ();
var
    nNodeID : LongInt;                  // 해당 원격 노드 ID.
    nSwEmgStatus : LongInt;             // 해당 원격 노드 모션 모듈의 소프트웨어 Emergency 상태 정보.

begin
    nNodeID := 1;

    { 해당 노드의 소프트웨어 Emergency 설정 상태를 확인한 후, 설정된 상태에 따라
    소프트웨어 Emergency 상태 설정 또는 해제 합니다. }

    if ceGnEmergency_Get ( nNodeID, nSwEmgStatus ) = ceERR_NONE then
    begin
            case nSwEmgStatus of
                    CE_FALSE :              // 소프트웨어 Emergency 비활성 상태
                            ceGnEmergency_Set ( nNodeID, CE_TRUE );
                    CE_TRUE :               ‘ 소프트웨어 Emergency 활성 상태
                            ceGnEmergency_Set ( nNodeID, CE_FALSE );
            end;
    end;

end;
```

Chapter

5

General Motion Functions

ceSDK 의 모션 함수 중 모션 환경 설정이나 이송 함수 이외에도 일반 모션 함수들을 지원합니다.

본 단원에서는 일반 모션 함수들을 소개합니다. 일반 모션 함수들은 그 기능에 있어 모든 내용이 반드시 습득 될 필요는 없지만, 모션 제어를 위해 꼭 필요한 내용으로 구성되어 있습니다. 기본적인 서보 ON 출력 신호제어 및 서보 드라이버 알람 리셋 기능이 제공됩니다.

5 General Motion Functions

5.1 함수 요약

"General Motion Functions" 에 관련된 함수들은 다음과 같습니다.

Summary of Functions
r VT_I4 cemGnServoOn_Set ([in] VT_I4 Channel, [in] VT_I4 Enable) 해당 모션 축의 서보 드라이버에 대해 SERVO-ON 신호 출력을 제어합니다.
r VT_I4 cemGnServoOn_Get ([in] VT_I4 Channel, [out] VT_PI4 Enable) 해당 모션 축의 서보 드라이버에 대해 SERVO-ON 신호 출력 상태를 반환합니다.
r VT_I4 cemGnAlarmReset ([in] VT_I4 Axis, [in] VT_I4 IsReset) 해당 모션 축의 알람 리셋(Alarm Reset) 신호 출력을 제어합니다.

5.2 함수 설명

NAME

cemGnServoOn_Set / cemGnServoOn_Get

- Servo-On 신호 출력 제어 및 출력 상태 반환

INFORMATION
1 General Motion Functions
! VC++ (6, 7, 8)/VB/ BCB/Delphi
: Level 2
K 다소 위험
서보 동작 시에 반드시 주의 환경을 점검하고, 안전 사항에 유의해야 합니다.

SYNOPSIS

r VT_I4 cemGnServoOn_Set ([in] VT_I4 Channel, [in] VT_I4 Enable)

r VT_I4 cemGnServoOn_Get ([in] VT_I4 Channel, [out] VT_PI4 Enable)

DESCRIPTION

cemGnServoOn_Set 함수는 지정한 축의 SVON(Servo-On) 신호 출력을 제어합니다.

서보 드라이버 를 사용하실 때는 외부에서 스위치를 이용하여 서보 드라이버의 ON/OFF 를 제어할 수 있도록 하는데, 이를 SVON 신호라 합니다. 이 함수는 SVON 신호의 ON/OFF 를 제어하는 함수입니다.

cemGnServoOn_Get 함수는 현재 SVON 신호의 출력 상태를 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.

▶ *Enable* : 이 매개변수를 통하여 SVON 신호의 출력 상태를 설정 혹은 설정 상태를 반환합니다.

Value	Meaning
0 (CE_FALSE)	SERVO-OFF.
1 (CE_TRUE)	SERVO-ON.

RETURN VALUE

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

REFERENCE

□ 서보 드라이버를 On 시켜야 하는 경우 원격 노드 탐색 함수를 통해 원격 노드 정보를 얻어온 이후에 cemGnServoOn_Set(Axis#, CE_TRUE)를 호출해 주셔야 서보 모터가 정상 동작 합니다.

□ 서보 드라이버 Servo-ON 신호의 출력 로직은 다음 명령을 통해 설정하실 수 있습니다.
cemCfgMioProperty_Set(Axis#, cemMPID_SVON_LOGIC, cemLOGIC_A) 또는
cemCfgMioProperty_Set(Axis#, cemMPID_SVON_LOGIC, cemLOGIC_B).

EXAMPLE

C/C++

```
#include “ceSDK.h”
#include “ceSDKDef.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 ( cemStReadMioStatuses ( nAxisNo,  &nMioStates ) == ceERR_NONE )
    {
            if ( (nMioStates >> cemIOST_SVON) & 0x1 == CE_FALSE )
            {
                    // Servo OFF 상태이므로 Servo ON 상태로 설정합니다.
                    cemGnServoOn_Set ( nAxisNo, CE_TRUE );
            }
            else
            {
                    // Servo ON 상태이므로 Servo OFF 상태로 설정합니다.
                    cemGnServoOn_Set ( nAxisNo, CE_FALSE );
            }
    }
}
```

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 cemStReadMioStatuses ( nAxisNo,  nMioStates ) = ceERR_NONE Then
```

```
        Call ceGnBitShift ( nMioStates, cemIOST_SVON, nResult )

        If  nResult == CE_FALSE Then
                ' Servo OFF 상태이므로 Servo ON 상태로 설정합니다.
                Call cemGnServoOn_Set ( nAxisNo, CE_TRUE )

        Else
                ' Servo ON 상태이므로 Servo OFF 상태로 설정합니다.
                cemGnServoOn_Set ( nAxisNo, CE_FALSE )
        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 cemStReadMioStatuses ( nAxisNo,  @nMioStates ) = ceERR_NONE then
    begin
            if  ( ( nMioStates shr cemIOST_SVON ) and $1 ) = CE_FALSE then
            begin
                    // Servo OFF 상태이므로 Servo ON 상태로 설정합니다.
                    cemGnServoOn_Set ( nAxisNo, CE_TRUE );
            end
            else
                    // Servo ON 상태이므로 Servo OFF 상태로 설정합니다.
                    cemGnServoOn_Set ( nAxisNo, CE_FALSE );
            end;
    end;

end;
```

NAME

cemGnAlarmReset

- 알람 리셋(Alarm Reset) 신호 출력 제어

INFORMATION	
1	General Motion Functions
!	VC++ (6, 7, 8)/VB
	BCB/Delphi
:	Level 1
J	위험 요소 없음

SYNOPSIS

r VT_I4 cemGnAlarmReset ([in] VT_I4 Axis, [in] VT_I4 IsReset)

DESCRIPTION

해당 모션 축의 ARST(알람 리셋) 신호 출력을 제어하는 함수입니다. 서보 드라이버에서 발생한 알람 상황을 초기화할 수 있도록 서보 드라이버 상에서 요구하는 입력 신호가 존재하는데, 이 신호 입력에 대응하기 위해, ARST 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.

PARAMETER

▶ ***Axis*** : 축 번호. 통합 축으로 관리되는 축 번호를 의미하며, 상수 값으로 0 (Zero Based) 이상, 최대 통합 축 개수 - 1 이하의 값을 축 번호로 설정할 수 있습니다.

▶ ***IsReset*** : 알람 리셋 신호 출력 여부를 결정합니다.

Value	Meaning
0 (CE_FALSE)	알람 리셋(Alarm Reset) 신호를 출력 하지 않음.
1 (CE_TRUE)	알람 리셋(Alarm Reset) 신호 출력

RETURN VALUE

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

REFERENCE

□ 서보 드라이버에 발생한 모든 알람을 ARST 신호 출력을 통해 해제할 수 없습니다. 서보 드라이버에서 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 조건등이 동작이 불가한 경우에 발생하는 심각한 상황은 알람 리셋 신호로 알람 상황이 해제되지 않습니다.

□ ARST 신호의 출력은 모션 모듈에서 출력하는 순수한 단방향 디지털 출력 신호이며, 서보 드라이버의 알람 상태를 제어하는 신호가 아닙니다. 알람의 종류는 사용하시는 서보 드라이버의 매뉴얼을 참고하시기 바랍니다.

□ 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 드라이버 사양마다 다르며, 서보 드라이버 알람 출력 시간이 너무 짧거나 길면, 알람 리셋이 되지 않거나 서보 드라이버에 무리를 줄 수도 있습니다.

EXAMPLE

C/C++

```
#include “ceSDK.h”
#include “ceSDKDef.h”

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

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

    cemGnAlarmReset ( nAxisNo, CE_TRUE );   // ARST ON

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

    cemGnAlarmReset ( nAxisNo, CE_FALSE ); // 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 cemGnAlarmReset ( nAxisNo, CE_TRUE )          ‘ ARST ON

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

    Call cemGnAlarmReset ( nAxisNo, CE_FALSE )         ‘ 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.
    cemGnAlarmReset ( nAxisNo, CE_TRUE );   // ARST ON
```

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

    cemGnAlarmReset ( nAxisNo, CE_FALSE ); // ARST OFF

end;
```

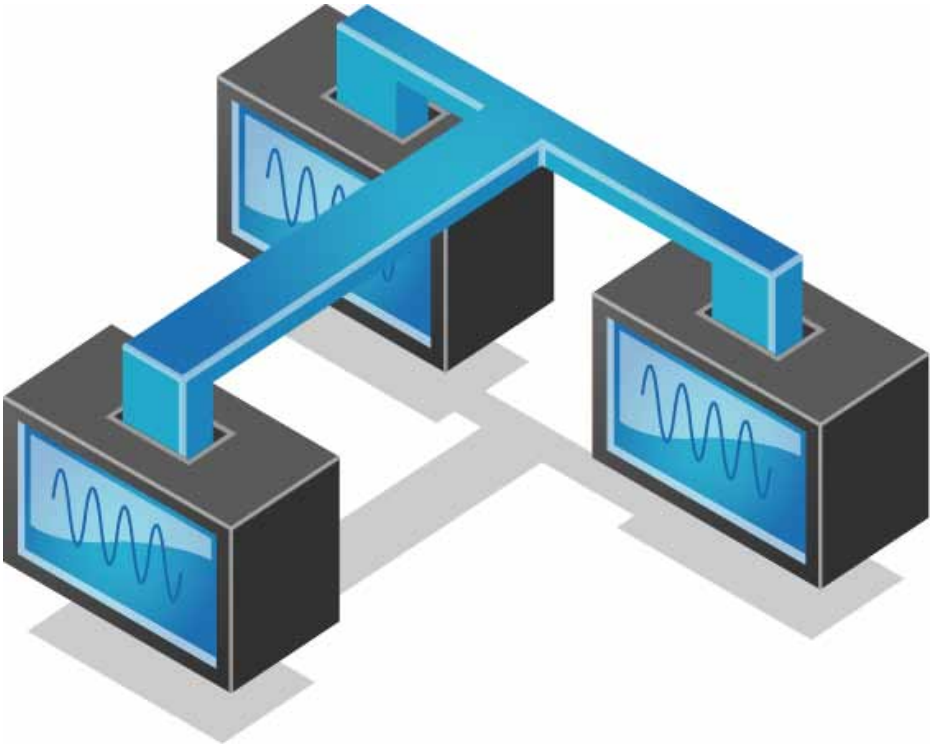
Chapter

6

Environment Configuration Functions

모션 환경설정의 다양한 함수는 결과적으로 ㈜ 커미조아 모션 컨트롤러의 환경 설정이 얼마나 자세하고 명확한지를 판단하게 합니다. 기본적인 모터 드라이버를 위한 입력 펄스 신호 설정부터, 모션 주변신호까지 다양하게 지원하는 환경 설정 함수는 다양한 주변신호에 대한 강력한 노이즈의 필터 기능까지 지원하고 있습니다.

이 단원에서는 모션컨트롤러의 환경을 설정하는 함수들을 소개합니다. 이 단원에서 설명되는 모든 환경은 별도의 모션 환경 설정 프로그램을 통하여 설정하여, cEIP Flash Memory 에 영구적으로 기록할 수 있습니다. 따라서 런타임시에 환경을 변경하고자하는 경우가 아니면, 본 단원에서 설명되는 함수를 사용하실 필요가 없습니다. 그러나 필요에 따라 런타임시에 동적으로 환경을 변경해야할 필요가 있을 수 있으며 이러한 때에는 본 단원에서 소개하는 함수들을 이용하여 동적으로 환경을 변경할 수 있습니다.



6 모션 환경 설정 함수

6.1 함수 요약

“모션 환경설정” 에 관련된 함수들은 다음과 같습니다.

Summary of Functions
r VT_I4 cemCfgMioProperty_Set ([in] VT_I4 Axis, [in] VT_I4 PropId, [in] VT_I4 PropVal) 모션 입·출력(MIO) 신호의 환경설정을 구성합니다.
r VT_I4 cemCfgMioProperty_Get ([in] VT_I4 Axis, [in] VT_I4 PropId, [out] VT_PI4 PropVal) 모션 입·출력 신호의 환경설정 값을 반환합니다.
r VT_I4 cemCfgFilter_Set ([in] VT_I4 Axis, [in] VT_I4 IsEnable) 각종 입·출력(Input/Output) 신호에 대해 노이즈 대응 필터(Filter) 기능을 설정합니다.
r VT_I4 cemCfgFilter_Get ([in] VT_I4 Axis, [out] VT_PI4 IsEnabled) 각종 입·출력 신호에 대해 노이즈 대응 필터 기능의 설정 상태를 반환합니다.
r VT_I4 cemCfgFilterAB_Set ([in] VT_I4 Channel, [in] VT_I4 Target, [in] VT_I4 IsEnable) EA/EB 또는 PA/PB 신호 입력 회로에 노이즈 대응 필터 기능을 설정합니다.
r VT_I4 cemCfgFilterAB_Get ([in] VT_I4 Channel, [in] VT_I4 Target, [out] VT_PI4 IsEnabled) EA/EB 또는 PA/PB 신호 입력 회로에 노이즈 대응 필터 기능의 설정 상태를 반환합니다.
r VT_I4 cemCfgInMode_Set ([in] VT_I4 Axis, [in] VT_I4 InputMode, [in] VT_I4 IsReverse) 인코더 펄스(Feedback Pulse) 신호의 입력 모드를 설정합니다.
r VT_I4 cemCfgInMode_Get ([in] VT_I4 Axis, [out] VT_PI4 InputMode, [out] VT_PI4 IsReverse) 인코더 펄스 신호의 입력 모드 설정 상태를 반환합니다.
r VT_I4 cemCfgOutMode_Set ([in] VT_I4 Axis, [in] VT_I4 OutputMode) 지령 펄스(Command Pulse) 신호의 출력 모드를 설정합니다.
r VT_I4 cemCfgOutMode_Get ([in] VT_I4 Axis, [out] VT_PI4 OutputMode) 지령 펄스 신호의 출력 모드 설정 상태를 반환합니다.
r VT_I4 cemCfgCtrlMode_Set ([in] VT_I4 Axis, [in] VT_I4 CtrlMode) 이송 목표 좌표의 기준을 커맨드 위치로 할지 피드백 위치로 할지를 설정하는 함수입니다.
r VT_I4 cemCfgCtrlMode_Get ([in] VT_I4 Axis, [out] VT_PI4 CtrlMode) 이송 목표 좌표의 기준 설정 상태를 반환합니다.
r VT_I4 cemCfgInOutRatio_Set ([in] VT_I4 Axis, [in] VT_R8 Ratio) 입력 펄스와 출력 펄스의 분해능 비율(Resolution Ratio) 을 설정합니 다.
r VT_I4 cemCfgInOutRatio_Get ([in] VT_I4 Axis, [out] VT_PR8 Ratio) 입력 펄스와 출력 펄스의 분해능 비율 설정 상태를 반환합니다.
r VT_I4 cemCfgUnitDist_Set ([in] VT_I4 Axis, [in] VT_R8 UnitDist) 지정된 모션 축에 대한 논리적 거리 단위를 설정합니다.
r VT_I4 cemCfgUnitDist_Get ([in] VT_I4 Axis, [out] VT_PR8 UnitDist) 지정된 모션 축에 대한 논리적 거리 단위의 설정 상태를 반환합니다.
r VT_I4 cemCfgUnitSpeed_Set ([in] VT_I4 Axis, [in] VT_R8 UnitSpeed) 지정된 모션 축에 대한 논리적 속도 단위를 설정합니다.

r VT_I4 cemCfgUnitSpeed_Get ([in] VT_I4 Axis, [out] VT_PR8 UnitSpeed) 지정된 모션 축에 대한 논리적 속도 단위의 설정 상태를 반환합니다.
r VT_I4 cemCfgSpeedRange_Set ([in] VT_I4 Axis, [in] VT_R8 MaxPPS) 지정된 모션 축에 대해 모션 속도를 제한하고, 제한 범위를 설정합니다.
r VT_I4 cemCfgSpeedRange_Get ([in] VT_I4 Axis, [out] VT_PR8 MinPPS, [out] VT_PR8 MaxPPS) 지정된 모션 축에 대해 모션 속도 제한 범위 설정 상태를 반환합니다.
r VT_I4 cemCfgSpeedPattern_Set ([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 비율을 통해 모션 이송의 실제 속도를 설정할 수 있습니다.
r VT_I4 cemCfgSpeedPattern_Get ([in] VT_I4 Axis, [out] VT_PI4 SpeedMode, [out] VT_PR8 WorkSpeed, [out] VT_PR8 Accel, [out] VT_PR8 Decel) 모션 이송의 전역 기준속도를 반환합니다. 반환된 이 속도의 비율을 통해 모션 이송의 실제 속도가 설정됩니다.
r VT_I4 cemCfgSoftLimit_Set ([in] VT_I4 Axis, [in] VT_I4 IsEnable, [in] VT_R8 LimitN, [in] VT_R8 LimitP) 모션의 소프트웨어적인 이송제한범위를 설정하여 이송 범위를 제한합니다.
r VT_I4 cemCfgSoftLimit_Get ([in] VT_I4 Axis, [out] VT_PI4 IsEnable, [out] VT_PR8 LimitN, [out] VT_PR8 LimitP) 모션의 소프트웨어적인 이송제한범위에 대한 해당 설정을 반환합니다.
r VT_I4 cemCfgRingCntr_Set ([in] VT_I4 Channel, [in] VT_I4 TargCntr, [in] VT_I4 IsEnable, [in] VT_R8 CntMax) 지정된 모션 축에 대해 링카운터(Rign-Counter) 기능을 설정합니다.
r VT_I4 cemCfgRingCntr_Get ([in] VT_I4 Channel, [in] VT_I4 TargCntr, [out] VT_PI4 IsEnable, [out] VT_PR8 CntMax) 지정된 모션 축에 대해 링카운터 기능에 대한 해당 설정을 반환합니다.
r VT_I4 cemCfgSeqMode_Set ([in] VT_I4 SeqMode) 이전의 이송 작업이 완료되지 않은 축에 새로운 이송 명령이 하달되었을 때의 처리 정책을 설정합니다.
r VT_I4 cemCfgSeqMode_Get ([out] VT_PI4 SeqMode) 이전의 이송 작업이 완료되지 않은 축에 새로운 이송 명령이 하달되었을 때의 처리 정책에 대한 설정 상태를 반환합니다.

6.2 함수 설명

NAME

cemCfgMioProperty_Set /

cemCfgMioProperty_Get

모션 입·출력(MIO) 신호 환경 설정 및 반환

INFORMATION	
1	Environment Config.
!	VC++ (6, 7, 8)/VB
	BCB/Delphi
:	Level 2
K	다소 주의

MIO 신호의 환경 설정을 위한 함수 입니다. 반드시 숙지해 주시기 바랍니다.

SYNOPSIS

r VT_I4 cemCfgMioProperty_Set ([in] VT_I4 Axis, [in] VT_I4 PropId, [in] VT_I4 PropVal)

r VT_I4 cemCfgMioProperty_Get ([in] VT_I4 Axis, [in] VT_I4 PropId, [out] VT_PI4 PropVal)

DESCRIPTION

cemCfgMioProperty_Set 함수는 각종 모션 입·출력(MIO) 신호에 대한 환경을 설정합니다. 이 함수 는 다양한 MIO 신호의 환경을 설정하는데 공통적으로 사용하는 함수입니다. PropId 에 따라 어떠한 환경을 설정할 지를 결정하게 됩니다.

cemCfgMioProperty_Get 함수는 각종 모션 입·출력 신호에 대하여 현재 설정되어 있는 환경 설정 값을 반환합니다. 어떠한 MIO 신호의 환경 설정 값을 반환할 지는 PropId 에 따라 결정됩니다.

PARAMETER

▶ ***Axis*** : 축 번호. 통합 축으로 관리되는 축 번호를 의미하며, 상수 값으로 0 (Zero Based) 이상, (최대 통합 축 개수 – 1) 이하의 값을 축 번호로 설정할 수 있습니다.

▶ ***PropId*** : 어떠한 MIO 신호 환경에 대하여 설정할 것인지를 지정하는 매개변수입니다. 이 값에 대해서는 아래 표를 참조하시기 바랍니다.

▶ ***PropVal*** : PropId 로 지정된 MIO 신호의 환경에 대해 설정 혹은 반환합니다.

PropId	Meaning & PropVal
0 (cemMPID_ALM_LOGIC)	ALM (Alarm) 신호의 입력 로직입니다. ▪ 0 (cemLOGIC_A) : A 접점 방식 => 평상시 Open, 감지되면 Close 되는 스위치 방식. ▪ 1 (cemLOGIC_B) : B 접점 방식 => 평상시 Close, 감지되면 Open 되는 스위치 방식.

PropId	Meaning & PropVal
1 (cemMPID_ALM_MODE)	ALM 입력 신호가 ON 되어 해당 축의 모션 작업이 정지할 때 정지되는 방식을 설정합니다. ▪ 0 (CE_FALSE) : 즉시 정지. ▪ 1 (CE_TRUE) : 감속 후 정지.
2 (cemMPID_CMP_LOGIC)	CMP (위치비교출력) 신호의 출력 방식을 설정합니다. ▪ 0 (Active low) : 평상시 HIGH 상태를 유지. 트리거 시점에서 LOW 로 떨어졌다가 다시 HIGH 로 올라갑니다. ▪ 1(Active high) : 평상시 LOW 상태를 유지. 트리거 시점에서 HIGH 로 올라갔다가 다시 LOW 로 떨어집니다.
3 (cemMPID_DR_LOGIC)	± DR(External Switch) 신호의 입력 로직을 설정합니다. ▪ 0 (cemLOGIC_A) : A 접점 방식. ▪ 1 (cemLOGIC_B) : B 접점 방식.
4 (cemMPID_EL_LOGIC)	± EL(End of Limit) 신호의 입력 로직을 설정합니다. ▪ 0 (cemLOGIC_A) : A 접점 방식. ▪ 1 (cemLOGIC_B) : B 접점 방식.
5 (cemMPID_EL_MODE)	± EL 신호가 ON 되어 정지할 때 정지 방식을 설정합니다. ▪ 0 (CE_FALSE) : 즉시 정지. ▪ 1 (CE_TRUE) : 감속 후 정지.
6 (cemMPID_ERC_LOGIC)	ERC (편차카운터 클리어) 신호 출력 출력 로직을 설정합니다. ▪ 0 (cemLOGIC_A) : A 접점 방식. ▪ 1 (cemLOGIC_B) : B 접점 방식.
7 (cemMPID_ERC_OUT)	원점복귀 완료 시에 ERC 신호 출력 여부를 설정합니다. ▪ 0 (CE_FALSE) : 원점복귀 완료 시에 ERC 신호 출력 없음. ▪ 1 (CE_TRUE) : 원점복귀 완료 시에 ERC 신호 출력.
8 (cemMPID_EZ_LOGIC)	EZ (엔코더 Z 상) 신호의 입력 로직을 설정합니다. ▪ 0 (cemLOGIC_A) : A 접점 방식. ▪ 1 (cemLOGIC_B) : B 접점 방식.
9 (cemMPID_INP_EN)	INP (In-Position) 신호 입력에 따른 동작 모드를 설정합니다. ▪ 0 (CE_FALSE) : INP 입력 신호 무시. ▪ 1 (CE_TRUE) : INP 입력 신호 활성 => Command 출력이 완료되더라도 INP 신호가 ON 되기 전까지는 작업이 완료되지 않은 것으 로 간주.
10 (cemMPID_INP_LOGIC)	INP 신호의 입력 로직을 설정합니다. ▪ 0 (cemLOGIC_A) : A 접점 방식. ▪ 1 (cemLOGIC_B) : B 접점 방식.
11 (cemMPID_LTC_LOGIC)	LTC (Latch) 신호의 입력 로직을 설정합니다. ▪ 0 (cemLOGIC_A) : A 접점 방식. ▪ 1 (cemLOGIC_B) : B 접점 방식.
12 (cemMPID_LTC_LTC2SRC)	두 번째 Latch Counter 의 대상 카운터를 설정합니다. ▪ 0 : Deviation counter value. ▪ 1 : Preset speed of command pulse.
13 (cemMPID_ORG_LOGIC)	ORG (원점 센서) 신호의 입력 로직을 설정합니다. ▪ 0 (cemLOGIC_A) : A 접점 방식. ▪ 1 (cemLOGIC_B) : B 접점 방식.

PropId	Meaning & PropVal
14 (cemMPID_SD_EN)	SD(Start of Deceleration) 신호의 입력 상태를 설정합니다. ▪ 0 (CE_FALSE) : SD 입력을 비활성화 합니다. ▪ 1 (CE_TRUE) : SD 입력을 활성화 합니다. 활성화 되었을 때 SD 신호에 따른 동작 방식은 cemMPID_SD_LATCH 와 cemMPID_SD_MODE 설정 값에 의해 결정됩니다.
15 (cemMPID_SD_LOGIC)	SD(Start of Deceleration) 신호의 입력 로직을 설정합니다. ▪ 0 (cemLOGIC_A) : A 접점 방식. ▪ 1 (cemLOGIC_B) : B 접점 방식.
16 (cemMPID_SD_LATCH)	SD 신호를 래치(Latch)할 것인지에 대한 속성값입니다. • 0 (CE_FALSE) : SD 가 ON 되어 감속중이거나 초기 속도로 이송 중일 때 SD 신호가 다시 OFF 상태로 변경되면 작업 속도까지 다시 가속됩니다. • 1 (CE_TRUE) : SD 가 ON 상태에서 OFF 상태로 바뀌어도 작업 속도로 가속하지 않습니다.
17 (cemMPID_SD_MODE)	SD 신호에 따른 동작 모드를 설정합니다. • 0 (CE_FALSE) : SD 신호가 ON 되면 초기 속도까지 감속 합니다.(정지하지 않음) • 1 (CE_TRUE) : SD 신호가 ON 되면 감속 후 정지합니다.
18 (cemMPID_STA_MODE)	STA (동기 Start) 신호 모드를 설정합니다. • 0 (CE_FALSE) : STA 입력 신호는 무시되며, 이송 명령이 내려지면 바로 이송을 시작합니다. • 1 (CE_TRUE) : 이송 명령이 내려져도 바로 이송을 시작하지 않고, STA 신호가 ON 이 되면 이송을 시작합니다.
19 (cemMPID_STA_TRG)	STA 신호가 ON 되는 형태를 설정합니다. • 0 (CE_FALSE) : Level (LOW) => STA 신호가 LOW LEVEL 일때 ON. • 1 (CE_TRUE): Falling Edge => STA 신호가 HIGH 상태에서 LOW 상태로 천이될 때 ON.
20 (cemMPID_STP_MODE)	STP (동기 신호 모드를 설정합니다. • 0 : Ignore STP => STP 입력 신호 무시. • 1 : Immediate stop => STP 입력이 ON 되면 즉시 정지. • 2 : Stop after decel => STP 입력이 ON 되면 감속 후 정지.
21 (cemMPID_CLR_CNTR)	CLR(카운터 클리어) 신호가 입력되었을 때 클리어 되도록 할 카운터를 선택합니다. 이 값은 4 비트의 값으로 설정하며 각 비트는 다음과 같습니다. • 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 (cemMPID_CLR_SIGTYPE)	CLR 신호의 신호 형태를 설정합니다. • 0 : Falling edge => 스위치가 Open 상태에서 Close 되는 순간에 카운터가 클리어 됩니다. • 1 : Rising edge => 스위치가 Close 상태에서 Open 되는 순간에 카운터가 클리어 됩니다. • 2 : Low level => 스위치가 Close 되어 있는 동안에는 계속 카운터가 클리어 됩니다. • 3 : High level => 스위치가 Open 되어 있는 동안에는 계속 카운터가 클리어 됩니다.

PropId	Meaning & PropVal
23 (cemMPID_CMP_PWIDTH)	CMP 출력은 One-shot pulse 로 출력되며, 출력되는 펄스의 폭을 조절할 수 있습니다. • 0 : 트리거 시점의 Command 펄스의 펄스 폭과 동일한 펄스 폭으로 출력합니다. • 양수 : 이 값에 1.5us 가 곱해진 값이 펄스 폭이 됩니다. 즉, 이 값을 1 로 하면 1.5us, 2 로 하면 3us...와 같이 됩니다.
24 (cemMPID_ERC_ONTIME)	ERC 출력 펄스의 펄스 폭을 설정합니니다. 설정 및 반환되는 PropVal 은 다음과 같습니다. 0 => 12us, 1 => 102us, 2 => 409us, 3 => 1.6ms, 4 => 13ms, 5 => 52ms, 6 => 104ms, 7=> Logic Level Output
25 (cemMPID_SVON_LOGIC)	SVON(Servo-On) 신호의 출력 로직을 설정합니다. ▪ 0 (cemLOGIC_A) : A 접점 방식. ▪ 1 (cemLOGIC_B) : B 접점 방식.
26 (cemMPID_ERC_OUT_EL)	± EL 신호가 ON 되어 정지할 때 ERC 출력 발생 여부를 설정합니다. 설정 및 반환되는 PropVal 은 다음과 같습니다. ▪ 0 (CE_FALSE) : ERC 출력을 발생하지 않습니다. ▪ 1 (CE_TRUE) : ERC 출력을 발생합니다.
27 (cemMPID_CNT_D_SRC)	Deviation(편차) 카운터의 입력 소스(Input Source) 를 선택합니다. 이 값은 4 비트의 값으로 설정하며 각 비트는 다음과 같습니다. ▪ Bit 0 : 출력 펄스와 EA/EB 단자를 통한 입력 카운트. ▪ Bit 1 : 출력 펄스와 PA/PB 단자를 통한 입력 카운트. ▪ Bit 2 : EA/EB 단자와 PA/PB 단자를 통한 입력 카운트.
28 (cemMPID_CNT_G_SRC)	General(범용) 카운터의 입력 소스(Input Source) 를 선택합니다. 이 값은 4 비트의 값으로 설정하며 각 비트는 다음과 같습니다. ▪ Bit 0 : 출력 펄스. ▪ Bit 1 : EA/EB 단자를 통한 입력 카운트. ▪ Bit 2 : PA/PB 단자를 통한 입력 카운트. ▪ Bit 3 : CLK 카운트를 2 로 나눈 값을 입력 카운트로 설정.
29 (cemMPID_LTC_TRGMODE)	LATCH 카운터의 트리거 모드를 설정합니다. ▪ 0 (cemLTM_LTC) : LTC 입력 신호에 의해서 포지션 래치가 수행됩니다. ▪ 1 (cemLTM_ORG) : ORG 입력 신호에 의해서 포지션 래치가 수행됩니다.
30 (cemMPID_SLIM_EN)	Software Limit 기능 사용 여부를 설정합니다. ▪ 0 (CE_FALSE) : Software Limit 기능을 사용하지 않습니다. ▪ 1 (CE_TRUE) : Software Limit 기능을 사용합니다.
31 (cemMPID_OUT_MODE)	Command(지령) 펄스 출력 모드를 선택합니다. ▪ 0 (cemOMODE_PDIR0) : Pulse & Dir 0. ▪ 1 (cemOMODE_PDIR1) : Pulse & Dir 1. ▪ 2 (cemOMODE_PDIR2) : Pulse & Dir 2. ▪ 3 (cemOMODE_PDIR3) : Pulse & Dir 3. ▪ 4 (cemOMODE_CWCCW0) : CW & CCW 0. ▪ 5 (cemOMODE_CWCCW1) : CW & CCW 1.

PropId	Meaning & PropVal
32 (cemMPID_IN_MODE)	Feedback(인코더) 펄스 입력 모드를 선택합니다. ▪ 0 (cemIMODE_AB1X) : 1X A/B. ▪ 1 (cemIMODE_AB2X) : 2X A/B. ▪ 2 (cemIMODE_AB4X) : 4X A/B. ▪ 3 (cemIMODE_CWCCW) : CW/CCW. ▪ 4 (cemIMODE_STEP) : Step Mode.
33 (cemMPID_IN_INV)	Feedback Count 값의 UP/DOWN 방향을 반대로 할 것인지를 선택합니다. ▪ 0 (CE_FALSE) : Feedback 카운터 UP/DOWN 방향 유지. ▪ 1 (CE_TRUE) : Feedback 카운터 UP/DOWN 방향 반대.
34 (cemMPID_CEMG_EN)	EMG(Emergency) 신호 입력 활성화 여부를 설정합니다. ▪ 0 (CE_FALSE) : EMG 비활성. ▪ 1 (CE_TRUE) : EMG 활성.

RETURN VALUE

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

EXAMPLE

C/C++

```
#include “ceSDK.h”
#include “ceSDKDef.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 ( cemCfgMioProperty_Get ( nAxisNo, cemMPID_ALM_LOGIC, &nAlmLogic ) == ceERR_NONE )
    {
            if ( nAlmLogic != cemLOGIC_B )
            {
                    // ALM 로직을 ‘B 접점 방식’으로 설정
                    cemCfgMioProperty_Set ( nAxisNo, cemMPID_ALM_LOGIC, cemLOGIC_B );
            }
    }

    // ALM 신호 ON 시 정지 모드 설정
    if ( cemCfgMioProperty_Get ( nAxisNo, cemMPID_ALM_MODE, &nAlmMode ) == ceERR_NONE )
    {
            if ( nAlmMode != CE_FALSE )
            {
                    // ALM 신호 ON 시 정지 모드를 ‘즉시 정지’ 모드로 설정
                    cemCfgMioProperty_Set ( nAxisNo, cemMPID_ALM_MODE, CE_FALSE );
            }
    }
}
```

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.

    long 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 cemCfgMioProperty_Get ( nAxisNo, cemMPID_ALM_LOGIC, nAlmLogic ) = ceERR_NONE Then
            If nAlmLogic <> cemLOGIC_B Then
```

```
                ' ALM 로직을 'B 접점 방식'으로 설정
                Call cemCfgMioProperty_Set ( nAxisNo, cemMPID_ALM_LOGIC, cemLOGIC_B )
        End If
    End If

    ' ALM 신호 ON 시 정지 모드 설정
    If cemCfgMioProperty_Get ( nAxisNo, cemMPID_ALM_MODE, nAlmMode ) = ceERR_NONE Then
        If nAlmMode <> CE_FALSE Then
            ' ALM 신호 ON 시 정지 모드를 '즉시 정지' 모드로 설정
            Call cemCfgMioProperty_Set ( nAxisNo, cemMPID_ALM_MODE, CE_FALSE )
        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 cemCfgMioProperty_Get ( nAxisNo, cemMPID_ALM_LOGIC, @nAlmLogic ) = ceERR_NONE then
    begin
        if nAlmLogic <> cemLOGIC_B then
        begin
            // ALM 로직을 'B 접점 방식'으로 설정
            cemCfgMioProperty_Set ( nAxisNo, cemMPID_ALM_LOGIC, cemLOGIC_B );
        end;
    end;

    // ALM 신호 ON 시 정지 모드 설정
    if cemCfgMioProperty_Get ( nAxisNo, cemMPID_ALM_MODE, @nAlmMode ) = ceERR_NONE then
    begin
        if nAlmMode <> CE_FALSE then
        begin
            // ALM 신호 ON 시 정지 모드를 '즉시 정지' 모드로 설정
            cemCfgMioProperty_Set ( nAxisNo, cemMPID_ALM_MODE, CE_FALSE );
        end;
    end;

end;
```

NAME

cemCfgFilter_Set / cemCfgFilter_Get

- MIO 신호에 대해 노이즈 필터(Noise Filter) 기능 설정 및 설정 상태 반환

INFORMATION	
1	Environment Config.
!	VC++ (6, 7, 8)/VB
	BCB/Delphi
:	Level 2
J	위험 요소 없음

SYNOPSIS

r VT_I4 cemCfgFilter_Set ([in] VT_I4 Axis, [in] VT_I4 IsEnable)

r VT_I4 cemCfgFilter_Get ([in] VT_I4 Axis, [out] VT_PI4 IsEnabled)

DESCRIPTION

cemCfgFilter_Set 함수는 지정한 축의 각종 MIO 신호에 대해 노이즈 필터 기능의 활성/비활성 상태를 설정합니다. 노이즈 필터 기능이 적용되면 펄스 폭이 너무 짧은 입력 신호는 무시되게 됩니다.

cemCfgFilter_Get 함수는 지정한 축의 각종 MIO 신호에 대해 노이즈 필터 기능 설정 상태를 반환합니다.

필터 로직이 적용되는 MIO 신호 및 펄스 폭은 아래의 REFERENCE 항목을 참조하십시오.

PARAMETER

▶ ***Axis*** : 축 번호. 통합 축으로 관리되는 축 번호를 의미하며, 상수 값으로 0 (Zero Based) 이상, (최대 통합 축 개수 - 1) 이하의 값을 축 번호로 설정할 수 있습니다.

▶ ***IsEnable*** : 해당 축의 노이즈 필터 기능의 활성/비활성 상태를 설정 혹은 반환합니다.

Value	Meaning
0 (CE_FALSE)	Noise Filter Disable. (노이즈 필터 기능 비활성)
1 (CE_TRUE)	Noise Filter Enable. (노이즈 필터 기능 활성)

RETURN VALUE

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

SEE ALSO

cemCfgFilterAB_Set, cemCfgFilterAB_Get

REFERENCE

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

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

EXAMPLE

C/C++

```
#include “ceSDK.h”
#include “ceSDKDef.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 ( cemCfgFilter_Get ( nAxisNo, &nFilterEnable ) == ceERR_NONE )
    {
            if ( nFilterEnable != CE_TRUE )
            {
                    // Noise Filter Enable
                    cemCfgFilter_Set ( nAxisNo, CE_TRUE );
            }
    }
}
```

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 cemCfgFilter_Get ( nAxisNo, nFilterEnable ) = ceERR_NONE Then

            If nFilterEnable <> CE_TRUE Then
                    ‘ Noise Filter Enable
                    Call cemCfgFilter_Set ( nAxisNo, CE_TRUE )
            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 cemCfgFilter_Get ( nAxisNo, @nFilterEnable ) = ceERR_NONE then
    begin
            if nFilterEnable <> CE_TRUE then
            begin
                    // Noise Filter Enable
                    cemCfgFilter_Set ( nAxisNo, CE_TRUE );
            end;
    end;

end;
```

NAME

cemCfgFilterAB_Set / cemCfgFilterAB_Get

- EA/EB, PA/PB 신호에 대해 노이즈 필터(Noise Filter) 기능 설정 및 설정 상태 반환

INFORMATION	
1	Environment Config.
!	VC++ (6, 7, 8)/VB BCB/Delphi
:	Level 2
J	위험 요소 없음

SYNOPSIS

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

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

DESCRIPTION

cemCfgFilterAB_Set 함수는 EA/EB(Encoder Feedback) 신호와 PA/PB(Manual Pulsar) 신호의 입력 회로에 노이즈 필터 기능의 활성/비활성 상태를 설정합니다.

필터를 적용하게 되면 펄스의 폭이 308ns 보다 작은 펄스는 노이즈로 간주되어서 필터됩니다. 이는 EA/EB 신호 또는 PA/PB 신호의 입력에 대해서 비교적 노이즈에 강한 처리를 할 수 있도록 합니다. 정리하자면, 노이즈 필터를 적용하게 되면 3.25MHz 이상의 주파수를 가지는 펄스는 노이즈로 간주되어 무시되므로 결과적으로 정상적인 처리가 될 수 있습니다.

필터의 적용 여부는 EA/EB 신호와 PA/PB 신호에 대하여 각각 서로 다르게 설정할 수 있으며, 이는 Target 매개변수를 통해서 함수의 적용 대상을 구분합니다.

cemCfgFilterAB_Get 함수는 EA/EB 신호와 PA/PB 신호의 입력 회로에 노이즈 필터 기능의 활성/비활성 상태를 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.

▶ *Target* : 함수의 적용 대상을 결정합니다. 이 매개변수에 적용 가능한 값은 다음과 같습니다.

Value	Meaning
0 (cemAB_ENC)	함수의 적용 대상이 EA/EB 신호임을 의미합니다.
1 (cemAB_PULSAR)	함수의 적용 대상이 PA/PB 신호임을 의미합니다.

▶ *IsEnable* : 필터 로직의 적용 여부를 설정 혹은 반환합니다.

Value	Meaning
0 (CE_FALSE) **[Default]**	Noise Filter disable.
1 (CE_TRUE)	Noise Filter enable.

RETURN VALUE

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

SEE ALSO

cemCfgFilter_Set, cemCfgFilter_Get

REFERENCE

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

EXAMPLE

C/C++

```
#include “ceSDK.h”
#include “ceSDKDef.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 ( cemCfgFilterAB_Get ( nAxisNo, cemAB_ENC, &nFilterEnable ) == ceERR_NONE )
    {
            if ( nFilterEnable != CE_TRUE )
            {
                    // EA/EB Noise Filter Enable
                    cemCfgFilterAB_Set ( nAxisNo, cemAB_ ENC, CE_TRUE );
            }
    }

    // PA/PB 신호 입력 회로에 노이즈 필터 기능 적용
    if ( cemCfgFilterAB_Get ( nAxisNo, cemAB_ PULSAR, &nFilterEnable ) == ceERR_NONE )
    {
            if ( nFilterEnable != CE_TRUE )
            {
                    // PA/PB Noise Filter Enable
                    cemCfgFilterAB_Set ( nAxisNo, cemAB_PULSAR, CE_TRUE );
            }
    }
}
```

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.

    long 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 cemCfgFilterAB_Get ( nAxisNo, cemAB_ENC, nFilterEnable ) = ceERR_NONE Then

            If nFilterEnable <> CE_TRUE Then
```

```
                ' EA/EB Noise Filter Enable
                Call cemCfgFilterAB_Set ( nAxisNo, cemAB_ ENC, CE_TRUE )
        End If
    End If

    ' PA/PB 신호 입력 회로에 노이즈 필터 기능 적용
    If cemCfgFilterAB_Get ( nAxisNo, cemAB_ PULSAR, nFilterEnable ) = ceERR_NONE Then

        If nFilterEnable <> CE_TRUE Then
            ' PA/PB Noise Filter Enable
            Call cemCfgFilterAB_Set ( nAxisNo, cemAB_PULSAR, CE_TRUE )
        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 cemCfgFilterAB_Get ( nAxisNo, cemAB_ENC, @nFilterEnable ) = ceERR_NONE then
    begin
        if nFilterEnable <> CE_TRUE then
        begin
            // EA/EB Noise Filter Enable
            cemCfgFilterAB_Set ( nAxisNo, cemAB_ ENC, CE_TRUE );
        end;
    end;

    // PA/PB 신호 입력 회로에 노이즈 필터 기능 적용
    if cemCfgFilterAB_Get ( nAxisNo, cemAB_ PULSAR, @nFilterEnable ) = ceERR_NONE then
    begin
        if nFilterEnable <> CE_TRUE then
        begin
            // PA/PB Noise Filter Enable
            cemCfgFilterAB_Set ( nAxisNo, cemAB_PULSAR, CE_TRUE );
        end;
    end;

end;
```

NAME

cemCfgInMode_Set / cemCfgInMode_Get

- 인코더 펄스 신호 입력 모드 설정 및 반환

INFORMATION	
1	Environment Config.
!	VC++ (6, 7, 8)/VB
	BCB/Delphi
:	Level 2
J	위험 요소 없음

SYNOPSIS

r VT_I4 cemCfgInMode_Set ([in] VT_I4 Axis, [in] VT_I4 InputMode, [in] VT_I4 IsReverse)

r VT_I4 cemCfgInMode_Get ([in] VT_I4 Axis, [out] VT_PI4 InputMode,

[out] VT_PI4 IsReverse)

DESCRIPTION

cemCfgInMode_Set /cemCfgInMode_Get 함수는 Feedback Pulse 의 입력 모드를 설정 혹은 반환합니다.
Feedback 펄스는 실제 모터 또는 구조물의 구동 거리를 확인하기 위하여 사용되며 거의 대부분 인코더 입력을 사용합니다. 사용자는 4 가지 형태의 Feedback 펄스의 입력 모드를 설정할 수 있습니다.

또한 이 함수는 입력 펄스의 로직(Feedback Count 값의 UP/DOWN 방향)을 설정합니다.

PARAMETER

▶ ***Axis*** : 축 번호. 통합 축으로 관리되는 축 번호를 의미하며, 상수 값으로 0 (Zero Based) 이상, (최대 통합 축 개수 – 1) 이하의 값을 축 번호로 설정할 수 있습니다.

▶ ***InputMode*** : Feedback Pulse 입력 모드를 설정 혹은 반환합니다. 설정 가능한 입력 모드는 다음과 같습니다.

Value	Meaning
0 (cemIMODE_AB1X)	1X A/B (1 채배 엔코더 입력 모드)
1 (cemIMODE_AB2X)	2X A/B (2 채배 엔코더 입력 모드)
2 (cemIMODE_AB4X)	4X A/B (4 채배 엔코더 입력 모드)
3 (cemIMODE_CWCCW)	CW/CCW (A 펄스 - 카운트 증가, B 펄스 - 카운트 감소)
4 (cemIMODE_STEP)	이 모드에서는 Feedback 위치 값을 읽으면 Command 위치값이 바이패스(bypass) 됩니다. 엔코더 피드백이 없는 경우(스텝모터)에 이 모드를 선택합니다.

▶ ***IsReverse*** : Feedback Count 값의 UP/DOWN 방향을 반대로 할 것인지 설정 혹은 반환합니다.

Value	Meaning
0 (CE_FALSE)	Feedback count 의 UP/DOWN 방향이 반대가 아닙니다.
1 (CE_TRUE)	Feedback count 의 UP/DOWN 방향이 반대입니다.

RETURN VALUE

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

EXAMPLE

C/C++

```
#include "ceSDK.h"
#include "ceSDKDef.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 ( cemCfgInMode_Get ( nAxisNo, &nInputMode, &nIsReverse ) == ceERR_NONE )
    {
        if ( nInputMode != cemIMODE_AB4X )
        {
            cemCfgInMode_Set ( nAxisNo, cemIMODE_AB4X, CE_FALSE );
        }
    }
}
```

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.

    long 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 cemCfgInMode_Get ( nAxisNo, nInputMode, nIsReverse ) = ceERR_NONE Then

        If nInputMode <> cemIMODE_AB4X Then
            Call cemCfgInMode_Set ( nAxisNo, cemIMODE_AB4X, CE_FALSE )
        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 cemCfgInMode_Get ( nAxisNo, @nInputMode, @nIsReverse ) = ceERR_NONE then
    begin
            if nInputMode <> cemIMODE_AB4X then
            begin
                    cemCfgInMode_Set ( nAxisNo, cemIMODE_AB4X, CE_FALSE );
            end;
    end;

end;
```

NAME

cemCfgOutMode_Set / cemCfgOutMode_Get

- 지령 펄스(Command Pulse) 신호 출력 모드 설정 및 반환

INFORMATION	
1	Environment Config.
!	VC++ (6, 7, 8)/VB
	BCB/Delphi
:	Level 2
K	다소 주의

SYNOPSIS

r VT_I4 cemCfgOutMode_Set ([in] VT_I4 Axis, [in] VT_I4 OutputMode)

r VT_I4 cemCfgOutMode_Get ([in] VT_I4 Axis, [out] VT_PI4 OutputMode)

DESCRIPTION

cemCfgOutMode_Set 함수는 Command Pulse 의 출력 모드를 설정합니다. Command 신호에 대한 하드웨어적인 회로는 별도로 제공되는 HW 매뉴얼의 “COMMAND 신호 (CW, CCW)” 단원을 참조하여 주시기 바랍니다.

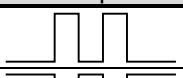
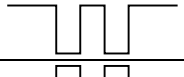
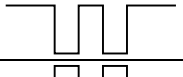
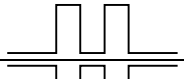
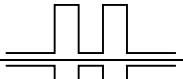
cemCfgOutMode_Get 함수는 현재 설정된 Command Pulse 출력 모드를 반환합니다.

PARAMETER

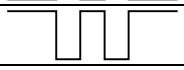
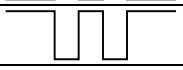
▶ ***Axis*** : 축 번호. 통합 축으로 관리되는 축 번호를 의미하며, 상수 값으로 0 (Zero Based) 이상, (최대 통합 축 개수 - 1) 이하의 값을 축 번호로 설정할 수 있습니다.

▶ ***OutputMode*** : Command Pulse 출력 모드의 설정 값입니다. 출력 모드는 다음과 같이 6 가지 모드를 가집니다.

가) Pulse & Direction Mode

Value	출력 형태			
	(+) 방향 운전 시		(-) 방향 운전 시	
	CW pin	CCW pin	CW pin	CCW pin
0 (cemOMODE_PDIR0)		(Low)		(High)
1 (cemOMODE_PDIR1)		(Low)		(High)
2 (cemOMODE_PDIR2)		(High)		(Low)
3 (cemOMODE_PDIR3)		(High)		(Low)

나) CW/CCW Mode

Value	출력 형태			
	(+) 방향 운전 시		(-) 방향 운전 시	
	CW pin	CCW pin	CW pin	CCW pin
4 (cemOMODE_CWCCW0)		(Low)	(Low)	
5 (cemOMODE_CWCCW1)		(High)	(High)	

RETURN VALUE

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

REFERENCE

□ 모든 경우에 해당하는 부분은 아니지만, 지령 펄스(Command Pulse)의 출력 설정 형태는 일반적으로 서보드라이버에서는 4 혹은 5 번 모드인 CW/CCW 모드를 주로 사용하며, 스텝드라이버에서는 0 번에서 3 번까지의 OUT/DIR 모드를 주로 사용합니다.

□ 일반적인 서보드라이버에서 CW/CCW 혹은 OUT/DIR 모드를 결정하기 위한 지령 펄스(Command Pulse) 입력 설정이 존재합니다. 서보드라이버의 매뉴얼을 반드시 참조하여, ㈜커미조아 모션 모듈과 동일하게 설정하여 주십시오. 동일하게 설정이 되지 않았을 경우에는, 다음과 같은 현상이 발생할 수 있습니다.

1. 서보드라이버를 통해 동작하는 모터의 방향이 한 방향으로만 동작하고 다른 방향으로는 동작하지 않습니다. 즉, 음의 방향(Negative Direction) 으로는 동작하는데 양의 방향(Positive Direction) 으로는 동작하지 않거나 그 반대의 경우를 의미합니다.

2. 방향이 전환 될 때마다 1 펄스(Pulse) 이내의 지령 펄스(Command Pulse) 입력을 무시하게 됩니다. 이 현상을 확인하시려면, 서보드라이버의 Command Pulse Counter 와 ㈜커미조아의 모션 모듈의 Command Pulse Counter 와의 연속적인 방향전환 이동 시에 Pulse Counter 차이를 모니터링(Monitoring) 하시면 됩니다.

이외에도 스텝드라이버의 경우에는 드라이버 입력부 회로의 사양에 따라 Open Collector 연결을 주로 사용하는데, 결선 방법과 상태에 따라서 일반적인 모터의 움직임에 이상 상황이 발생할 수 있습니다. 이때에는 해당 스텝드라이버의 입력부 사양을 첨부하여, ㈜커미조아 고객 지원 팀으로 문의해 주시면 친절하게 상담하여 드리겠습니다.

EXAMPLE

C/C++

```
#include “ceSDK.h”
#include “ceSDKDef.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, CWCCW0 (4 번 모드)로 설정합니다. */

    if ( cemCfgOutMode_Get ( nAxisNo, &nOutputMode ) == ceERR_NONE )
    {
            if ( nOutputMode != cemOMODE_CWCCW0 )
            {
                    cemCfgOutMode_Set ( nAxisNo, cemOMODE_CWCCW0 );
            }
    }
}
```

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, CWCCW0 (4 번 모드)로 설정합니다.

    If cemCfgOutMode_Get ( nAxisNo, nOutputMode ) = ceERR_NONE Then

            If nOutputMode <> cemOMODE_CWCCW0 Then
                    Call cemCfgOutMode_Set ( nAxisNo, cemOMODE_CWCCW0 )
            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, CWCCW0 (4 번 모드)로 설정합니다.

    if cemCfgOutMode_Get ( nAxisNo, @nOutputMode ) = ceERR_NONE then
    begin
            if nOutputMode <> cemOMODE_CWCCW0  then
            begin
                    cemCfgOutMode_Set ( nAxisNo, cemOMODE_CWCCW0 );
            end;
    end;

end;
```

NAME

cemCfgCtrlMode_Set / cemCfgCtrlMode_Get

- 각 축의 제어 모드 설정 및 반환

INFORMATION
1 Environment Config.
! VC++ (6, 7, 8)/VB
BCB/Delphi
: Level 2
K 다소 주의
이송 목표 좌표의 기준을 설정합니다.

SYNOPSIS

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

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

DESCRIPTION

cemCfgCtrlMode_Set 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.

cemCfgCtrlMode_Get 함수는 각 축의 제어모드에 대하여 현재 원격 노드에 설정되어 있는 값을 읽어 들이는 함수입니다.

PARAMETER

▶ ***Axis*** : 축 번호. 통합 축으로 관리되는 축 번호를 의미하며, 상수 값으로 0 (Zero Based) 이상, (최대 통합 축 개수 - 1) 이하의 값을 축 번호로 설정할 수 있습니다.

▶ ***CtrlMode*** : cemCfgCtrlMode_Set 함수의 인자이며, 제어 모드를 설정합니다. 이 값의 의미는 다음과 같습니다.

Value	Meaning
0 (cemCTRL_OPEN) **[Default]**	Open Loop 제어 모드로 설정합니다. 이 모드에서는 이송명령의 목표 좌표가 항상 커맨드 위치를 기준으로 설정됩니다.
1 (cemCTRL_SEMI_C)	Semi-Closed Loop 제어 모드로 설정합니다. 이 모드에서는 이송 명령의 목표 좌표가 항상 피드백 위치를 기준으로 설정됩니다. 예를 들어서 10000 의 좌표로 이송하라는 명령이 하달되었을 때 커맨드 위치는 무시하고, 피드백 위치가 10000 이 되도록 이송의 목표좌표가 설정됩니다.
2 (cemCTRL_FULL_C)	이 값은 향후 버전을 위해서 예약된 값이며, 현재는 사용되지 않습니다.

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

Value	Meaning
0 (cemCTRL_OPEN) **[Default]**	Open Loop 제어 모드입니다. 이 모드에서는 이송명령의 목표 좌표가 항상 커맨드 위치를 기준으로 설정됩니다.
1 (cemCTRL_SEMI_C)	Semi-Closed Loop 제어 모드입니다. 이 모드에서는 이송명령의 목표 좌표가 항상 피드백 위치를 기준으로 설정됩니다.

RETURN VALUE

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

SEE ALSO

cemSxMoveTo, cemSxMoveToStart, cemIxLineTo, cemIxLineToStart

REFERENCE

□ 여기서 설정하는 제어모드는 절대 좌표 이송 명령에서만 영향을 미치고, 상대 좌표 이송 명령에서는 항상 Open Loop 제어 모드로 운용됩니다.

□ 제어 모드에 따른 차이에 대한 이해를 돕기 위하여 예를 들어 보겠습니다. 현재 커맨드 위치가 5000, 피드백 위치가 4000 인 상태에서 cemSxMoveTo() 함수를 사용하여 10000 의 위치로 이송하 라는 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 차이만큼 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 “ceSDK.h”
#include “ceSDKDef.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 ( cemCfgCtrlMode_Get ( nAxisNo, nCtrlMode ) == ceERR_NONE )
    {
            if ( nCtrlMode != cemCTRL_SEMI_C )
            {
                    // Semi-Closed Loop 제어 모드로 설정
                    cemCfgCtrlMode_Set ( nAxisNo, cemCTRL_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 cemCfgCtrlMode_Get ( nAxisNo, nCtrlMode ) = ceERR_NONE Then

            If nCtrlMode <> cemCTRL_SEMI_C Then
                    ‘ Semi-Closed Loop 제어 모드로 설정
                    Call cemCfgCtrlMode_Set ( nAxisNo, cemCTRL_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 cemCfgCtrlMode_Get ( nAxisNo, @nCtrlMode ) = ceERR_NONE then
    begin
            if nCtrlMode <> cemCTRL_SEMI_C then
            begin
                    // Semi-Closed Loop 제어 모드로 설정
                    cemCfgCtrlMode_Set ( nAxisNo, cemCTRL_OPEN );
            end;
    end;

end;
```

NAME

cemCfgInOutRatio_Set / cemCfgInOutRatio_Get
- 입력 펄스와 출력 펄스의 분해능 비율
(Resolution ratio) 설정 및 반환

INFORMATION	
1	Environment Config.
!	VC++ (6, 7, 8)/VB
	BCB/Delphi
:	Level 2
J	위험 요소 없음

SYNOPSIS

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

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

DESCRIPTION

cemCfgInOutRatio_Set 함수는 입력 펄스(Feedback Pulse)와 출력 펄스(Command Pulse)의 분해능 비율(Resolution ratio) 을 설정합니다. 여기서 입력 펄스의 분해능이란 엔코더의 1 회전 시에 발생되는 펄스 수를 의미하며, 출력 펄스의 분해능이란 모터를 1 회전시키기 위해 필요한 출력 펄스 수를 의미합니다.

cemCfgInOutRatio_Set() 함수는 설정되어있는 입력 펄스와 출력 펄스의 분해능 비율을 반환합니다.

PARAMETER

▶ ***Axis*** : 축 번호. 통합 축으로 관리되는 축 번호를 의미하며, 상수 값으로 0 (Zero Based) 이상, (최대 통합 축 개수 - 1) 이하의 값을 축 번호로 설정할 수 있습니다.

▶ ***Ratio*** : 입력 펄스와 출력 펄스의 분해능 비율을 설정 혹은 반환합니다. 이 값은 아래와 같이 설정합니다.

Ratio = (Feedback pulse resolution) / (Command pulse resolution)

RETURN VALUE

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

SEE ALSO

cemStPosition_Get, cemStSpeed_Get

REFERENCE

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

□ In/Out Ratio 는 cemStPosition_Get 함수와 cemStSpeed_Get 함수에서 대상 카운터를 입력 펄스 카운터(cemCNT_FEED)로 설정 한 경우에만 영향을 미칩니다.

EXAMPLE

C/C++

```
#include “ceSDK.h”
#include “ceSDKDef.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
    cemStPosition_Get, cemStSpeed_Get 함수의 대상 카운터가 입력 펄스일 경우에만 영향을 미칩니다. */


    // 분해능 비율을 얻어와 분해능 비율을 ‘1’로 설정합니다.
    if ( cemCfgInOutRatio_Get ( nAxisNo, &fInOutRatio ) != ceERR_NONE )
    {
            if ( fInOutRatio != 1.0f )
            {
                    cemCfgInOutRatio_Set ( 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
    ‘ cemStPosition_Get, cemStSpeed_Get 함수의 대상 카운터가 입력 펄스일 경우에만 영향을 미칩니다.


    ‘ 분해능 비율을 얻어와 분해능 비율을 ‘1’로 설정합니다.
    If cemCfgInOutRatio_Get ( nAxisNo, fInOutRatio ) <> ceERR_NONE Then
```

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

End Sub
```

Visual Basic

```
private 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
    cemStPosition_Get, cemStSpeed_Get 함수의 대상 카운터가 입력 펄스일 경우에만 영향을 미칩니다. }


    // 분해능 비율을 얻어와 분해능 비율을 '1'로 설정합니다.
    if cemCfgInOutRatio_Get ( nAxisNo, fInOutRatio ) <> ceERR_NONE then
    begin
        if fInOutRatio <> 1 then
        begin
            cemCfgInOutRatio_Set ( nAxisNo, 1 );
        end;
    end;

end;
```

NAME

cemCfgUnitDist_Set / cemCfgUnitDist_Get

- 논리적 단위 거리 설정 및 반환

INFORMATION	
1	Environment Config.
!	VC++ (6, 7, 8)/VB
	BCB/Delphi
:	Level 2
J	위험 요소 없음

SYNOPSIS

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

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

DESCRIPTION

cemCfgUnitDist_Set 함수는 논리적 단위 거리에 대한 펄스 수를 설정합니다. 여기서 논리적 단위 거리라 함은 이송 함수에서 사용하는 거리 또는 위치에 대한 단위량을 의미합니다. 이 함수를 사용하여 논리적 단위 거리를 지정하지 않는 경우에는 초기값인 '1' 로 사용됩니다.

cemCfgUnitDist_Get 함수는 논리적 단위 거리에 대한 펄스 수를 반환합니다.

PARAMETER

▶ ***Axis*** : 축 번호. 통합 축으로 관리되는 축 번호를 의미하며, 상수 값으로 0 (Zero Based) 이상, (최대 통합 축 개수 - 1) 이하의 값을 축 번호로 설정할 수 있습니다.

▶ ***UnitDist*** : 논리적 거리 1 을 이송하기 위해서 출력되어야 하는 펄스 수를 설정 혹은 반환합니다.

RETURN VALUE

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

SEE ALSO

cemCfgUnitSpeed_Set, cemCfgUnitSpeed_Get

REFERENCE

☐. Unit distance & Unit speed

모션컨트롤러에서 이송거리는 기본적으로는 출력 펄스 수에 의해 결정되고, 이송 속도는 출력 펄스 주파수에 의해서 결정됩니다. 따라서 이송거리를 물리적인 거리 단위로 하기 위해서는 매번 원하는 물리적인 거리를 이송하기 위해서 필요한 펄스의 수를 계산해야 합니다. 하지만, ㈜커미조아 모션컨트롤러는 모든 이송 함수와 속도 설정 함수에서 사용되는 논리적 거리와 논리적 속도를 사용자가 정의할 수 있도록 하고 있으며, 이것을 정의하는 것이 “Unit distance”와 “Unit speed” 입니다. 일반적으로 “Unit distance” Du 는 다음과 같이 계산하면 됩니다.

$$D_u = P_r / L_r$$

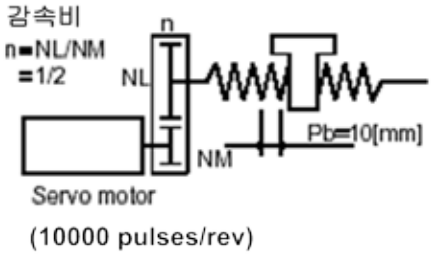
Pr : 모터 1 회전에 필요한 펄스 수. (모터의 Command 분해능)

Lr : 모터 1 회전시에 이동되는 기구물의 거리.

결과적으로 “Unit distance” 에 설정해야 하는 값은 논리적 거리 ‘1’을 이동하기 위해서 필요한 출력 펄스의 수가 됩니다. 그리고 특별한 경우가 아니면 “Unit speed” 는 “Unit distance” 와 같은 값을 설정하면 됩니다. 하지만 “Unit speed”와 “Unit distance”는 필요에 따라 서로 다른 단위를 사용할 수 있습니다.

☐. Unit distance & 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 값이 무한소수(나누어 떨어지지 않음)이면 소수점 오차가 누적될 수 있습니다. 따라서 이러한 경우에는 Unit distance 를 ‘1’로 하고 사용자가 논리적 단위 거리를 처리하는 것이 바람직합니다.

EXAMPLE

C/C++

```
#include “ceSDK.h”
#include “ceSDKDef.h”

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

void OnSetUnitDist ()
{
    /*
    모터 1 회전시 이동거리 (Lr) = 5mm, 모터 Command 분해능(Pr) = 10000 pulse/rev 인 기구물이 있다면

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

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

    cemCfgUnitDist_Set ( nAxisNo, 2000 );         // Unit Distance 를 2000 으로 설정
    cemCfgUnitSpeed_Set ( nAxisNo, 2000 );     // Unit Speed 를 2000 으로 설정
}

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

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

Visual Basic

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

Private Sub OnSetUnitDist ()

    ‘ 모터 1 회전시 이동거리 (Lr) = 5mm, 모터 Command 분해능(Pr) = 10000 pulse/rev 인 기구물이 있다면
    ‘ Unit distance(Du) = Pr/Lr = 10000/5 = 2000
    ‘ Unit speed(Vu) = 2000

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

    Call cemCfgUnitDist_Set ( nAxisNo, 2000 )          ‘ Unit Distance 를 2000 으로 설정
    Call cemCfgUnitSpeed_Set ( 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 cemCfgSpeedPattern_Set ( nAxisNo, cemSMODE_T, 100, 1000, 1000 )

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

End Sub
```

Delphi

```
procedure OnSetUnitDist ();
begin
    { 모터 1 회전시 이동거리 (Lr) = 5mm, 모터 Command 분해능(Pr) = 10000 pulse/rev 인 기구물이 있다면
            Unit distance(Du) = Pr/Lr = 10000/5 = 2000
            Unit speed(Vu) = 2000

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

    cemCfgUnitDist_Set ( cemX1, 2000 );          // Unit Distance 를 2000 으로 설정
    cemCfgUnitSpeed_Set ( cemX1, 2000 );         // Unit Speed 를 2000 으로 설정

end;


procedure OnMove ();
begin
    // 속도 = 100(mm/s), 가/감속도 = 1000(mm/s^2)으로 설정합니다.
    cemCfgSpeedPattern_Set ( cemX1, cemSMODE_T, 100, 1000, 1000 );

    // (+)방향으로 20mm 이송합니다. (실제는 20 * 2000 = 40000 펄스가 출력됩니다.)
    cemSxMoveStart ( cemX1, 20 );

end;
```

NAME

cemCfgUnitSpeed_Set / cemCfgUnitSpeed_Get

- 논리적 속도 단위 설정 및 반환

INFORMATION	
1	Environment Config.
!	VC++ (6, 7, 8)/VB
	BCB/Delphi
:	Level 2
J	위험 요소 없음

SYNOPSIS

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

r VT_I4 cemCfgUnitSpeed_Get ([in] VT_I4 Axis, [out] VT_PR8 UnitSpeed)

DESCRIPTION

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

cemCfgUnitSpeed_Get 함수는 논리적 단위 속도에 대한 실제 펄스 출력속도(PPS) 설정 상태를 반환합니다.

PARAMETER

▶ ***Axis*** : 축 번호. 통합 축으로 관리되는 축 번호를 의미하며, 상수 값으로 0 (Zero Based) 이상, (최대 통합 축 개수 - 1) 이하의 값을 축 번호로 설정할 수 있습니다.

▶ ***UnitSpeed*** : 단위 속도에 대한 펄스 출력 속도를 설정 혹은 반환합니다.

RETURN VALUE

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

SEE ALSO

cemCfgUnitDist_Set, cemCfgUnitDist_Get

REFERENCE

□ 사용자의 특성에 따라 속도에 대한 단위가 다를 수 있습니다. 즉, 어떤 사용자는 속도 단위를 RPM 으로 표현하는 것이 용이할 수 있고 어떤 사용자는 m/sec 로 표현하는 것이 용이할 수 있습니다. cemCfgUnitSpeed_Set 함수는 사용자가 속도의 단위를 결정하도록 하는 함수입니다. 다음의 예를 참고하시기 바랍니다.

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

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

EXAMPLE

C/C++

```
#include “ceSDK.h”
#include “ceSDKDef.h”

long nAxisNo = 0;     // 대상 축을 선택합니다.

void OnSetUnitSpeed ()
{
    /* 1 회전에 필요한 펄스수가 3600 펄스일 때
    거리의 단위를 각도 (°)로, 속도의 단위를 RPM 으로 설정합니다. */

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

    // 단위 속도를 1rpm 으로 설정합니다. 3600(pulse) / 60(sec) = 60(pps)
    cemCfgUnitSpeed_Set ( nAxisNo, 60 );
}


void OnMove ()
{
    /* 가속도와 감속도를 각각 200rpm/s 로 설정합니다. 이렇게 하면 작업 속도가 100rpm 이므로
    가속및 감속에는 0.5sec 가 소요됩니다.*/
    cemCfgSpeedPattern_Set ( nAxisNo, cemSMODE_T, 100, 200, 200 );

    // 모터를 720° 회전 시킵니다. 실제로는 720 * 10 pulse 가 출력됩니다.
    cemSxMoveStart ( nAxisNo, 720 );
}
```

Visual Basic

```
Private Sub OnSetUnitSpeed ()

    ‘ 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.
    Call cemCfgUnitDist_Set ( cemX1, 10 )

    ‘ 단위 속도를 1rpm 으로 설정합니다. 3600(pulse) / 60(sec) = 60(pps)
    Call cemCfgUnitSpeed_Set ( cemX1, 60 )

End Sub


Private Sub OnMove ()

    ‘ 가속도와 감속도를 각각 200rpm/s 로 설정합니다. 이렇게 하면 작업 속도가 100rpm 이므로
    ‘가속및 감속에는 0.5sec 가 소요됩니다.
    Call cemCfgSpeedPattern_Set ( cemX1, cemSMODE_T, 100, 200, 200 )

    ‘ 모터를 720˚ 회전 시킵니다. 실제로는 720 * 10 pulse 가 출력됩니다.
    Call cemSxMoveStart ( cemX1, 720 )

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.
    cemCfgUnitDist_Set ( cemX1, 10 );

    // 단위 속도를 1rpm 으로 설정합니다. 3600(pulse) / 60(sec) = 60(pps)
    cemCfgUnitSpeed_Set (cemX1, 60 );

end;


procedure OnMove ();
begin
    { 가속도와 감속도를 각각 200rpm/s 로 설정합니다. 이렇게 하면 작업 속도가 100rpm 이므로
    가속및 감속에는 0.5sec 가 소요됩니다. }
    cemCfgSpeedPattern_Set (cemX1, cemSMODE_T, 100, 200, 200 );

    // 모터를 720˚ 회전 시킵니다. 실제로는 720 * 10 pulse 가 출력됩니다.
    cemSxMoveStart (cemX1, 720 );

end;
```

NAME

cemCfgSpeedRange_Set /

cemCfgSpeedRange_Get

- 모션 속도 제한 범위 설정 및 반환

INFORMATION	
1	Environment Config.
!	VC++ (6, 7, 8)/VB
	BCB/Delphi
:	Level 2
J	위험 요소 없음

SYNOPSIS

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

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

DESCRIPTION

cemCfgSpeedRange_Set 함수는 모션에 적용할 수 있는 최저/최고 속도를 설정합니다. 이 함수는 실제적으로는 출력 펄스의 주파수 범위를 설정하는 역할을 합니다. 출력 펄스의 주파수는 최대 6.5MHz 까지 설정 가능 하며 기본적으로 설정되는 주파수 범위는 10Hz ~ 655,350Hz 입니다. 최저 속도는 최고 속도 설정에 따라서 자동으로 결정됩니다.

cemCfgSpeedRange_Get 함수는 현재 설정되어 있는 최저/최고 속도의 범위를 반환합니다.

PARAMETER

▶ ***Axis*** : 축 번호. 통합 축으로 관리되는 축 번호를 의미하며, 상수 값으로 0 (Zero Based) 이상, (최대 통합 축 개수 – 1) 이하의 값을 축 번호로 설정할 수 있습니다.

▶ ***MaxPPS*** : 모션의 최고 속도를 PPS 단위로 설정 혹은 반환합니다.

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

▶ ***MinPPS*** : cemCfgSpeedRange_Get() 함수의 인자이며, 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 관계는 다음과 같습니다.

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

RETURN VALUE

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

EXAMPLE

C/C++

```
#include "ceSDK.h"
#include "ceSDKDef.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( cemCfgSpeedRange_Get ( nAxisNo, &fMinPPS, &fMaxPPS ) == ceERR_NONE )
    {
            if ( fMaxPPS < 6553500 )
            {
                    // 모션 최대 속도를 6553500 으로 설정합니다.
                    cemCfgSpeedRange_Set ( 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 cemCfgSpeedRange_Get ( nAxisNo, fMinPPS, fMaxPPS ) = ceERR_NONE Then

            If fMaxPPS < 6553500 Then
                    ' 모션 최대 속도를 6553500 으로 설정합니다.
                    Call cemCfgSpeedRange_Set ( 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 cemCfgSpeedRange_Get ( nAxisNo, @fMinPPS, @fMaxPPS ) = ceERR_NONE then
    begin
            if fMaxPPS < 6553500 then
            begin
                    // 모션 최대 속도를 6553500 으로 설정합니다.
                    cemCfgSpeedRange_Set ( nAxisNo, 6553500 );
            end;
    end;

end;
```

NAME

cemCfgSpeedPattern_Set /

cemCfgSpeedPattern_Get

- 모션 이송 기준 속도 설정 및 반환

INFORMATION	
1	Environment Config.
!	VC++ (6, 7, 8)/VB
	BCB/Delphi
:	Level 2
J	위험 요소 없음

SYNOPSIS

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

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

DESCRIPTION

cemCfgSpeedPattern_Set 함수는 지정한 축에 대해 속도 모드, 작업속도, 가속도 및 감속도를 설정합니다. 이 속도는 각 모션제어의 기준 속도로 설정됩니다.

cemCfgSpeedPattern_Get 함수는 지정한 축에 대해 설정되어 있는 속도 설정 환경을 반환합니다.

PARAMETER

▶ ***Axis*** : 축 번호. 통합 축으로 관리되는 축 번호를 의미하며, 상수 값으로 0 (Zero Based) 이상, (최대 통합 축 개수 - 1) 이하의 값을 축 번호로 설정할 수 있습니다.

▶ ***SpeedMode*** : cemCfgSpeedPattern_Set 함수의 인자이며, 속도 모드를 설정합니다. 속도 모드는 다음과 같습니다.

Value	Meaning
0 (cemSMODE_C)	CONSTANT 속도모드 => 가감속을 수행하지 않습니다.
1 (cemSMODE_T)	TRAPEZOIDAL 속도모드 => 사다리꼴 가감속을 수행합니다.
2 (cemSMODE_S)	S-CURVE 속도모드 => S-CURVE 가감속을 수행합니다.
-1 (cemSMODE_KEEP)	이전 속도 모드를 유지합니다.

▶ ***SpeedMode*** : cemCfgSpeedPattern_Get 함수의 인자이며, 설정되었는 속도 모드를 반환합니다.

Value	Meaning
0 (cemSMODE_C)	CONSTANT 속도모드 => 가감속을 수행하지 않습니다.
1 (cemSMODE_T)	TRAPEZOIDAL 속도모드 => 사다리꼴 가감속을 수행합니다.
2 (cemSMODE_S)	S-CURVE 속도모드 => S-CURVE 가감속을 수행합니다.

▶ ***WorkSpeed*** : 작업 속도를 설정 혹은 반환합니다.

▶ ***Accel*** : 가속도를 설정 혹은 반환합니다.

▶ ***Decel*** : 감속도를 설정 혹은 반환합니다.

SEE ALSO

cemSxSpeedRatio_Set, cemSxSpeedRatio_Get

RETURN VALUE

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

REFERENCE

□ 단축 모션 제어의 경우 cemSxSpeedRatio_Set 함수를 통해 설정된 모션 이송 기준 속도의 비율로 모션 제어의 속도를 결정할 수 있습니다.

□ 원점 복귀 제어는 cemHomeSpeedPattern_Set 함수를, 보간 제어는 cemIxSpeedPattern_Set 함수를 통해 작업 속도를 설정합니다.

□ 한번 설정한 속도설정은 변경하기 전까지 계속해서 적용됩니다. 따라서 속도를 변경할 필요가 없는 경우에는 이송명령을 수행할 때마다 속도설정을 해줄 필요는 없습니다.

□ 속도와 가/감속도의 단위는 cemCfgUnitSpeed_Set 함수에 의해 결정됩니다.

□ CONSTANT Speed Mode

Constant speed mode 에서는 가속/감속을 적용하지 않고 일정 속도로 모션 이송 작업을 수행합니다.
여기서 적용되는 일정 속도는 WorkSpeed 에서 주어진 값이 적용됩니다.

□ TRAPEZOIDAL Speed Mode

Trapezoidal speed mode 에서는 모션 이송 작업을 수행하는데 있어, 속도 패턴을 [그림 6-1]과 같이 Linear acceleration -> Working speed(constant) -> Linear deceleration 의 형태로 운용하는 모드입니다.

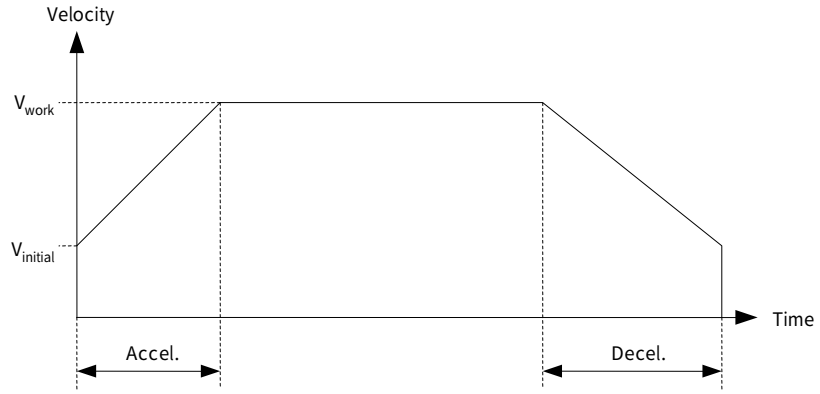


그림 6-1 Trapezoidal speed pattern

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

□ S-CURVE Speed Mode

S-curve speed mode 에서는 모션 이송 작업을 수행할 때 S 자 형태로 가속과 감속을 수행합니다. S-curve speed mode 에서 가(감)속 구간은 [그림 6-2] 와 같이 S-curve section 과 Linear acceleration section 으로 구성됩니다.

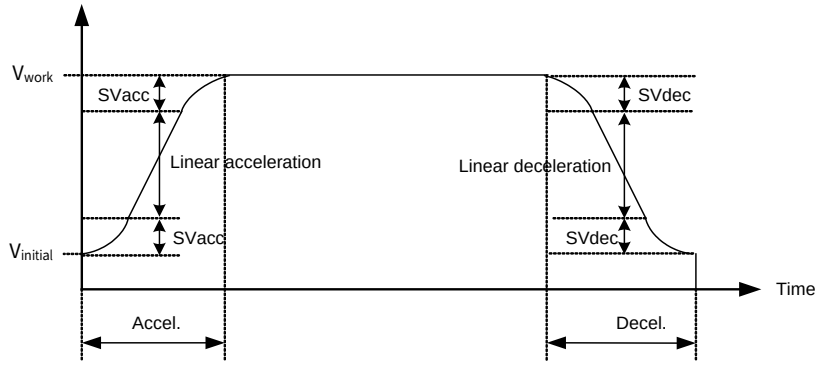


그림 6-2 S-curve speed pattern

※ S-curve speed mode 에서는 설정한 가(감)속 값이 S-curve section 을 포함한 전체 가(감)속 시간을 결정하는 매개변수로 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

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 “ceSDK.h”
#include “ceSDKDef.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.*/

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

    // 설정되어있는 속도 패턴 및 작업속도, 가속도, 감속도를 반환합니다.
    cemCfgSpeedPattern_Get ( 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 cemCfgSpeedPattern_Set ( nAxisNo, cemSMODE_S, 2000, 10000, 10000 )

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

End Sub
```

Delphi

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

begin
    { 0 번축의 속도 패턴을 S-Curve 로 설정하고,
    작업속도를 2000, 가속도를 10000, 감속도를 10000 으로 설정합니다. }
     cemCfgSpeedPattern_Set ( ceX1, cemSMODE_S, 2000, 10000, 10000 );

    ‘ 설정되어있는 속도 패턴 및 작업속도, 가속도, 감속도를 반환합니다.
    cemCfgSpeedPattern_Get ( ceX1, @nSpeedMode, @fVel, @fAcc, @fDec );

end;
```

NAME

cemCfgSoftLimit_Set / cemCfgSoftLimit_Get

- 소프트웨어 이송 제한 범위 설정 및 반환

INFORMATION	
1	Environment Config.
!	VC++ (6, 7, 8)/VB
	BCB/Delphi
:	Level 2
J	다소 주의

SYNOPSIS

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

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

DESCRIPTION

cemCfgSoftLimit_Set 함수는 소프트웨어 리밋(Limit) 기능을 활성 또는 비활성화 하고 소프트웨어 리밋 범위를 설정합니다. 소프트웨어적인 Limit 은 리밋 센서의 설치가 용이하지 않을 때 안전성을 위하여 모션의 이송 범위를 소프트웨어적인 이송제한범위를 설정하여 제한하는 것입니다.

소프트웨어적인 Limit 은 Command pulse 카운터의 절대값이 지정한 +/- Limit 값보다 같거나 크게 되면 모션을 자동 정지하도록 합니다.

cemCfgSoftLimit_Get 함수는 소프트웨어 리밋(Limit) 활성 상태 및 범위 설정상태를 반환합니다.

PARAMETER

▶ ***Axis*** : 축 번호. 통합 축으로 관리되는 축 번호를 의미하며, 상수 값으로 0 (Zero Based) 이상, 최대 통합 축 개수 - 1 이하의 값을 축 번호로 설정할 수 있습니다.

▶ ***IsEnable*** : 소프트웨어 리밋(Limit) 기능의 활성/비활성 여부를 설정 혹은 반환합니다.

▶ ***LimitN*** : (-) 방향 Limit 값을 설정 혹은 반환합니다.

▶ ***LimitP*** : (+) 방향 Limit 값을 설정 혹은 반환합니다.

RETURN VALUE

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

REFERENCE

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

이 내용을 식으로 표현하면 다음과 같습니다.

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

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

SW Limit 제한 범위는 Unit Distance 를 고려하여, 값의 Overflow 를 주의해야 합니다.

EXAMPLE

C/C++

```
#include “ceSDK.h”
#include “ceSDKDef.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 ( cemCfgSoftLimit_Get( nAxisNo, &nIsEnable, &fElNeg, &fElPos ) == ceERR_NONE )
    {
            if ( nIsEnable != CE_TRUE )
            {
                    // Software Limit 기능을 활성화 하고, -EL 은 -100000, +EL 은 100000 으로 설정합니다.
                    cemCfgSoftLimit_Set ( nAxisNo, CE_TRUE, -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. */
    cemSxMoveStart ( 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 cemCfgSoftLimit_Get ( cemX1, nIsEnable, fElNeg, fElPos ) = ceERR_NONE Then

            If nIsEnable <> CE_TRUE Then
                    ' Software Limit 기능을 활성화 하고, -EL 은 -100000, +EL 은 100000 으로 설정합니다.
                    Call cemCfgSoftLimit_Set ( cemX1, CE_TRUE, -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 cemSxMoveStart ( cemX1, 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 cemCfgSoftLimit_Get ( cemX1, @nIsEnable, @fElNeg, @fElPos ) = ceERR_NONE then
    begin
            if nIsEnable <> CE_TRUE then
                    // Software Limit 기능을 활성화 하고, -EL 은 -100000, +EL 은 100000 으로 설정합니다.
                    cemCfgSoftLimit_Set ( cemX1, CE_TRUE, -100000, 100000 );
            end;
    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. }
    cemSxMoveStart ( cemX1, 100000 );

end;
```

NAME

cemCfgRingCntr_Set / cemCfgRingCntr_Get

- 링 카운터(Ring-Counter) 기능 설정 및 반환

INFORMATION	
1	Environment Config.
!	VC++ (6, 7, 8)/VB
	BCB/Delphi
:	Level 2
J	다소 주의

SYNOPSIS

r VT_I4 cemCfgRingCntr_Set ([in] VT_I4 Channel, [in] VT_I4 TargCntr, [in] VT_I4 IsEnable, [in] VT_R8 CntMax)

r VT_I4 cemCfgRingCntr_Get ([in] VT_I4 Channel, [in] VT_I4 TargCntr,

[out] VT_PI4 IsEnable, [out] VT_PR8 CntMax)

DESCRIPTION

cemCfgRingCntr_Set 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.

cemCfgRingCntr_Get 함수는 링 카운터 기능의 설정 상태를 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.

▶ ***TargCntr*** : 링 카운터 기능 대상 카운터를 설정 혹은 반환합니다.

Value	Meaning
0 (cemCNT_COMM)	Command Counter.
1 (cemCNT_FEED)	Feedback Counter.

▶ ***IsEnable*** : 링 카운터 기능 활성/비활성 여부를 설정 혹은 반환합니다.

Value	Meaning
0 (CE_FALSE)	링 카운터 기능을 사용하지 않습니다.
1 (CE_TRUE)	링 카운터 기능을 사용합니다.

▶ ***CntMax*** : 링 카운터 범위를 설정 혹은 반환합니다. 링 카운터 기능이 활성화되면 지정한 카운터는 0~CntMax 사이의 값에서만 카운트 됩니다.

RETURN VALUE

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

EXAMPLE

C/C++

```
#include "ceSDK.h"
#include "ceSDKDef.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.*/
    cemCfgRingCntr_Set (    nAxisNo,              // 대상 축 설정
                            cemCNT_COMM,          // 링 카운터 대상 카운터 설정
                            CE_TRUE,              // 링 카운터 기능 활성/비활성 설정
                            40000                 // 링 카운터 범위 설정
                            );

    // 해당 축의 링카운터 설정 상태를 반환합니다.
    cemCfgRingCntr_Get ( nAxis, cemCNT_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. */

    cemSxVMoveStart ( nAxisNo );
}
```

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 cemCfgRingCntr_Set ( nAxisNo, cemCNT_COMM, CE_TRUE, 40000 )

    ‘ 해당 축의 링카운터 설정 상태를 반환합니다.
    Call cemCfgRingCntr_Get ( nAxis, cemCNT_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 cemSxVMoveStart ( nAxisNo )

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. }
    cemCfgRingCntr_Set ( cemX1, cemCNT_COMM, CE_TRUE, 40000 );

    // 해당 축의 링카운터 설정 상태를 반환합니다.
    cemCfgRingCntr_Get ( cemX1, cemCNT_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. }

    cemSxVMoveStart ( cemX1 );

end;
```

NAME

cemCfgSeqMode_Set / cemCfgSeqMode_Get

- 시퀀스(Sequence) 모드 설정 및 반환

INFORMATION	
1	Environment Config.
!	VC++ (6, 7, 8)/VB
	BCB/Delphi
:	Level 2
J	위험 요소 없음

SYNOPSIS

r VT_I4 cemCfgSeqMode_Set ([in] VT_I4 SeqMode)

r VT_I4 cemCfgSeqMode_Get ([out] VT_PI4 SeqMode)

DESCRIPTION

cemCfgSeqMode_Set 함수는 현재 이송이 진행되고 있는 축에 새로운 이송 명령이 하달되었을 때, 처리를 어떻게 할 것인지에 대한 모드를 설정합니다.
ceSDK 에서는 현재 이송 명령이 진행되고 있으므로 에러로 처리하는 모드와 이전 이송 명령이 완료될 때까지 내부적으로 루프를 돌면서 기다리다가 이전 명령이 완료되면 새로운 명령을 실행하는 모드를 지원합니다.

cemCfgSeqMode_Get 함수는 설정되어 있는 시퀀스 모드를 반환합니다.

PARAMETER

▶ ***SeqMode*** : 현재 이송이 진행되고 있는 축에 새로운 이송 명령이 하달되었을 때 이의 처리를 어떻게 할 것인지에 대한 시퀀스(Sequence) 모드를 설정 혹은 반환합니다. 이 값의 의미는 다음과 같습니다.

Value	Meaning
0 (cemSEQM_SKIP_RUN) **[Default]**	이송이 진행되고 있는 축에 새로운 이송 명령이 하달되면 -5170 (cemERR_MOT_SEQ_SKIPPED) 에러 값과 함께 이송 명령을 수행하지 않고 곧바로 반환됩니다. 다양한 에러 코드의 확인 본 매뉴얼의 부록 편에 명시되어 있습니다.
1 (cemSEQM_WAIT_RUN)	현재 이송이 진행되고 있는 축에 새로운 이송 명령이 하달되면 이송 함수 내부에서 루프를 돌면서 이전 이송이 완료되기를 기다리다가 이전 이송이 완료되면 현재 하달된 이송 명령을 수행합니다.

RETURN VALUE

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

REFERENCE

시퀀스(Sequence) 모드가 cemSEQM_SKIP_RUN[Default]으로 설정된 경우에 이전 이송 명령이 아직 끝나지 않은 상태에서 새로운 이송 명령이 하달되면 에러 처리되고 해당 이송 명령은 실행되지 않습니다. 따라서 이 모드에서 이송 명령을 내릴때 사용자는 이전의 이송 명령이 완료되었음을 확인하는 것이 바람직합니다.

EXAMPLE

C/C++

```
#include “ceSDK.h”
#include “ceSDKDef.h”

void OnSetSeqMode ()
{
    long nSeqMode;              // 시퀀스 모드 정보.

    // 설정되어있는 시퀀스 모드를 확인 후, cemSEQM_SKIP_RUN 모드로 설정합니다.
    if ( cemCfgSeqMode_Get ( &nSeqMode ) == ceERR_NONE )
    {
            if ( nSeqMode != cemSEQM_SKIP_RUN )
            {
                    cemCfgSeqMode_Set ( cemSEQM_SKIP_RUN )
            }
    }
}


void OnMove ()
{
    SxVMoveStart ( cemX1 );             // 속도 이송 명령 수행

    if ( SxMoveStart ( cemX1, 100000 ) != ceERR_NONE )
    {
            // -5170 에러 코드 반환
            // 시퀀스 모드가 cemSEQM_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, cemSEQM_SKIP_RUN 모드로 설정합니다.
    If cemCfgSeqMode_Get ( nSeqMode ) = ceERR_NONE Then

            If nSeqMode <> cemSEQM_SKIP_RUN Then
                    Call cemCfgSeqMode_Set ( cemSEQM_SKIP_RUN )
            End If
```

```
    End If

End Sub


Private Sub OnMove ()

    Call SxVMoveStart ( cemX1 )                    ' 속도 이송 명령 수행

    If SxMoveStart ( cemX1, 100000 ) <> ceERR_NONE Then
            ' -5170 에러 코드 반환
            ' 시퀀스 모드가 cemSEQM_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, cemSEQM_SKIP_RUN 모드로 설정합니다.
    if cemCfgSeqMode_Get ( @nSeqMode ) = ceERR_NONE then
    begin
            if nSeqMode <> cemSEQM_SKIP_RUN then
            begin
                    cemCfgSeqMode_Set ( cemSEQM_SKIP_RUN );
            end;
    end;

end;


procedure OnMove ();
begin
    SxVMoveStart ( cemX1 );                ' 속도 이송 명령 수행

    if SxMoveStart ( cemX1, 100000 ) <> ceERR_NONE then
    begin
            { -5170 에러 코드 반환
            시퀀스 모드가 cemSEQM_SKIP_RUN 모드로 설정되어 있으므로
            이송 명령이 완료되지 않은 상태에서 다른 이송 명령은 실행되지 않습니다. }
    end;

end;
```

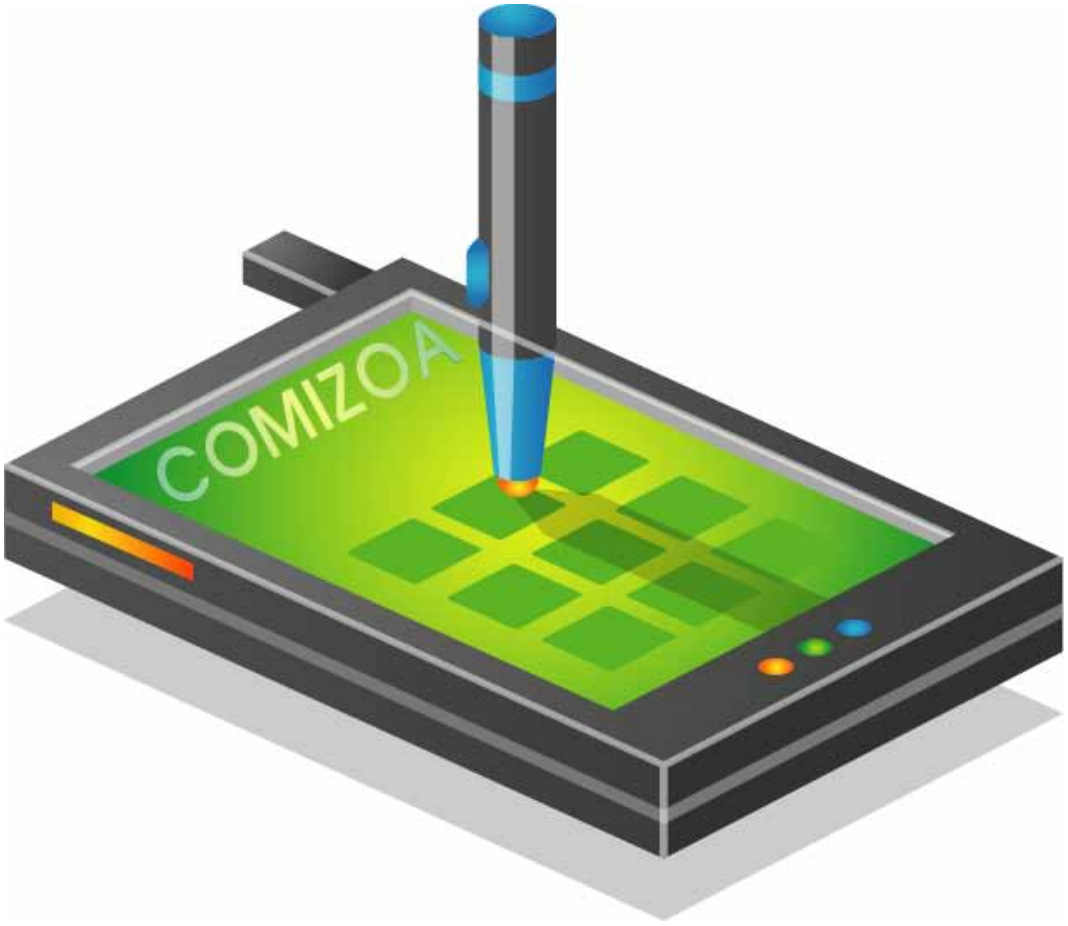
Chapter

7

Basic Motion Control

기본 P2P (Point to Point) 제어는 모션 제어에 있어 모터 구동의 첫걸음 이자, 가장 중요한 부분입니다. 고객여러분들께서는 단축 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 소개합니다. 이 장에서는 단축 모션 제어부터 기본 보간 제어, 원점 복귀 등의 내용으로 구성되어 있습니다. 단축 제어는 단일 축을 독립적으로 제어하는 작업을 의미합니다. 보간 제어는 직선 보간과 원호 보간 기능으로 구성되어 있습니다. 다양한 원점복귀 기능을 통해 초기 위치를 결정할 수 있습니다.



7 기본 모션 제어 편

7.1 단축(Single-Axis) 모션 제어

각 축의 속도 환경을 설정하고, 단축 이송 함수를 사용하여 이송 작업을 수행합니다. 그리고 필요에 따라 정지 함수를 사용하여 이송 중인 축을 정지합니다.

7.1.1 함수 요약

"단축 모션 제어" 에 관련된 함수는 다음과 같습니다.

Summary of Functions
r VT_I4 cemSxSpeedRatio_Set ([in] VT_I4 Axis, [in] VT_I4 SpeedMode, [in] VT_R8 VelRatio, [in] VT_R8 AccRatio, [in] VT_R8 DecRatio) 단축 구동 시 해당 축에 대해 속도 방식 및 속도 비율을 설정합니다.
r VT_I4 cemSxSpeedRatio_Get ([in] VT_I4 Axis, [out] VT_PI4 SpeedMode, [out] VT_PR8 VelRatio, [out] VT_PR8 AccRatio, [out] VT_PR8 DecRatio) 단축 구동 시 해당 축에 대해 속도 방식 및 속도 비율의 설정 상태를 반환합니다.
r VT_I4 cemSxMoveStart ([in] VT_I4 Axis, [in] VT_R8 Distance) 단축 상대 좌표 이송을 시작합니다. 이 함수는 모션 시작 후 바로 반환됩니다.
r VT_I4 cemSxMove ([in] VT_I4 Axis, [in] VT_R8 Distance, [in] VT_I4 IsBlocking) 단축 상대 좌표 이송을 시작합니다. 이 함수는 모션이 완료되기 전까지 반환되지 않습니다.
r VT_I4 cemSxMoveToStart ([in] VT_I4 Axis, [in] VT_R8 Position) 단축 절대 좌표 이송을 시작합니다. 이 함수는 모션 시작 후 바로 반환됩니다.
r VT_I4 cemSxMoveTo ([in] VT_I4 Axis, [in] VT_R8 Position, [in] VT_I4 IsBlocking) 단축 절대 좌표 이송을 시작합니다. 이 함수는 모션이 완료되기 전까지 반환되지 않습니다.
r VT_I4 cemSxVMoveStart ([in] VT_I4 Axis, [in] VT_I4 Direction) 단축 연속 속도 이송을 시작합니다. 이 함수는 모션 시작 후 바로 반환됩니다.
r VT_I4 cemSxStop ([in] VT_I4 Axis, [in] VT_I4 IsWaitComplete, [in] VT_I4 IsBlocking) 단축 이송을 감속 후 정지합니다. 이 정지함수는 이송 완료시까지 대기 할 수 있습니다.
r VT_I4 cemSxStopEmg ([in] VT_I4 Axis) 단축 이송을 비상 정지 합니다. 이 정지 함수는 감속을 무시합니다.
r VT_I4 cemSxIsDone ([in] VT_I4 Axis, [out] VT_PI4 IsDone) 단축 이송의 완료를 확인합니다.
r VT_I4 cemSxWaitDone ([in] VT_I4 Axis, [in] VT_I4 IsBlocking) 단축 이송의 완료 시점까지 대기합니다.
r VT_I4 cemSxTargetPos_Get ([in] VT_I4 Channel, [out] VT_PR8 Position) 해당 통합 축에 대해 마지막으로 이송한(상대 or 절대 좌표) 위치를 반환합니다.
r VT_I4 cemSxOptIniSpeed_Set ([in] VT_I4 Axis, [in] VT_R8 IniSpeed) 단축 모션의 초기 속도를 설정합니다.

r VT_I4 cemSxOptIniSpeed_Get ([in] VT_I4 Axis, [out] VT_PR8 IniSpeed) 단축 모션의 초기 속도 설정을 반환합니다.
r VT_I4 cemSxMoveStart2V ([in] VT_I4 Axis, [in] VT_R8 Distance, [in] VT_R8 Vel2) 단축 모션 상대 좌표 2 단계 속도 이송을 시작합니다. 이 함수는 모션 시작 후 바로 반환됩니다.
r VT_I4 cemSxMoveToStart2V ([in] VT_I4 Axis, [in] VT_R8 Position, [in] VT_R8 Vel2) 단축 모션 절대 좌표 2 단계 속도 이송을 시작합니다. 이 함수는 모션 시작 후 바로 반환됩니다.
r VT_I4 cemSxCorrection_Set ([in] VT_I4 Axis, [in] VT_I4 CorrMode, [in] VT_R8 CorrAmount, [in] VT_R8 CorrVel, [in] VT_I4 CntrMask) 단축 모션의 백래쉬 혹은 슬립 보정을 위해 설정하는 함수입니다.
r VT_I4 cemSxCorrection_Get ([in] VT_I4 Axis, [out] VT_PI4 CorrMode, [out] VT_PR8 CorrAmount, [out] VT_PR8 CorrVel, [out] VT_PI4 CntrMask) 단축 모션의 백래쉬 혹은 슬립 보정의 설정을 반환하는 함수입니다.

7.1.2 함수 설명

NAME

cemSxSpeedRatio_Set / cemSxSpeedRatio_Get

- 단축이송 속도 비율 설정 및 반환

INFORMATION	
1	Single Axis Control
!	VC++ (6, 7, 8)/VB BCB/Delphi
:	Level 3
J	다소 주의

SYNOPSIS

r VT_I4 cemSxSpeedRatio_Set ([in] VT_I4 Axis, [in] VT_I4 SpeedMode, [in] VT_R8 VelRatio, [in] VT_R8 AccRatio, [in] VT_R8 DecRatio)

r VT_I4 cemSxSpeedRatio_Get ([in] VT_I4 Axis, [out] VT_PI4 SpeedMode, [out] VT_PR8 VelRatio, [out] VT_PR8 AccRatio, [out] VT_PR8 DecRatio)

DESCRIPTION

cemSxSpeedRatio_Set 함수는 단축 구동 시 해당 축에 대한 속도 모드 및 속도 비율을 설정 합니다.

모션 속도는 기준 속도의 비율로 설정이 가능하며, 이것은 cemCfgSpeedPattern_Set 함수에 의해서 설정된 기준 속도를 의미합니다.

cemSxSpeedRatio_Get() 함수는 단축 구동 시 해당 축에 대해 설정되어 있는 속도 모드 및 속도 비율을 반환합니다.

PARAMETER

▶ ***Axis*** : 축 번호. 통합 축으로 관리되는 축 번호를 의미하며, 상수 값으로 0 (Zero Based) 이상, (최대 통합 축 개수 - 1) 이하의 값을 축 번호로 설정할 수 있습니다.

▶ ***SpeedMode*** : cemSxSpeedRatio_Set 함수의 인자이며, 속도 모드를 설정합니다. 설정 가능한 속도 모드는 다음과 같습니다.

Value	Meaning
0 (cemSMODE_C)	CONSTANT 속도 모드 => 가감속을 수행하지 않습니다.
1 (cemSMODE_T)	TRAPEZOIDAL 속도 모드 => 사다리꼴 가감속을 수행합니다.
2 (cemSMODE_S)	S-CURVE 속도 모드 => S-CURVE 가감속을 수행합니다.
-1 (cemSMODE_KEEP)	이전 속도 모드를 그대로 유지합니다. 즉, 속도모드를 변경하지 않습니다.

▶ ***SpeedMode*** : cemSxSpeedRatio_Get 함수의 인자이며, 설정된 속도 모드를 반환합니다.

Value	Meaning
0 (cemSMODE_C)	CONSTANT 속도 모드 => 가감속을 수행하지 않습니다.
1 (cemSMODE_T)	TRAPEZOIDAL 속도 모드 => 사다리꼴 가감속을 수행합니다.
2 (cemSMODE_S)	S-CURVE 속도 모드 => S-CURVE 가감속을 수행합니다.

▶ ***VelRatio*** : 작업 속도 비율(Ratio) 을 설정 혹은 반환합니다. 이 값의 단위는 %입니다.

▶ ***AccRatio*** : 가속도 비율(Ratio) 을 설정 혹은 반환합니다. 이 값의 단위는 %입니다.

▶ ***DecRatio*** : 감속도 비율(Ratio) 을 설정 혹은 반환합니다. 이 값의 단위는 %입니다.

RETURN VALUE

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

SEE ALSO

cemCfgSpeedPattern_Set, cemCfgSpeedPattern_Get

REFERENCE

속도 비율(Ratio) 의 정확한 의미를 알고 싶습니다.

속도 설정은 비율로 설정이 됩니다. 비율에 의한 속도 값은 모션 기준 속도 값에 배수(Multiplication)가 되거나 제법(Division) 이 된 속도 값을 의미합니다. 기준이 되는 속도 값은 cemCfgSpeedPattern_Set 으로 설정됩니다.

㈜ 커미조아 ceSDK 에서는 기준 속도(Standard Speed) 개념을 이용하고 있습니다. 모션 속도는 기준 속도의 비율로 설정이 가능하며, 이것은 cemCfgSpeedPattern_Set 함수에 의해서 설정된 기준 속도를 의미합니다. ㈜ 커미조아의 ceSDK 는 기준 속도의 값을 비율(Ratio)로 설정할 수 있는 커다란 이점을 가지고 있습니다.

EXAMPLE

C/C++

```
#include “ceSDK.h”
#include “ceSDKDef.h”

void OnSetSpeed ()
{
    long nAxisNo = 1;            // 모션 이송 속도를 설정할 대상 축을 선택합니다.

    /* 단축 모션의 작업 속도 비율을 설정합니다. cemCfgSpeedPattern_Set()함수를 통해서
    설정된 모션 이송 속도를 기준으로 속도 비율이 적용됩니다. */

    cemCfgSpeedPattern_Set ( nAxisNo, cemSMODE_T, 1000, 10000, 10000 );

    cemSxSpeedRatio_Set (   nAxisNo,          // 대상 축 선택
                            cemSMODE_T,   // 속도 모드 선택
                            50,               // 작업 속도 비율. 1000 * 0.5 = 500 pps
                            80,               // 가속도 비율. 10000 * 0.8 = 8000 pps^2
                            80                // 감속도 비율. 10000 * 0.8 = 8000 pps^2
                            );
}
```

NAME

cemSxMove / cemSxMoveStart

- 단축 상대좌표 이송

INFORMATION	
1	Single Axis Control
!	VC++ (6, 7, 8)/VB
	BCB/Delphi
:	Level 3
J	이송 함수

실제 이송이 진행되는 함수이므로, 사전에 반드시 안전을 확인 합니다.

SYNOPSIS

r VT_I4 cemSxMove ([in] VT_I4 Axis, [in] VT_R8 Distance, [in] VT_I4 IsBlocking)

r VT_I4 cemSxMoveStart ([in] VT_I4 Axis, [in] VT_R8 Distance)

DESCRIPTION

cemSxMove / cemSxMoveStart 함수는 하나의 축에 대하여 현재의 위치에서 지정한 거리(상대 위치) 만큼 이송을 수행합니다.

cemSxMove 함수는 모션이 완료되기 전까지 반환되지 않으며, cemSxMoveStart 함수는 모션을 시작시킨 후에 바로 반환됩니다.

PARAMETER

▶ ***Axis*** : 축 번호. 통합 축으로 관리되는 축 번호를 의미하며, 상수 값으로 0 (Zero Based) 이상, (최대 통합 축 개수 - 1) 이하의 값을 축 번호로 설정할 수 있습니다.

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

▶ ***IsBlocking*** : cemSxMove 함수의 인자이며, 완료될 때까지 기다리는 동안 윈도우 메시지를 블록(Blocking)할 것인지를 결정합니다. 단, 쓰레드 내에서 실행할 때는 이 값을 1(CE_TRUE)로 설정해 주어야 합니다.

Value	Meaning
0 (CE_FALSE)	윈도우 메시지를 블록(Blocking) 하지 않습니다. 따라서 해당 모션이 진행되는 동안 윈도우 이벤트를 처리합니다.
1 (CE_TRUE)	윈도우 메시지를 블록(Blocking) 합니다. 따라서 해당 모션이 완료되는 동안에는 윈도우 이벤트가 처리되지 않습니다.

RETURN VALUE

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

SEE ALSO

cemSxMoveTo, cemSxMoveToStart, cemSxVMoveStart

REFERENCE

□ 논리적 단위 거리는 cemCfgUnitDist_Set 함수에 의해 결정됩니다.

□ cemSxMoveStart 함수를 사용하는 경우에는 cemSxIsDone 함수나 cemSxWaitDone 함수를 사용하여 모션의 완료를 확인할 수 있습니다.

□ cemSxMove 함수를 사용하는 경우에는 내부적으로 루프를 수행하면서 모션이 완료되기를 기다리는데, 이때 블록 모드(Blocking Mode) 설정에 따라 윈도우 이벤트를 처리하는 방식이 달라집니다. 그러나 일반적으로 윈도우의 작업 쓰레드(Work Thread) 에서는 블록 모드를 사용하여, 함수 내부에서 지연 없이 쓰레드 내부의 작업에 집중할 수 있도록 설정하는 것이 바람직합니다.

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

□ 스텝 드라이버를 사용 중인 고객님들께서는 다음을 참조해 주십시오.
스텝 드라이버는 INP 출력이 없는 경우가 일반적인데, 고객님의 부주의나 잘못된 설정으로 INP 입력에 대한 설정이 Enable 로 되어 있을 경우 INP 입력이 스텝 드라이버를 통해 발생하지 않는 이유 때문에 모션 완료가 되지 않는 경우가 발생할 수 있습니다. 고객 여러분들께서는 스텝 드라이버 사용시에 이점을 주의하시기 바랍니다.

□ 서보 드라이버의 LSP, LSN 신호를 사용 중인 고객님께서는 다음을 참조해 주십시오.
서보드라이버의 입력 신호 중 하나인 EL(End of Limit) 신호는 저희 ㈜ 커미조아 모션 컨트롤러뿐만 아니라 서보드라이버에도 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 신호를 서보 드라이버 측에 전달하기 위한 용도로 사용됩니다.

그러나, 모션 소프트웨어에서 INP 설정이 되어 있는 경우 EL 신호가 검출 된 후에 일부 서보 드라이버에서는 진행 방향에서 정지한 후 더 이상 움직이지 않는 상황이 발생하며, 이 상황에서 INP 신호가 출력되지 않아, 모션 이송이 완료되지 못하고, 명시적으로 STOP 명령을 통해 강제적으로 모션을

종료해야 되는 현상이 발생할 수 있습니다. 이 현상은 EL 모드를 통해 원점 복귀를 하는 상황에서도 발생할 수 있습니다.
따라서, 이러한 경우에는 반드시 원점 복귀나 EL 검출 시에 인터럽트 이벤트나 타이머를 통해 INP 를 무기한 대기하는 현상에 대해서 적절히 대처하시거나 INP 신호 사용을 배제 해야 합니다.

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

EXAMPLE

C/C++

```
#include “ceSDK.h”
#include “ceSDKDef.h”

long nAxisNo = 1;     // 이송 대상 축을 선택합니다.

void OnSetSpeed ()
{
    / / 대상 축의 기본 속도를 설정합니다.
    cemCfgSpeedPattern_Set ( nAxisNo, cemSMODE_T, 10000, 50000, 50000 );
}


void OnMove ()
{
    /* cemCfgSpeedPattern_Set() 함수로 설정한 속도 설정을 그대로 유지 하면서
    단축 이송에 대해 속도 비율을 설정하여 이송 작업을 수행합니다. */
    cemSxSpeedRatio_Set ( nAxisNo, cemSMODE_KEEP, 50, 100, 100 );

    // 대상 축을 현재 위치에서 ‘10000’만큼 상대 이송합니다.
    if ( cemSxMoveStart( nAxisNo, 10000 ) == ceERR_NONE )
    {
            //블록 모드를 CE_FALSE 로 하면 UI 메시지 처리가 가능합니다.
            cemSxWaitDone ( nAxisNo, CE_FALSE );
    }

    // cemSxMoveStart(), cemSxWaitDone()을 아래 코드로 대체 할 수도 있습니다.
    // cemSxMove ( nAxisNo, 10000, CE_FALSE );


    // 대상 축의 현재 위치에 관계 없이 ‘0’ 위치로 절대 이송합니다.
    if ( cemSxMoveToStart ( nAxisNo, 0 ) == ceERR_NONE )
    {
            long nIsDone;      // 단축 이송 완료 정보.
            While ( 1 )
            {
```

```
                    // 단축 이송 작업 완료를 확인합니다.
                    cemSxIsDone ( nAxisNo, &nIsDone );
                    if ( nIsDone == CE_TRUE )   break;
            }
    }
}
```

Visual Basic

```
Dim nAxisNo As Long          ' 이송 대상 축을 선택합니다.

nAxisNo = 1


Private Sub OnSetSpeed ()
    / / 대상 축의 기본 속도를 설정합니다.
    Call cemCfgSpeedPattern_Set ( nAxisNo, cemSMODE_T, 10000, 50000, 50000 )
End Sub


Private Sub OnMove ()
    Dim nIsDone As Long      ' 단축 이송 완료 정보

    ' cemCfgSpeedPattern_Set() 함수로 설정한 속도 설정을 그대로 유지 하면서
    ' 단축 이송에 대해 속도 비율을 설정하여 이송 작업을 수행합니다.
    Call cemSxSpeedRatio_Set ( nAxisNo, cemSMODE_KEEP, 50, 100, 100 )

    ' 대상 축을 현재 위치에서 '10000'만큼 상대 이송합니다.
    If cemSxMoveStart ( nAxisNo, 10000 ) = ceERR_NONE Then
            ' 블록 모드를 CE_FALSE 로 하면 UI 메시지 처리가 가능합니다.
            Call cemSxWaitDone ( nAxisNo, CE_FALSE )
    End If

    ' cemSxMoveStart(), cemSxWaitDone()을 아래 코드로 대체 할 수도 있습니다.
    ' cemSxMove ( nAxisNo, 10000, CE_FALSE )


    ' 대상 축의 현재 위치에 관계 없이 '0' 위치로 절대 이송합니다.
    nIsDone = CE_FALSE
    If cemSxMoveToStart ( nAxisNo, 0 ) = ceERR_NONE Then

            ' 단축 이송 작업 완료를 확인합니다.
            While ( nIsDone = CE_FALSE  )
                    Call cemSxIsDone ( nAxisNo, nIsDone )
            Wend
    End If

End Sub
```

Delphi

```
Procedure OnSetSpeed ();
```

```
begin
    / / 대상 축의 기본 속도를 설정합니다.
    cemCfgSpeedPattern_Set ( cemX1, cemSMODE_T, 10000, 50000, 50000 );

end;


procedure OnMove ();
var
    nIsDone : LongInt;          // 단축 이송 완료 정보

begin
    { cemCfgSpeedPattern_Set() 함수로 설정한 속도 설정을 그대로 유지 하면서
    단축 이송에 대해 속도 비율을 설정하여 이송 작업을 수행합니다. }
    cemSxSpeedRatio_Set ( cemX1, cemSMODE_KEEP, 50, 100, 100 );

    // 대상 축을 현재 위치에서 '10000'만큼 상대 이송합니다.
    if cemSxMoveStart ( nAxisNo, 10000 ) = ceERR_NONE then
    begin
            ' 블록 모드를 CE_FALSE 로 하면 UI 메시지 처리가 가능합니다.
            cemSxWaitDone ( cemX1, CE_FALSE );
    end

    // cemSxMoveStart(), cemSxWaitDone()을 아래 코드로 대체 할 수도 있습니다.
    // cemSxMove ( cemX1, 10000, CE_FALSE );


    // 대상 축의 현재 위치에 관계 없이 '0' 위치로 절대 이송합니다.
    nIsDone := CE_FALSE;
    if cemSxMoveToStart ( cemX1, 0 ) = ceERR_NONE then
    begin
            // 단축 이송 작업 완료를 확인합니다.
            while nIsDone = CE_FALSE  do
            begin
                    cemSxIsDone ( cemX1, @nIsDone );
            end;
    end;

end;
```

NAME

cemSxMoveTo / cemSxMoveToStart

- 단축 절대좌표 이송

INFORMATION
1 Single Axis Control
! VC++ (6, 7, 8)/VB
BCB/Delphi
: Level 3
J 이송 함수
실제 이송이 진행되는 함수 이므로, 사전에 반드시 안전 을 확인 합니다.

SYNOPSIS

r VT_I4 cemSxMoveTo ([in] VT_I4 Axis, [in] VT_R8 Position, [in] VT_I4 IsBlocking)

r VT_I4 cemSxMoveToStart ([in] VT_I4 Axis, [in] VT_R8 Position)

DESCRIPTION

cemSxMoveTo /cemSxMoveToStart 함수는 하나의 축에 대하여 지정한 절대 좌표로의 이송을 수행합니다.

cemSxMoveTo 함수는 모션이 완료되기 전까지 반환되지 않으며, cemSxMoveToStart 함수는 모션을 시작시킨 후에 바로 반환됩니다.

PARAMETER

▶ ***Axis*** : 축 번호. 통합 축으로 관리되는 축 번호를 의미하며, 상수 값으로 0 (Zero Based) 이상, (최대 통합 축 개수 - 1) 이하의 값을 축 번호로 설정할 수 있습니다.

▶ ***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*** : cemSxMoveTo 함수의 인자이며, 완료될 때까지 기다리는 동안 윈도우 메시지를 블록(Blocking)할 것인지를 결정합니다. 단, 쓰레드 내에서 실행할 때는 이 값을 1(CE_TRUE)로 설정해 주어야 합니다.

Value	Meaning
0 (CE_FALSE)	윈도우 메시지를 블록(Blocking) 하지 않습니다. 따라서 해당 모션이 진행되는 동안 윈도우 이벤트를 처리합니다.
1 (CE_TRUE)	윈도우 메시지를 블록(Blocking) 합니다. 따라서 해당 모션이 완료되는 동안에는 윈도우 이벤트가 처리되지 않습니다.

RETURN VALUE

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

SEE ALSO

cemSxMove, cemSxMoveStart, cemSxVMoveStart

REFERENCE

□ 논리적 단위 거리는 cemCfgUnitDist_Set 함수에 의해 결정됩니다.

□ cemSxMoveToStart 함수를 사용하는 경우에는 cemSxIsDone 함수나 cemSxWaitDone 함수를 사용하여 모션의 완료를 확인할 수 있습니다.

□ cemSxMoveTo 함수를 사용하는 경우에는 내부적으로 루프를 수행하면서 모션이 완료되기를 기다리는데, 이때 블록 모드(Blocking Mode) 설정에 따라 윈도우 이벤트를 처리하는 방식이 달라집니다. 그러나 일반적으로 윈도우의 작업 쓰레드(Work Thread) 에서는 블록 모드를 사용하여, 함수 내부에서 지연 없이 쓰레드 내부의 작업에 집중할 수 있도록 설정하는 것이 바람직합니다.

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

□ 스텝 드라이버를 사용 중인 고객님들께서는 다음을 참조해 주십시오.
스텝 드라이버는 INP 출력이 없는 경우가 일반적인데, 고객님의 부주의나 잘못된 설정으로 INP 입력에 대한 설정이 Enable 로 되어 있을 경우 INP 입력이 스텝 드라이버를 통해 발생하지 않는 이유 때문에 모션 완료가 되지 않는 경우가 발생할 수 있습니다. 고객 여러분들께서는 스텝 드라이버 사용시에 이점을 주의해 주시기를 부탁 드립니다.

□ 서보 드라이버의 LSP, LSN 신호를 사용 중인 고객님께서는 다음을 참조해 주십시오.
서보드라이버의 입력 신호 중 하나인 EL(End of Limit) 신호는 저희 ㈜ 커미조아 모션 컨트롤러뿐만 아니라 서보드라이버에도 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 신호를 서보 드라이버 측에 전달하기 위한 용도로 사용됩니다.

그러나, 모션 소프트웨어에서 INP 설정이 되어 있는 경우 EL 신호가 검출 된 후에 일부 서보 드라이버에서는 진행 방향에서 정지 한 후 더 이상 움직이지 않는 상황이 발생하며, 이 상황에서 INP 신호가 출력되지 않아, 모션 이송이 완료되지 못하고, 명시적으로 STOP 명령을 통해 강제적으로 모션을 종료해야 되는 현상이 발생할 수 있습니다. 이 현상은 EL 모드를 통해 원점 복귀를 하는 상황에서도 발생할 수 있습니다.

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

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

EXAMPLE

```
//* cemSxMove / cemSxMoveStart 예제를 참고하여 주시기 바랍니다.
```

NAME

cemSxVMoveStart

- 단축 연속 속도 이송

INFORMATION
1 Single Axis Control
! VC++ (6, 7, 8)/VB
BCB/Delphi
: Level 3
J 이송 함수
실제 이송이 진행되는 함수이므로, 사전에 반드시 안전을 확인 합니다.

SYNOPSIS

r VT_I4 cemSxVMoveStart ([in] VT_I4 Axis, [in] VT_I4 Direction)

DESCRIPTION

cemSxVMoveStart 함수는 작업 속도까지 가속한 후에 작업 속도를 유지하며 정지 함수가 호출될 때까지 지정한 방향으로의 모션을 계속 수행합니다.

이 함수는 모션을 시작시킨 후에 바로 반환됩니다.

PARAMETER

▶ ***Axis*** : 축 번호. 통합 축으로 관리되는 축 번호를 의미하며, 상수 값으로 0 (Zero Based) 이상, (최대 통합 축 개수 - 1) 이하의 값을 축 번호로 설정할 수 있습니다.

▶ ***Direction*** : 이 매개변수를 통하여 모션의 방향을 설정합니다.

Value	Meaning
0 (cemDIR_N)	(-) 방향 => Negative direction.
1 (cemDIR_P)	(+) 방향 => Positive direction.

RETURN VALUE

Value	Meaning
음수	수행 실패. 자세한 내용은 '에러처리' 편을 참고합니다.
0 (ceERR_NONE)	수행 성공.

EXAMPLE

C/C++

```
#include “ceSDK.h”
#include “ceSDKDef.h”

/* 연속 속도 이송 함수를 이용하여 조그 이송을 합니다.
조그 버튼이 눌려진 상태이면 이송하다가, 조그 버튼을 놓으면 이송을 정지 합니다. */

/************************************************************************************************
* (+) Move 버튼의 MouseDown 이벤트 발생 시 (+) 방향으로 이송을 시작합니다.
************************************************************************************************/
void OnJogBtnDownPos ()
{
    cemSxVMoveStart ( cemX1, cemDIR_P );
}

/************************************************************************************************
* (-) Move 버튼의 MouseDown 이벤트 발생 시 (-) 방향으로 이송을 시작합니다.
************************************************************************************************/
void OnJogBtnDownNeg ()
{
    cemSxVMoveStart ( cemX1, cemDIR_N );
}

/************************************************************************************************
* (+)/(-) Move 버튼의 MouseUp 이벤트 발생 시 이송을 정지합니다.
************************************************************************************************/
void OnJogBtnUp ()
{
    // 이송 축을 정해진 속도 설정으로 감속 후 정지합니다.
    cemSxStop (     cemX1,          // 대상 축 선택.
                    CE_FALSE,       // 동작 완료 시까지 함수 반환 여부 설정.
                    CE_FALSE        // 메시지 블록 여부 설정
                    );
    // cemSxStopEmg () 함수 호출 시 감속 없이 급 정지 합니다.
}
```

Visual Basic

```
‘ 연속 속도 이송 함수를 이용하여 조그 이송을 합니다.
‘ 조그 버튼이 눌려진 상태이면 이송하다가, 조그 버튼을 놓으면 이송을 정지 합니다.

‘ ============================================================
‘ (+) Move 버튼의 MouseDown 이벤트 발생 시 (+) 방향으로 이송을 시작합니다.
‘ ============================================================
Private Sub OnJogBtnPos_MouseDown ()
    Call cemSxVMoveStart ( cemX1, cemDIR_P )
End Sub

‘ ============================================================
```

```
' (-) Move 버튼의 MouseDown 이벤트 발생 시 (-) 방향으로 이송을 시작합니다.
' ============================================================
Private Sub OnJogBtnNeg_MouseDown ()
    Call cemSxVMoveStart ( cemX1, cemDIR_N )
End Sub

' ============================================================
' (+) Move 버튼의 MouseUp 이벤트 발생 시 이송을 정지합니다.
' ============================================================
Private Sub OnJogBtnPos_MouseUp ()

    ' 이송 축을 정해진 속도 설정으로 감속 후 정지합니다.
    Call cemSxStop ( cemX1, CE_FALSE, CE_FALSE )
    ' cemSxStopEmg () 함수 호출 시 감속 없이 급 정지 합니다.
End Sub

' ============================================================
' (-) Move 버튼의 MouseUp 이벤트 발생 시 이송을 정지합니다.
' ============================================================
Private Sub OnJogBtnNeg_MouseUp ()
    Call cemSxStop ( cemX1, CE_FALSE, CE_FALSE )
End Sub
```

Delphi

```
{ 연속 속도 이송 함수를 이용하여 조그 이송을 합니다.
조그 버튼이 눌려진 상태이면 이송하다가, 조그 버튼을 놓으면 이송을 정지 합니다. }

// ***********************************************************************************************
// (+) Move 버튼의 MouseDown 이벤트 발생 시 (+) 방향으로 이송을 시작합니다.
// ***********************************************************************************************
procedure OnJogBtnPos_MouseDown ();
begin
    cemSxVMoveStart ( cemX1, cemDIR_P );
end;

// ***********************************************************************************************
// (-) Move 버튼의 MouseDown 이벤트 발생 시 (-) 방향으로 이송을 시작합니다.
// ***********************************************************************************************
procedure OnJogBtnNeg_MouseDown ();
begin
    cemSxVMoveStart ( cemX1, cemDIR_N );
end;

// ***********************************************************************************************
// (+)/(-) Move 버튼의 MouseUp 이벤트 발생 시 이송을 정지합니다.
// ***********************************************************************************************
procedure OnJogBtn_MouseUp ();
begin
    // 이송 축을 정해진 속도 설정으로 감속 후 정지합니다.
    cemSxStop ( cemX1, CE_FALSE, CE_FALSE );
    // cemSxStopEmg() 함수 호출 시 감속 없이 급 정지 합니다.
end;
```

NAME

cemSxMoveStart2V

- 단축 모션 상대좌표 2 단계 속도 이송

INFORMATION
1 Single Axis Control
! VC++ (6, 7, 8)/VB
BCB/Delphi
: Level 3
J 이송 함수
실제 이송이 진행되는 함수이므로, 사전에 반드시 안전을 확인 합니다.

SYNOPSIS

r VT_I4 cemSxMoveStart2V ([in] VT_I4 Axis, [in] VT_R8 Distance, [in] VT_R8 Vel2)

DESCRIPTION

cemSxMoveStart2V 함수는 하나의 축에 대하여 현재 위치에서 지정한 거리(상대 좌표)만큼 2 단계 속도로 이송을 수행합니다.
이송을 시작하면 cemCfgSpeedPattern_Set 함수를 통해 설정된 작업 속도까지 가속을 한 후 Vel2 로 설정한 2 차 작업속도까지 가속을 한 뒤 감속하여 이송을 완료합니다.

이 함수는 모션을 시작시킨 후에 바로 반환됩니다.

PARAMETER

▶ ***Axis*** : 축 번호. 통합 축으로 관리되는 축 번호를 의미하며, 상수 값으로 0 (Zero Based) 이상, (최대 통합 축 개수 - 1) 이하의 값을 축 번호로 설정할 수 있습니다.

▶ ***Distance*** : 이송할 거리를 지정합니다. 이 값은 현재의 위치에 대한 상대 좌표이며, 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.

▶ ***Vel2*** : 이 매개변수를 통하여 2 차 작업 속도를 설정합니다.

RETURN VALUE

Value	Meaning
음수	수행 실패. 자세한 내용은 ‘에러처리’ 편을 참고합니다.
0 (ceERR_NONE)	수행 성공.

REFERENCE

□ cemSxMoveStart2V / cemSxMoveToStart2V 의 2 차 속도 패턴

단축 2 단계 속도 이송은 Accel -> Vwork -> Accel2 -> Vwork2 -> Decel 의 속도 패턴으로 이송합니다.

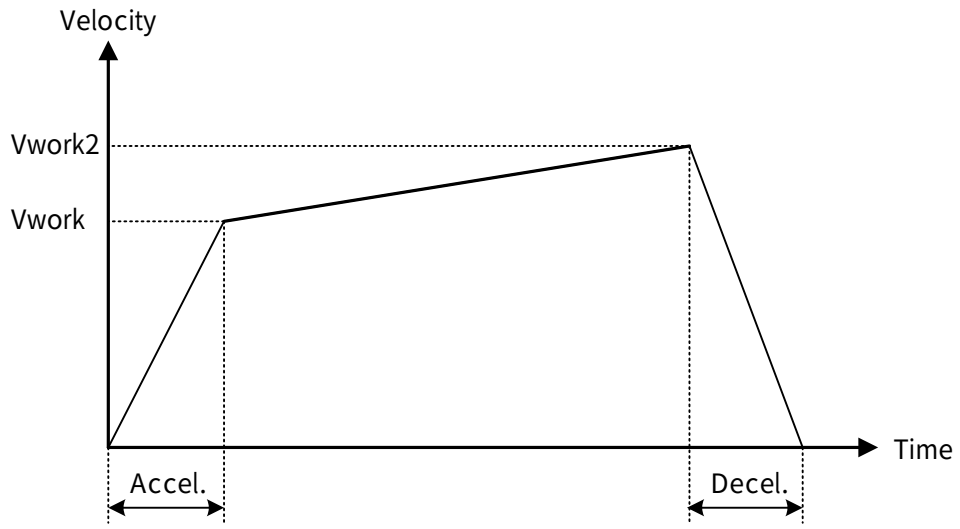


그림 7-1 단축 2 단계 이송 속도 패턴

즉, 정해진 가속도를 통해 작업 속도 까지 가속을 한 후, 정해진 감속도로 감속을 시작하기 전까지 Vwork2 작업 속도 까지 가속 한 후 감속하여 지정한 이송 거리만큼 이송 작업을 수행합니다.

EXAMPLE

C/C++

```
#include “ceSDK.h”
#include “ceSDKDef.h”

long nAxisNo = 1;     // 이송 대상 축을 선택합니다.

void OnSetSpeed ()
{
    / / 대상 축의 기본 속도를 설정합니다.
    cemCfgSpeedPattern_Set ( nAxisNo, cemSMODE_T, 1000, 5000, 5000 );
}


void OnMove ()
{
    long nRetVal;

    /* 대상 축의 기본 작업 속도까지 가속을 한 후 설정한 2 차 작업 속도까지 가속을 한 후
    감속하여 현재 위치에서 ‘10000’만큼 상대 위치까지 이송합니다. */
    nRetVal = cemSxMoveStart2V (    nAxisNo,        // 대상 축 설정.
                                    10000,          // 이송할 거리 설정.
                                    20000           // 2 차 작업 속도
```

```
    )

    if ( nRetVal == ceERR_NONE )
    {
            //블록 모드를 CE_FALSE 로 하면 UI 메시지 처리가 가능합니다.
            cemSxWaitDone ( nAxisNo, CE_FALSE );
    }

    /* 대상 축의 기본 작업 속도까지 가속을 한 후 설정한 2 차 작업 속도까지 가속을 한 후
    감속하여 현재 위치에 관계 없이 '0' 위치로 절대 이송합니다. */

    if ( cemSxMoveToStart2V ( nAxisNo, 0, 20000 ) == ceERR_NONE )
    {
            long nIsDone;      // 단축 이송 완료 정보.
            While ( 1 )
            {
                    // 단축 이송 작업 완료를 확인합니다.
                    cemSxIsDone ( nAxisNo, &nIsDone );
                    if ( nIsDone == CE_TRUE )   break;
            }
    }
}
```

```
Visual Basic

Dim nAxisNo As Long          ' 이송 대상 축을 선택합니다.
nAxisNo = 1

Private Sub OnSetSpeed ()
    ' 대상 축의 기본 속도를 설정합니다.
    cemCfgSpeedPattern_Set ( nAxisNo, cemSMODE_T, 1000, 5000, 5000 )
End Sub


Private Sub OnMove ()

    Dim nIsDone As Long       ' 단축 이송 완료 정보.

    ' 대상 축의 기본 작업 속도까지 가속을 한 후 설정한 2 차 작업 속도까지 가속을 한 후
    ' 감속하여 현재 위치에서 '10000'만큼 상대 위치까지 이송합니다.

    If cemSxMoveStart2V ( nAxisNo, 10000, 20000 ) = ceERR_NONE Then
            ' 블록 모드를 CE_FALSE 로 하면 UI 메시지 처리가 가능합니다.
            Call cemSxWaitDone ( nAxisNo, CE_FALSE )
    End If

    ' 대상 축의 기본 작업 속도까지 가속을 한 후 설정한 2 차 작업 속도까지 가속을 한 후
    ' 감속하여 현재 위치에 관계 없이 '0' 위치로 절대 이송합니다.

    If cemSxMoveToStart2V ( nAxisNo, 0, 20000 ) = ceERR_NONE Then

            nIsDone = CE_FALSE

            ' 단축 2 단계 이송 작업 완료를 확인합니다.
```

```
        While ( nIsDone = CE_FALSE )
            Call cemSxIsDone ( nAxisNo, nIsDone )
        Wend
    End If

End Sub
```

Delphi

```
procedure OnSetSpeed ();
begin
    // 대상 축의 기본 속도를 설정합니다.
    cemCfgSpeedPattern_Set ( cemX1, cemSMODE_T, 1000, 5000, 5000 );
end;


procedure OnMove ();
var
    nIsDone : LongInt;          // 단축 이송 완료 정보.

begin
    { 대상 축의 기본 작업 속도까지 가속을 한 후 설정한 2 차 작업 속도까지 가속을 한 후
    감속하여 현재 위치에서 '10000'만큼 상대 위치까지 이송합니다. }

    if cemSxMoveStart2V ( cemX1, 10000, 20000 ) = ceERR_NONE then
    begin
        // 블록 모드를 CE_FALSE 로 하면 UI 메시지 처리가 가능합니다.
        cemSxWaitDone ( cemX1, CE_FALSE );
    end;

    { 대상 축의 기본 작업 속도까지 가속을 한 후 설정한 2 차 작업 속도까지 가속을 한 후
    감속하여 현재 위치에 관계 없이 '0' 위치로 절대 이송합니다. }

    if cemSxMoveToStart2V ( cemX1, 0, 20000 ) = ceERR_NONE then
    being
        nIsDone := CE_FALSE;

        // 단축 2 단계 이송 작업 완료를 확인합니다.
        while nIsDone = CE_FALSE do
        begin
            cemSxIsDone (cemX1, @nIsDone );
        end;
    end;

end;
```

NAME

cemSxMoveToStart2V

- 단축 모션 절대좌표 2 단계 속도 이송

INFORMATION
1 Single Axis Control
! VC++ (6, 7, 8)/VB
BCB/Delphi
: Level 3
J 이송 함수
실제 이송이 진행되는 함수이므로, 사전에 반드시 안전을 확인 합니다.

SYNOPSIS

r VT_I4 cemSxMoveToStart2V ([in] VT_I4 Axis, [in] VT_R8 Position, [in] VT_R8 Vel2)

DESCRIPTION

cemSxMoveToStart2V 함수는 하나의 축에 대하여 현재 좌표에서 지정한 좌표(절대 좌표)만큼 2 단계 속도로 이송을 수행합니다.

이송을 시작하면 cemCfgSpeedPattern_Set 함수를 통해 설정된 작업 속도까지 가속을 한 후 Vel2 로 설정한 2 차 작업속도까지 가속을 한 후 감속하여 이송을 완료합니다.

이 함수는 모션을 시작시킨 후에 바로 반환됩니다.

PARAMETER

▶ ***Axis*** : 축 번호. 통합 축으로 관리되는 축 번호를 의미하며, 상수 값으로 0 (Zero Based) 이상, (최대 통합 축 개수 - 1) 이하의 값을 축 번호로 설정할 수 있습니다.

▶ ***Position*** : 이송할 절대 좌표 값을 지정하며, 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.

▶ ***Vel2*** : 이 매개변수를 통하여 2 차 작업 속도를 설정합니다.

RETURN VALUE

Value	Meaning
음수	수행 실패. 자세한 내용은 ‘에러처리’ 편을 참고합니다.
0 (ceERR_NONE)	수행 성공.

EXAMPLE

```
//* cemSxMoveStart2V 예제를 참고하여 주시기 바랍니다.
```

NAME

cemSxStop / cemSxStopEmg

- 단축 이송 정지

- 단축 비상 정지

INFORMATION
1 Single Axis Control
! VC++ (6, 7, 8)/VB
BCB/Delphi
: Level 3
J 정지 함수
고속 이송시에 비상 정지를 주의 하십시오. 기구물의 손상이나 안전사고에 원인이 될 수 있습니다.

SYNOPSIS

r VT_I4 cemSxStop ([in] VT_I4 Axis, [in] VT_I4 IsWaitComplete, [in] VT_I4 IsBlocking)

r VT_I4 cemSxStopEmg ([in] VT_I4 Axis)

DESCRIPTION

cemSxStop /cemSxStopEmg 함수는 지정한 축에 대한 모션을 정지합니다.

cemSxStop 함수는 정지 시에 감속 후 정지를 수행하며, cemSxStopEmg 함수는 감속 없이 즉시 정지를 수행합니다.

PARAMETER

▶ *Axis* : 축 번호. 통합 축으로 관리되는 축 번호를 의미하며, 상수 값으로 0 (Zero Based) 이상, (최대 통합 축 개수 - 1) 이하의 값을 축 번호로 설정할 수 있습니다.

▶ *IsWaitComplete* : cemSxStop 함수의 인자이며, 이 동작이 완료될 때까지 함수를 반환할 것인지를 결정합니다.

Value	Meaning
0 (CE_FALSE)	대상 축의 정지가 완료될 때까지 대기하지 않고 함수를 벗어납니다.
1 (CE_TRUE)	대상 축의 정지가 완료될 때까지 대기합니다.

▶ *IsBlocking* : cemSxStop 함수의 인자이며, 완료될 때까지 기다리는 동안 윈도우 메시지를 블록(Blocking)할 것인지를 결정합니다.

Value	Meaning
0 (CE_FALSE)	윈도우 메시지를 블록(Blocking) 하지 않습니다. 따라서 해당 모션이 진행되는 동안 윈도우 이벤트를 처리합니다.
1 (CE_TRUE)	윈도우 메시지를 블록(Blocking) 합니다. 따라서 해당 모션이 완료되는 동안에는 윈도우 이벤트가 처리되지 않습니다.

RETURN VALUE

Value	Meaning
음수	수행 실패. 자세한 내용은 '에러처리' 편을 참고합니다.
0 (ceERR_NONE)	수행 성공.

EXAMPLE

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

NAME

cemSxIsDone

- 단축 모션 완료 확인

INFORMATION	
1	Single Axis Control
!	VC++ (6, 7, 8)/VB
	BCB/Delphi
:	Level 3
J	위험 요소 없음

SYNOPSIS

r VT_I4 cemSxIsDone ([in] VT_I4 Axis, [out] VT_PI4 IsDone)

DESCRIPTION

cemSxIsDone 함수는 단일 축에 대하여 모션 완료를 확인합니다.

PARAMETER

▶ *Axis* : 축 번호. 통합 축으로 관리되는 축 번호를 의미하며, 상수 값으로 0 (Zero Based) 이상, (최대 통합 축 개수 - 1) 이하의 값을 축 번호로 설정할 수 있습니다.

▶ *IsDone* : 이 매개변수를 통해 모션 작업이 완료되었는지를 판단할 수 있습니다.

Value	Meaning
0 (CE_FALSE)	모션작업이 완료되지 않음.
1 (CE_TRUE)	모션작업이 완료 됨.

RETURN VALUE

Value	Meaning
음수	수행 실패. 자세한 내용은 '에러처리' 편을 참고합니다.
0 (ceERR_NONE)	수행 성공.

SEE ALSO

cemSxWaitDone

EXAMPLE

```
//* cemSxMove / cemSxMoveStart 예제를 참고하여 주시기 바랍니다.
```

NAME

cemSxWaitDone

- 단축 모션 완료 대기

INFORMATION	
1	Single Axis Control
!	VC++ (6, 7, 8)/VB
	BCB/Delphi
:	Level 3
J	위험 요소 없음

SYNOPSIS

r VT_I4 cemSxWaitDone ([in] VT_I4 Axis, [in] VT_I4 IsBlocking)

DESCRIPTION

cemSxWaitDone() 함수는 단일 축에 대하여 이송 작업이 완료될 때까지 기다립니다.

PARAMETER

▶ *Axis* : 축 번호. 통합 축으로 관리되는 축 번호를 의미하며, 상수 값으로 0 (Zero Based) 이상, 최대 통합 축 개수 - 1 이하의 값을 축 번호로 설정할 수 있습니다.

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

Value	Meaning
0 (CE_FALSE)	윈도우 메시지를 블록(Blocking) 하지 않습니다. 따라서 해당 모션이 진행되는 동안 윈도우 이벤트를 처리합니다.
1 (CE_TRUE)	윈도우 메시지를 블록(Blocking) 합니다. 따라서 해당 모션이 완료되는 동안에는 윈도우 이벤트가 처리되지 않습니다.

RETURN VALUE

Value	Meaning
음수	수행 실패. 자세한 내용은 '에러처리' 편을 참고합니다.
0 (ceERR_NONE)	수행 성공.

SEE ALSO

cemSxIsDone

EXAMPLE

```
//* cemSxMove / cemSxMoveStart 예제를 참고하여 주시기 바랍니다.
```

NAME

cemSxTargetPos_Get

- 마지막 이송 (상대 or 절대 좌표) 위치 반환

INFORMATION	
1	Single Axis Control
!	VC++ (6, 7, 8)/VB
	BCB/Delphi
:	Level 3
J	위험 요소 없음

SYNOPSIS

r VT_I4 cemSxTargetPos_Get ([in] VT_I4 Channel, [out] VT_PR8 Position)

DESCRIPTION

대상 축에 대하여 마지막으로 수행한 단축 이송 명령의 위치(상대 혹은 절대 좌표)를 반환합니다.

cemSxTargetPos_Get 함수를 통해 마지막 이송 위치 반환 대상 함수는 다음과 같습니다.
cemSxMove, cemSxMoveStart, cemSxMoveTo, cemSxMoveToStart, cemSxMoveStart2V, cemSxMoveToStart2V

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 축이 마지막으로 이송한 (상대 혹은 절대 좌표) 위치를 반환합니다.

RETURN VALUE

Value	Meaning
음수	수행 실패. 자세한 내용은 '에러처리' 편을 참고합니다.
0 (ceERR_NONE)	수행 성공.

EXAMPLE

C/C++

```
#include “ceSDK.h”
#include “ceSDKDef.h”

void OnGetSxTargetPos ()
{
    double fTargetPos;          // 마지막으로 수행한 단축 이송 명령의 위치 정보.
    double fGetPos;             // 현재 위치 정보.
```

```
    if ( cemSxTargetPos_Get ( cemX1, &fTargetPos ) == ceERR_NONE )
    {
            cemStPosition_Get ( cemX1, cemCNT_COMM, &fGetPos );
            if ( fTargetPos == fGetPos )
            {
                    // 단축 이송 명령에 대한 이송 완료 확인.
            }
    }
}
```

Visual Basic

```
Private Sub OnGetSxTargetPos ()

    Dim fTargetPos As Double        ' 마지막으로 수행한 단축 이송 명령의 위치 정보.
    Dim fGetPos As Double           ' 현재 위치 정보.

    If cemSxTargetPos_Get ( cemX1, fTargetPos ) = ceERR_NONE Then

            Call cemStPosition_Get ( cemX1, cemCNT_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;            ' 마지막으로 수행한 단축 이송 명령의 위치 정보.
    fGetPos : Double;               ' 현재 위치 정보.

begin
    if cemSxTargetPos_Get ( cemX1, @fTargetPos ) = ceERR_NONE then
    begin
            cemStPosition_Get ( cemX1, cemCNT_COMM, @fGetPos );

            if fTargetPos = fGetPos then
            begin
                    // 단축 이송 명령에 대한 이송 완료 확인.
            end;
    end;

end;
```

NAME

cemSxOptIniSpeed_Set /

cemSxOptIniSpeed_Get

- 단축 모션 초기 속도 설정 및 반환

INFORMATION	
1	Single Axis Control
!	VC++ (6, 7, 8)/VB
	BCB/Delphi
:	Level 3
J	위험 요소 없음

SYNOPSIS

r VT_I4 cemSxOptIniSpeed_Set ([in] VT_I4 Axis, [in] VT_R8 IniSpeed)

r VT_I4 cemSxOptIniSpeed_Get ([in] VT_I4 Axis, [out] VT_PR8 IniSpeed)

DESCRIPTION

cemSxOptIniSpeed_Set 함수는 모션의 초기 속도를 설정합니다.

cemSxOptIniSpeed_Get 함수는 모션의 초기 속도 설정 상태를 반환합니다.

PARAMETER

▶ *Axis* : 축 번호. 통합 축으로 관리되는 축 번호를 의미하며, 상수 값으로 0 (Zero Based) 이상, (최대 통합 축 개수 - 1) 이하의 값을 축 번호로 설정할 수 있습니다.

▶ *IniSpeed* : 이 매개변수를 통하여 모션의 초기 속도를 설정 혹은 반환합니다.

RETURN VALUE

Value	Meaning
음수	수행 실패. 자세한 내용은 '에러처리' 편을 참고합니다.
0 (ceERR_NONE)	수행 성공.

EXAMPLE

C/C++

```
#include “ceSDK.h”
#include “ceSDKDef.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 ( cemSxOptIniSpeed_Get ( nAxisNo, &fIniSpeed ) == ceERR_NONE )
```

```
    {
        if ( fIniSpeed != 100 )
        {
            // 해당 축의 초기 속도 값을 '100'으로 설정합니다.
            cemSxOptIniSpeed_Set ( 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 cemSxOptIniSpeed_Get ( nAxisNo, fIniSpeed ) = ceERR_NONE Then

        If fIniSpeed <> 100 Then
            ' 해당 축의 초기 속도 값을 '100'으로 설정합니다.
            Call cemSxOptIniSpeed_Set ( nAxisNo, 100 )
        End If
    End If

End Sub
```

Delphi

```
procedure OnSetInitSpeed ();
var
    fIniSpeed : Double          // 초기 속도 정보.

begin
    // 해당 축의 초기 속도 값을 확인 후, 초기 속도를 '100'으로 설정합니다.
    if cemSxOptIniSpeed_Get ( cemX1, @fIniSpeed ) = ceERR_NONE then
    begin
        if fIniSpeed <> 100 then
        begin
            // 해당 축의 초기 속도 값을 '100'으로 설정합니다.
            cemSxOptIniSpeed_Set ( nAxisNo, 100 );
        end;
    end;

end;
```

NAME

cemSxCorrection_Set / cemSxCorrection_Get

- 백래쉬 / 슬립 보정 설정 및 반환

INFORMATION	
1	Single Axis Control
!	VC++ (6, 7, 8)/VB
	BCB/Delphi
:	Level 3
J	위험 요소 없음

SYNOPSIS

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

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

DESCRIPTION

cemSxCorrection_Set 함수는 백래쉬(Backlash) 또는 슬립(Slip)에 대한 보정 환경을 설정합니다. 구조적으로 백래쉬나 슬립 현상이 심하게 일어나는 경우에는 이에 대한 보정이 필요할 수 있습니다. 백래쉬는 일반적으로 모터의 구동 방향이 바뀔 때 발생됩니다. 따라서 ㈜커미조아 모션컨트롤러의 백래쉬 보정은 모터의 제어 방향이 바뀔 때에만 적용됩니다. 백래쉬 보정을 활성화하면 모션컨트롤러에서 지령되는 이송 명령의 방향이 이전의 이송 방향과 다른 경우에 자동적으로 백래쉬 보정 설정에 따라 보정 펄스가 출력된 후에 지정된 이송을 수행합니다. 이 설정은 단축 구동뿐 아니라, 보간 구동에서도 적용됩니다.

슬립은 일반적으로 정지 후 재 기동 시에 발생합니다. 따라서 슬립 보정은 이송 방향에 상관없이 기동 시에 보정 펄스가 출력됩니다. 슬립 보정을 활성화한 후에 이송 명령이 하달되면 모션컨트롤러는 슬립 보정 설정에 따라 보정 펄스가 출력된 후에 지정된 이송을 수행합니다.

cemSxCorrection_Get 함수는 백래쉬 또는 슬립 보정 설정 상태를 반환합니다.

PARAMETER

▶ ***Axis*** : 축 번호. 통합 축으로 관리되는 축 번호를 의미하며, 상수 값으로 0 (Zero Based) 이상, (최대 통합 축 개수 – 1) 이하의 값을 축 번호로 설정할 수 있습니다.

▶ ***CorrMode*** : 보정 모드를 설정 혹은 반환합니다. 보정 모드의 의미는 다음과 같습니다.

BIT No.	Meaning
0 (cemCORR_DIS)	보정기능을 비활성화합니다.
1 (cemCORR_BACK)	보정모드를 백래쉬 보정모드로 설정합니다.
2 (cemCORR_SLIP)	보정모드를 슬립 보정모드로 설정합니다.

▶ *CorrAmount* : 보정 펄스의 수를 설정 혹은 반환합니다. 단, 이 값은 논리적 단위 거리로 설정해야 합니다. 따라서 "Unit distance"(Du)를 '1'이 아닌 값으로 설정한 경우에 실제 출력되는 보정 펄스 수(Nc)는 다음과 같습니다.

Nc = CorrAmount * Du

그리고 보정펄스 수(Nc)는 0 ~ 4095 의 값이어야 합니다.

▶ *CorrVel* : 보정펄스의 출력 주파수를 설정 혹은 반환합니다. 단, 이 값은 논리적 속도 단위로 설정해야 합니다. 따라서 "Unit speed"(Vu)를 '1'이 아닌 값으로 설정한 경우에 실제 출력 주파수(Fc)는 다음과 같습니다.

Fc (PPS) = CorrVel * Vu

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

▶ *CntrMask* : 보정펄스가 출력되는 동안에 각 카운터의 동작 여부를 아래의 표와 같이 각 비트 별로 설정 혹은 반환합니다. 예를 들어 이 값의 BIT0 을 1 로 하면 보정펄스가 출력되는 동안에도 Counter 의 값은 증가 또는 감소합니다. 그리고 BIT0 을 0 으로 하면 보정펄스가 출력되는 동안에는 Counter 의 값이 변화하지 않습니다.

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

RETURN VALUE

Value	Meaning
음수	수행 실패. 자세한 내용은 '에러처리' 편을 참고합니다.
0 (ceERR_NONE)	수행 성공.

EXAMPLE

C/C++

```
#include “ceSDK.h”
#include “ceSDKDef.h”

long nAxisNo = 1;     // 백래쉬 보정 설정 할 대상 축을 설정합니다.

void OnSetSxCorrection ()
{
    // 해당 축에 대해 백래쉬 보정 모드를 설정 합니다.
    // ( 보정 펄스 수 : 1000, 보정 펄스 출력 속도 : 1000 PPS )
    cemSxCorrection_Set(    nAxisNo,                // 대상 축 설정.
                            cemCORR_BACK,           // 보정 모드 선택.
                            1000,                   // 보정 펄스 수 설정.
                            1000,                   // 보정 펄스 출력 속도 설정
                            0x1                     // 대상 카운터 설정.
                            );
}


void OnMove ()
{
    // 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.
    cemSxMove ( nAxisNo, 10000, CE_FALSE );

    //이송 방향이 이전과 동일하므로 아래에서는 백래쉬 보정을 하지 않습니다.
    cemSxMove ( nAxisNo, 10000, CE_FALSE );

    // 이송 방향이 전환되므로 백래쉬 보정을 수행합니다.
    // 1000PPS 의 속도로 (-)1000 펄스를 출력 한 후에 지정한 SxMove()가 수행됩니다.
    cemSxMove ( nAxisNo, -10000, CE_FALSE );
}
```

Visual Basic

```
Dim nAxisNo As Long          ‘ 백래쉬 보정 설정 할 대상 축을 설정합니다.
nAxisNo = 1

Private Sub OnSetSxCorrection ()

    ‘ 해당 축에 대해 백래쉬 보정 모드를 설정 합니다.
    ‘ ( 보정 펄스 수 : 1000, 보정 펄스 출력 속도 : 1000 PPS )
    Call cemSxCorrection_Set( nAxisNo, cemCORR_BACK, 1000, 1000, &H1 )

End Sub


Private Sub OnMove ()
```

```
    ‘ 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.
    Call cemSxMove ( nAxisNo, 10000, CE_FALSE )

    ‘ 이송 방향이 이전과 동일하므로 아래에서는 백래쉬 보정을 하지 않습니다.
    Call cemSxMove ( nAxisNo, 10000, CE_FALSE )

    ‘ 이송 방향이 전환되므로 백래쉬 보정을 수행합니다.
    ‘ 1000PPS 의 속도로 (-)1000 펄스를 출력 한 후에 지정한 SxMove()가 수행됩니다.
    Call cemSxMove ( nAxisNo, -10000, CE_FALSE )

End Sub
```

Delphi

```
procedure OnSetSxCorrection ();
begin

    { 해당 축에 대해 백래쉬 보정 모드를 설정 합니다.
    ( 보정 펄스 수 : 1000, 보정 펄스 출력 속도 : 1000 PPS ) }
    cemSxCorrection_Set( cemX1, cemCORR_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. }
    cemSxMove ( cemX1, 10000, CE_FALSE );

    // 이송 방향이 이전과 동일하므로 아래에서는 백래쉬 보정을 하지 않습니다.
    cemSxMove ( cemX1, 10000, CE_FALSE );

    { 이송 방향이 전환되므로 백래쉬 보정을 수행합니다.
    1000PPS 의 속도로 (-)1000 펄스를 출력 한 후에 지정한 SxMove()가 수행됩니다. }
    cemSxMove ( cemX1, -10000, CE_FALSE )

end;
```

7.2 기본 보간 제어 (Interpolation Motion)

이 단원에서는 보간(Interpolation) 모션 제어에 관련된 함수들을 소개합니다. 보간 모션 제어란 두 축 이상의 축이 연동되어 직선 보간(Linear Interpolation), 원호 보간(Circular Interpolation) 등의 모션을 수행하는 것을 의미합니다.

보간 제어는 사용자가 원하는 경로를 추종하기 위해 보간 이송에 관련된 각 축의 속도가 자동으로 조절되면서 이송을 수행합니다.

㈜ 커미조아의 모션 모듈은 해당 원격 노드 내의 모든 모션 제어 축을 대상으로 직선 보간 제어 기능을 제공합니다. 당사의 모션 모듈이 제공하는 보간 제어의 사양 표는 다음과 같습니다.

보간 제어 방식	축 구성 제한 사항
직선 보간 제어	단일 원격 노드 내의 모든 모션 제어 축.
원호 보간 제어	단일 모션 모듈 내의 모션 제어 축.

㈜커미조아의 모션 모듈은 해당 원격 노드 안에서 축 구성에 제약이 없는 직선보간, 2 축 원호 보간 이송 작업을 자동으로 수행하는 기능을 제공합니다. 직선 보간과 원호 보간은 “기본 보간 제어” 기능으로 분류됩니다.

7.2.1 직선 보간

ceSDK 에서는 동일 원격 노드에 속하는 모션 모듈에 대하여 축들간의 “직선 보간” 구동에 아무런 제약 없이 사용하실 수 있습니다. 예를 들어 동일 원격 노드의 1 번 모션 모듈과 7 번 모션 모듈의 축간에 “직선 보간” 구동이 가능합니다.

7.2.2 원호 보간

ceSDK 에서는 동일 모션 모듈에 대하여 “원호 보간” 구동 기능을 사용하실 수 있습니다. 즉, 서로 다른 모션 모듈에 대해서 “원호 보간” 구동은 허용되지 않습니다.

7.2.3 함수 요약

"기본 보간 제어" 에 관련된 함수는 다음과 같습니다.

Summary of Functions
r VT_I4 cemIxMapAxes ([in] VT_I4 MapIndex, [in] VT_I4 NodeID, [in] VT_I4 MapMask1, [in] VT_I4 MapMask2) 해당 원격 노드의 모션 모듈에 대해 보간 대상 축 그룹(Group)을 설정합니다.
r VT_I4 cemIxUnMap ([in] VT_I4 MapIndex) 설정된 보간 대상 축 그룹(Group)을 해제합니다.
r VT_I4 cemIxSpeedPattern_Set ([in] VT_I4 MapIndex, [in] VT_I4 IsVectorSpeed, [in] VT_I4 SpeedMode, [in] VT_R8 Vel, [in] VT_R8 Acc, [in] VT_R8 Dec) 보간 이송 속도 환경을 설정합니다. 보간 이송 속도 환경은 마스터 속도 모드와 백터 속도 모드를 설정할 수 있습니다.
r VT_I4 cemIxSpeedPattern_Get ([in] VT_I4 MapIndex, [out] VT_PI4 IsVectorSpeed, [out] VT_PI4 SpeedMode, [out] VT_PR8 Vel, [out] VT_PR8 Acc, [out] VT_PR8 Dec) 설정된 보간 이송 속도 환경을 반환합니다.
r VT_I4 cemIxVelCorrMode_Set ([in] VT_I4 MapIndex, [in] VT_I4 VelCorrOpt1, [in] VT_I4 VelCorrOpt2) 보간 속도 보정 모드를 설정합니다.
r VT_I4 cemIxVelCorrMode_Get ([in] VT_I4 MapIndex, [out] VT_PI4 VelCorrOpt1, [out] VT_PI4 VelCorrOpt2) 설정된 보간 속도 보정 모드를 반환합니다.
r VT_I4 cemIxLine ([in] VT_I4 MapIndex, [in] VT_PR8 DistList, [in] VT_I4 IsBlocking) 보간 제어에 있어 상대 좌표 직선 보간 이송을 수행합니다. 이 함수는 모션이 완료되기 전까지 반환되지 않습니다.
r VT_I4 cemIxLineStart ([in] VT_I4 MapIndex, [in] VT_PR8 DistList) 보간 제어에 있어 상대 좌표 직선 보간 이송을 수행합니다. 이 함수는 모션 시작 후 바로 반환됩니다.
r VT_I4 cemIxLineTo ([in] VT_I4 MapIndex, [in] VT_PR8 PosList, [in] VT_I4 IsBlocking) 보간 제어에 있어 절대 좌표 직선 보간 이송을 수행합니다. 이 함수는 모션이 완료되기 전까지 반환되지 않습니다.
r VT_I4 cemIxLineToStart ([in] VT_I4 MapIndex, [in] VT_PR8 PosList) 보간 제어에 있어 절대 좌표 직선 보간 이송을 수행합니다. 이 함수는 모션 시작 후 바로 반환됩니다.
r VT_I4 cemIxArcA ([in] VT_I4 MapIndex, [in] VT_R8 XCentOffset, [in] VT_R8 YCentOffset, [in] VT_R8 EndAngle, [in] VT_I4 IsBlocking) 보간 제어에 있어 원호 보간을 수행하며, 상대적 중심 좌표와 각도를 통해 원호 보간을 수행합니다. 이 함수는 모션이 완료되기 전까지 반환되지 않습니다.
r VT_I4 cemIxArcAStart ([in] VT_I4 MapIndex, [in] VT_R8 XCentOffset, [in] VT_R8 YCentOffset, [in] VT_R8 EndAngle) 보간 제어에 있어 원호 보간을 수행하며, 상대적 중심 좌표와 각도를 통해 원호 보간을 수행합니다. 이 함수는 모션 시작 후 바로 반환됩니다.
r VT_I4 cemIxArcATo ([in] VT_I4 MapIndex, [in] VT_R8 XCent, [in] VT_R8 YCent, [in] VT_R8 EndAngle, [in] VT_I4 IsBlocking) 보간 제어에 있어 원호 보간을 수행하며, 절대적 중심 좌표와 각도를 통해 원호 보간을 수행합니다. 이 함수는 모션이 완료되기 전까지 반환되지 않습니다.

r VT_I4 cemIxArcAToStart ([in] VT_I4 MapIndex, [in] VT_R8 XCent, [in] VT_R8 YCent, [in] VT_R8 EndAngle) 보간 제어에 있어 원호 보간을 수행하며, 절대적 중심 좌표와 각도를 통해 원호 보간을 수행합니다. 이 함수는 모션 시작 후 바로 반환됩니다.
r VT_I4 cemIxArcP ([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 제어에 있어 원호 보간을 수행하며, 상대적 중심 좌표와 종점 좌표를 통해 원호 보간을 수행합니다. 이 함수는 모션이 완료되기 전까지 반환되지 않습니다.
r VT_I4 cemIxArcPStart ([in] VT_I4 MapIndex, [in] VT_R8 XCentOffset, [in] VT_R8 YCentOffset, [in] VT_R8 XEndPointDist, [in] VT_R8 YEndPointDist, [in] VT_I4 Direction) 보간 제어에 있어 원호 보간을 수행하며, 상대적 중심 좌표와 종점 좌표를 통해 원호 보간을 수행합니다. 이 함수는 모션 시작 후 바로 반환됩니다.
r VT_I4 cemIxArcPTo ([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 제어에 있어 원호 보간을 수행하며, 절대적 중심 좌표와 종점 좌표를 통해 원호 보간을 수행합니다. 이 함수는 모션이 완료되기 전까지 반환되지 않습니다.
r VT_I4 cemIxArcPToStart ([in] VT_I4 MapIndex, [in] VT_R8 XCent, [in] VT_R8 YCent, [in] VT_R8 XEndPos, [in] VT_R8 YEndPos, [in] VT_I4 Direction) 보간 제어에 있어 원호 보간을 수행하며, 절대적 중심 좌표와 종점 좌표를 통해 원호 보간을 수행합니다. 이 함수는 모션 시작 후 바로 반환됩니다.
r VT_I4 cemIxArc3P ([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 제어에 있어 원호 보간을 수행하며, 현재 좌표와 상대적 두 좌표를 통해 원호 보간을 수행합니다. 이 함수는 모션이 완료되기 전까지 반환되지 않습니다.
r VT_I4 cemIxArc3PStart ([in] VT_I4 MapIndex, [in] VT_R8 P2X, [in] VT_R8 P2Y, [in] VT_R8 P3X, [in] VT_R8 P3Y, [in] VT_R8 EndAngle) 보간 제어에 있어 원호 보간을 수행하며, 현재 좌표와 상대적 두 좌표를 통해 원호 보간을 수행합니다. 이 함수는 모션 시작 후 바로 반환됩니다.
r VT_I4 cemIxArc3PTo ([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 제어에 있어 원호 보간을 수행하며, 현재 좌표와 절대적 두 좌표를 통해 원호 보간을 수행 합니다. 이 함수는 모션이 완료되기 전까지 반환되지 않습니다.
r VT_I4 cemIxArc3PToStart ([in] VT_I4 MapIndex, [in] VT_R8 P2X, [in] VT_R8 P2Y, [in] VT_R8 P3X, [in] VT_R8 P3Y, [in] VT_R8 EndAngle) 보간 제어에 있어 원호 보간을 수행하며, 현재 좌표와 절대적 두 좌표를 통해 원호 보간을 수행 합니다. 이 함수는 모션 시작 후 바로 반환됩니다.
r VT_I4 cemIxStop ([in] VT_I4 MapIndex) 보간 제어 구동 이송을 감속 후 정지합니다.
r VT_I4 cemIxStopEmg ([in] VT_I4 MapIndex) 보간 제어 구동 이송을 비상 정지합니다.
r VT_I4 cemIxIsDone ([in] VT_I4 MapIndex, [out] VT_PI4 IsDone) 보간 제어 구동 이송의 완료를 확인합니다.
r VT_I4 cemIxWaitDone ([in] VT_I4 MapIndex, [in] VT_I4 IsBlocking) 보간 제어 구동 이송의 완료 시점까지 대기합니다.

7.2.4 함수 설명

NAME

cemIxMapAxes

- 보간 대상 축 그룹(Group) 설정

INFORMATION
1 Interpolation Motion
! VC++ (6, 7, 8)/VB
BCB/Delphi
: Level 3
K 다소 주의
보간 대상 축 그룹 설정은 모든 보간 이송 작업 전에 선행되어 야 합니다.

SYNOPSIS

r VT_I4 cemIxMapAxes ([in] VT_I4 MapIndex, [in] VT_I4 NodeID, [in] VT_I4 MapMask1, [in] VT_I4 MapMask2)

DESCRIPTION

cemIxMapAxes 함수는 보간 작업을 수행할 축들을 맵 번호(Map index)로 맵핑(Mapping) 합니다. 맵 번호는 다른 “기본보간제어”에 관련된 함수들의 첫 번째 매개변수로 전달됨으로써 각 보간 함수들이 제어해야 할 축들에 대한 정보가 간편하게 전달됩니다. 따라서 다른 “기본보간제어”에 관련된 함수들을 사용하기 전에 가장 먼저 이 함수를 사용하여 “기본보간제어”에 사용할 축들을 맵핑 하여야 합니다.

PARAMETER

▶ ***MapIndex*** : 맵 번호(Map index). 맵 번호는 반드시 0 ~ 15 안에서 설정하여야 합니다.

▶ ***NodeID*** : 원격 노드 ID. 보간 제어를 수행할 원격 노드의 ID 를 설정합니다.

▶ ***MapMask1*** : 축 맵에 포함할 축들을 지정할 마스크 값(하위 32 비트, BIT0 ~ BIT31).

보간 제어는 대상 원격 노드 내의 축들로 제한되며, 축 그룹에 포함할 축 번호는 원격 노드안에서의 제한된 통합 축 번호로 설정하여야 합니다. 즉, 대상 원격 노드의 첫번째 모션 모듈에 속한 첫번째 축이 0 번 축이 됩니다.

이 값의 BIT0~BIT31 을 이용하여 그룹에 포함할 축들을 지정합니다. 각 비트의 값이 0 이면 해당 축(비트의 순서와 일치하는 축)은 배제되는 것이며 1 이면 해당 축이 포함되는 것입니다. 이 매개변수의 각 비트 별 정보는 아래 표와 같습니다.

▶ ***MapMask2*** : 축 맵에 포함할 축들을 지정할 마스크 값(상위 32 비트, BIT32 ~ BIT63).

예) 8 축을 사용하는 경우

Bit Number	Meaning
BIT0 (cemX1_MASK)	0 번 축의 포함여부 : 0 => 포함안함, 1 => 포함
BIT1 (cemY1_MASK)	1 번 축의 포함여부 : 0 => 포함안함, 1 => 포함
BIT2 (cemZ1_MASK)	2 번 축의 포함여부 : 0 => 포함안함, 1 => 포함
BIT3 (cemU1_MASK)	3 번 축의 포함여부 : 0 => 포함안함, 1 => 포함
BIT4 (cemX2_MASK)	4 번 축의 포함여부 : 0 => 포함안함, 1 => 포함
BIT5 (cemY2_MASK)	5 번 축의 포함여부 : 0 => 포함안함, 1 => 포함
BIT6 (cemZ2_MASK)	6 번 축의 포함여부 : 0 => 포함안함, 1 => 포함
BIT7 (cemU2_MASK)	7 번 축의 포함여부 : 0 => 포함안함, 1 => 포함

RETURN VALUE

Value	Meaning
음수	수행 실패. 자세한 내용은 '에러처리' 편을 참고합니다.
0 (ceERR_NONE)	수행 성공.

SEE ALSO

cemIxUnMap

EXAMPLE

C/C++

```
#include “ceSDK.h”
#include “ceSDKDef.h”

#define MAP0 0        // 보간 맵 번호

/* 다음과 같은 2 개의 원격 노드가 존재한다고 가정합니다.
Node Master 1 : NodeID = 1, 16 Axes
Node Master 2 : NodeID = 2, 8 Axes

통합 축 번호 16 번, 17 번 축을 대상으로 보간 맵 설정을 수행합니다.
보간 제어는 원격 노드안의 제한된 통합 축 번호로 관리되므로, 2 번째 원격 노드에서의
제한된 통합 축 번호인 0 번, 1 번 축을 MapMask 로 설정하여야 합니다. */

// 2 번 노드에 있는 0 번 축과 1 번 축을 하나의 맵으로 묶습니다.
cemIxMapAxes ( MAP0, 2, cemX1_MASK | cemY1_MASK, 0 );
// 또는 cemIxMapAxes ( MAP0, 2, 0x3, 0x0 );

// MAP0 로 묶여 있는 축들을 해제 합니다.
// cemIxUnMap ( MAP0 );
```

Visual Basic

```
‘ 맵 번호 MAP0 는 이미 선언되어 있다고 가정합니다
‘ 보간 제어는 원격 노드안의 제한된 통합 축 번호로 관리됩니다.

‘ 2 번 노드에 있는 0 번 축과 1 번 축을 하나의 축 맵으로 설정합니다.
Call cemIxMapAxes ( MAP0, 2, cemX1_MASK Or cemY1_MASK, 0 )
‘ 또는 Call cemIxMapAxes ( MAP0, 2, &H3, &H0 )

‘ 맵 번호 MAP0 의 축 그룹을 해제 합니다.
‘ cemIxUnMap( MAP0 )
```

Delphi

```
// 맵 번호 MAP0 는 이미 선언되어 있다고 가정합니다
// 보간 제어는 원격 노드안의 제한된 통합 축 번호로 관리됩니다.

begin
    // 2 번 노드에 있는 0 번 축과 1 번 축을 하나의 맵으로 설정합니다.
    cemIxMapAxes ( MAP0, 2, cemX1_MASK or cemY1_MASK, 0 );
    // 또는 cemIxMapAxes ( MAP0, 2, $3, $0 );

    // 맵 번호 MAP0 의 축 그룹을 해제 합니다.
    // cemIxUnMap( MAP0 );

end;
```

NAME

cemIxUnMap

- 보간 대상 축 그룹(Group) 해제

INFORMATION
1 Interpolation Motion
! VC++ (6, 7, 8)/VB
BCB/Delphi
: Level 3
K 다소 주의
대상 축 그룹에 대하여 보간 이송 작업 완료 후, 보간 제어 사용 종료를 하고자 하는 경우에 해제합니다.

SYNOPSIS

r VT_I4 cemIxUnMap ([in] VT_I4 MapIndex)

DESCRIPTION

보간 이송 작업을 수행할 보간 대상 축 그룹을 해제합니다.

PARAMETER

▶ ***MapIndex*** : 맵 번호(Map index). 보간 대상 축 그룹을 해제할 맵 번호를 설정합니다.

RETURN VALUE

Value	Meaning
음수	수행 실패. 자세한 내용은 '에러처리' 편을 참고합니다.
0 (ceERR_NONE)	수행 성공.

SEE ALSO

cemIxMapAxes

EXAMPLE

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

NAME

cemIxSpeedPattern_Set /

cemIxSpeedPattern_Get

- 보간 이송 속도 설정 및 반환

INFORMATION	
1	Interpolation Motion
!	VC++ (6, 7, 8)/VB
	BCB/Delphi
:	Level 3
K	다소 주의

SYNOPSIS

r VT_I4 cemIxSpeedPattern_Set ([in] VT_I4 MapIndex, [in] VT_I4 IsVectorSpeed,
[in] VT_I4 SpeedMode, [in] VT_R8 Vel, [in] VT_R8 Acc, [in] VT_R8 Dec)

r VT_I4 cemIxSpeedPattern_Get ([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

cemIxSpeedPattern_Set 함수는 “기본보간제어”의 이송 속도에 대한 환경을 설정합니다. 사용자가 지정한 작업 속도는 “IsVectorSpeed”의 설정 값이 ‘1(CE_TRUE)’이면 벡터 속도 모드(Vector Speed Mode), ‘0(CE_FALSE)’이면 마스터 속도 모드(Master Speed Mode)가 적용됩니다. “벡터 속도 모드”에 대한 자세한 내용은 아래의 “REFERENCE” 항목을 참조하십시오.

cemIxSpeedPattern_Get 함수는 “기본보간제어” 의 이송 속도에 대한 설정된 속도 환경을 반환합니다.

PARAMETER

▶ ***MapIndex*** : 맵 번호(Map index). 이 맵 번호를 사용하기 전에 먼저 cemIxMapAxes 함수를 통하여 해당 맵번호에 유효한 축들이 맵핑 되어 있어야 합니다.

▶ ***IsVectorSpeed*** : 마스터 속도 모드 또는 벡터 속도 모드를 설정 혹은 반환합니다.

Value	Meaning
0 (CE_FALSE)	마스터 속도 모드. (Master Speed Mode)
1 (CE_TRUE)	벡터 속도 모드. (Vector Speed Mode)

▶ ***SpeedMode*** : 가감속 모드를 설정 혹은 반환합니다. 설정 값은 다음과 같습니다.

Value	Meaning
0 (cemSMODE_C)	CONSTANT 속도모드 => 가감속을 수행하지 않습니다.
1 (cemSMODE_T)	TRAPEZOIDAL 속도모드 => 사다리꼴 가감속을 수행합니다.
2 (cemSMODE_S)	S-CURVE 속도모드 => S-CURVE 가감속을 수행합니다.

▶ *Vel* : 마스터 속도 모드 일 때는 기준 작업 속도의 비율(%)을, 벡터 속도 모드 일 때는 PPS 단위를 사용하여 작업 속도를 설정 혹은 반환합니다.

▶ *Acc* : 마스터 속도 모드 일 때는 기준 가속도의 비율(%)을, 벡터 속도 모드 일 때는 PPS 단위를 사용하여 가속도를 설정 혹은 반환합니다.

▶ *Dec* : 마스터 속도 모드 일 때는 기준 감속도의 비율(%)을, 벡터 속도 모드 일 때는 PPS 단위를 사용하여 감속도를 설정 혹은 반환합니다.

RETURN VALUE

Value	Meaning
음수	수행 실패. 자세한 내용은 '에러처리' 편을 참고합니다
0 (ceERR_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 서보 드라이버나 스텝 드라이버에서는 보간 속도 설정에 반드시 주의를 하셔야 합니다.

□ 보간 작업 속도를 벡터 속도로 설정해야만 하는 특별한 경우를 제외하고는 보간 작업 속도를 마스터 속도로 설정하는 것이 모터의 최대속도를 활용하는데 있어서 편리합니다.

□ 직선 보간 이송시 작업속도의 적용

마스터 속도 모드(Master Speed Mode)로 보간 작업 시에는 각 축의 속도가 각 축의 이송거리에 비례하여 자동으로 설정됩니다. 이때 cemIxSpeedPattern_Set 함수의 WorkSpeed 매개변수를 통하여 지정되는 보간 작업 속도는 마스터 축의 작업 속도로 적용됩니다.

각 보간 이송시에 이송거리가 가장 큰 축을 "마스터 축" 이라고 하며 마스터 축의 속도를 "마스터 속도"라 합니다. 각 보간 이송시에 마스터 축의 속도는 사용자가 지정한 보간 작업속도로 설정되며, 마스터 축 이외의 다른 축의 속도는 마스터 축과 해당 축의 이송 거리 비에 따라서 자동으로 설정됩니다.

□ 보간 작업속도의 적용 예

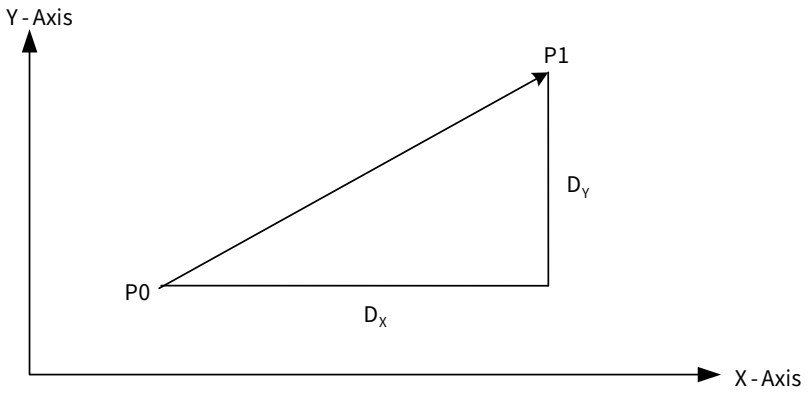
cemIxSpeedPattern_Set 함수의 WorkSpeed 를 10000 으로 설정하고 X, Y, Z 축의 보간 작업을 수행하는 경우에 이송 거리에 따른 각 축의 속도 관계는 다음과 같습니다. (표에서 배경이 회색으로 되어 있는 것은 마스터 축임을 의미하는 것입니다.)

	이송 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 6] 마스터 속도 모드에 대한 실제 속도 적용 예시 표

□ 직선 보간 이동 시의 벡터 속도

[그림 7-2] 은 2 축(편의상 X, Y 축으로 가정) 직선 보간 이송을 그래프로 나타낸 것입니다.



[그림 7-2] X, Y 축간의 직선 보간 이송

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

$$DP = \sqrt{D_X{}^2 + D_Y{}^2}$$

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.

$$V_X = \frac{D_X \times V}{\sqrt{D_X{}^2 + D_Y{}^2}} \qquad V_Y = \frac{D_Y \times V}{\sqrt{D_X{}^2 + D_Y{}^2}}$$

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

3 축(편의상 X, Y, Z 축으로 가정)의 경우 각 축의 속도에 대한 관계식은 다음과 같습니다.

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

4 축의 경우에는 각축의 속도에 대한 관계식은 다음과 같습니다.

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

EXAMPLE

C/C++

```
#include “ceSDK.h”
#include “ceSDKDef.h”

#define MAP0        0       // 보간 맵 정의
#define NODE_ID     1       // 해당 원격 노드 ID 정의

cemIxMapAxes ( MAP0, NODE_ID, cemX1_MASK | cemY1_MASK, 0 );

/* MAP0 로 설정된 축들을 벡터 스피드 모드로
작업 속도 1000, 가속도 10000, 감속도 10000 으로 설정합니다.*/
cemIxSpeedPattern_Set (     MAP0,           // 보간 맵 번호
                            CE_TRUE,        // CE_FALSE : 마스터 모드, CE_TRUE : 벡터 모드
                            cemSMODE_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};
cemIxLineStart ( MAP0, fDistList );
```

Visual Basic

```
Dim nMapIdx As Long          '보간 맵 정의
Dim nNodeID As Long          '해당 원격 노드 ID 정의
Dim fDistList(2) As Double   ' 이송 좌표값의 배열 정보

nMapIdx = 0
nNodeID = 1

Call cemIxMapAxes ( nMapIdx, nNodeID, cemX1_MASK Or cemY1_MASK, 0 )

' MAP0 로 설정된 축들을 벡터 스피드 모드로
' 작업 속도 1000, 가속도 10000, 감속도 10000 으로 설정합니다.
Call cemIxSpeedPattern_Set ( nMapIdx, CE_TRUE, cemSMODE_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
cemIxLineStart ( nMapIdx, fDistList(0) )
```

Delphi

```
var
    nMapIdx  : LongInt;                 // 보간 맵 정의
    nNodeID : LongInt;                  // 해당 원격 노드 ID 정의
    fDistList : Array[0..1] of Double;  // 이송 좌표값의 배열 정보

begin
    nMapIdx := 0;
    nNodeID := 1;

    cemIxMapAxes ( nMapIdx, nNodeID, cemX1_MASK or cemY1_MASK, 0 );

    { MAP0 로 설정된 축들을 벡터 스피드 모드로
    작업 속도 1000, 가속도 10000, 감속도 10000 으로 설정합니다. }
    cemIxSpeedPattern_Set ( nMapIdx, CE_TRUE, cemSMODE_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;
    cemIxLineStart ( nMapIdx, @fDistList );

end;
```

NAME

cemIxVelCorrMode_Set /

cemIxVelCorrMode_Get

- 직선 보간 속도 보정 모드 설정 및 반환

INFORMATION	
1	Interpolation Motion
!	VC++ (6, 7, 8)/VB
	BCB/Delphi
:	Level 3
K	다소 주의

SYNOPSIS

r VT_I4 cemIxVelCorrMode_Set ([in] VT_I4 MapIndex, [in] VT_I4 VelCorrOpt1, [in] VT_I4 VelCorrOpt2)

r VT_I4 cemIxVelCorrMode_Get ([in] VT_I4 MapIndex, [out] VT_PI4 VelCorrOpt1, [out] VT_PI4 VelCorrOpt2)

DESCRIPTION

cemIxVelCorrMode_Set 함수는 직선 보간 속도 보정 모드를 설정합니다.

직선 보간 이송 시에 각 축의 속도는 각 축의 이송 거리에 비례하게 됩니다. 따라서 이송 거리가 가장 긴 축의 속도가 가장 빠르게 되며, 이 축을 마스터축(Master axis)이라 하고 나머지 축들을 슬레이브축(Slave axis)이라 합니다.

기본적으로 직선보간 시에 마스터축의 속도는 cemIxSpeedPattern_Set 함수에 의해서 지정됩니다. 그리고 슬레이브축들의 속도는 마스터축과의 이송 거리비에 비례하여 이송속도가 계산되게 됩니다. 그런데 이렇게 계산된 슬레이브축의 이송속도가 정격속도를 초과하게 되면 문제가 발생할 수 있습니다. 예를 들어, 스텝모터를 사용하는 경우에는 탈조현상이 일어날 수 있습니다.
따라서 이러한 경우에는 슬레이브축의 속도를 정격속도 이하로 낮추는 것이 바람직한데 슬레이브축의 속도만을 낮추면 보간 경로를 유지하지 못하므로 마스터축의 속도를 조정하여 결과적으로 슬레이브축의 속도가 낮아지도록 해야합니다. ㈜커미조아의 cEIP 모션제어 모듈은 이러한 보정을 자동으로 수행해줍니다.

cemIxVelCorrMode_Get 함수는 설정된 직선 보간 속도 보정 모드를 반환합니다.

PARAMETER

▶ ***MapIndex*** : 맵 번호(Map index). 이 맵번호를 사용하기 전에 먼저 cemIxMapAxes 함수를 통하여 해당 맵 번호에 유효한 축들이 맵핑 되어 있어야 합니다.

▶ *VelCorrOpt1* : 직선 보간 속도 보정을 할 것인지를 설정 혹은 설정 상태를 반환합니다.

Value	Meaning
0	속도 보정을 하지 않습니다.
1	작업 속도만 보정합니다.
2 [Default]	작업 속도 및 가속도, 감속도를 모두 보정합니다. (스텝모터의 경우 작업속도가 크지 않아도 가속도, 감속도가 정격속도를 넘어서면 탈조현상이 일어날 수 있으므로 가속도, 감속도 모두 보정을 해주는 것을 권장합니다.)

▶ *VelCorrOpt2* : 직선 보간 속도 보정 시 슬레이브 축의 작업 속도 및 가속, 감속 한계 값을 정격 속도만을 사용할 지, 아니면 정격 속도에 보간 속도비(%) 를 곱한 값을 사용할 지를 설정 혹은 설정 상태를 반환합니다.

Value	Meaning
0	작업 속도 및 가속, 감속 모두에 대하여 정격 값 설정을 이송 속도의 한계 값으로 적용합니다.
1 [Default]	작업속도의 한계 값을 정격 속도에 보간 속도비(%)를 곱한 값을 한계 값으로 적용하고, 가속, 감속의 한계 값은 정격 값을 적용합니다 .
2	작업 속도와 가속, 감속 모두에 대하여 정격 속도에 보간 속도비(%)를 곱한 값을 한계 값으로 적용합니다.

이 옵션은 보간 속도 모드가 마스터 속도 모드로 설정된 경우에만 적용되며 벡터 속도 모드에서는 무시됩니다.

RETURN VALUE

Value	Meaning
음수	수행 실패. 자세한 내용은 '에러처리' 편을 참고합니다.
0 (ceERR_NONE)	수행 성공.

EXAMPLE

C/C++

```
#include “ceSDK.h”
#include “ceSDKDef.h”

#define MAP0          0
#define NODE_ID     1

cemIxMapAxes ( MAP0, NODE_ID, cemX1_MASK | cemY1_MASK, 0 );

/*MAP0 로 설정된 축들에 대하여 직선 보간 속도 보정을 작업 속도 및 가감속에 적용하고,
정격 속도에 보간 속도비(%)를 곱한 값을 한계 값으로 적용 되도록 설정합니다. */

Long nVelCorrOpt1 = 2, nVelCorrOpt2 = 2;
cemIxVelCorrMode_Set ( MAP0, nVelCorrOpt1, nVelCorrOpt2 );

// 설정된 직선 보간 속도 보정 모드를 반환합니다.
cemIxSpeedPattern_Get ( MAP0, &nVelCorrOpt1, &nVelCorrOpt2 );
```

Visual Basic

```
Dim nMapIdx As Long          ‘보간 맵 정의
Dim nNodeID As Long          ‘해당 원격 노드 ID 정의
Dim nVelCorrOpt1 As Long, nVelCorrOpt2 As Long        ‘ 속도 보정 모드 정보

nVelCorrOpt1 = 2     ‘ 작업 속도 및 가속도, 감속도를 모두 보정
nVelCorrOpt2 = 2     ‘ 속도 설정에 대하여 정격 속도에 보간 속도비(%)를 곱한 값을 한계 값으로 적용

Call cemIxMapAxes ( nMapIdx, nNodeID, cemX1_MASK Or cemY1_MASK, 0 )

‘ nMapIdx 로 설정된 축들에 대하여 직선 보간 속도 보정을 작업 속도 및 가감속에 적용하고,
‘ 정격 속도에 보간 속도비(%)를 곱한 값을 한계 값으로 적용 되도록 설정합니다. */

Call cemIxVelCorrMode_Set ( nMapIdx, nVelCorrOpt1, VelCorrOpt2 )

‘ 설정된 직선 보간 속도 보정 모드를 반환합니다.
Call cemIxSpeedPattern_Get ( MAP0, nVelCorrOpt1, VelCorrOpt2 )
```

Delphi

```
var
    nMapIdx : LongInt;                          // 보간 맵 정의
    nNodeID : LongInt;                          // 해당 원격 노드 ID 정의
    nVelCorrOpt1, nVelCorrOpt2 : LongInt;       // 속도 보정 모드 정보

begin
```

```
nVelCorrOpt1 := 2;          // 작업 속도 및 가속도, 감속도를 모두 보정
nVelCorrOpt2 := 2;          // 속도 설정에 대하여 정격 속도에 보간 속도비(%)를 곱한 값을 한계 값으로 적용

cemIxMapAxes ( nMapIdx, nNodeID, cemX1_MASK or cemY1_MASK, 0 );

{ nMapIdx 로 설정된 축들에 대하여 직선 보간 속도 보정을 작업 속도 및 가감속에 적용하고,
정격 속도에 보간 속도비(%)를 곱한 값을 한계 값으로 적용 되도록 설정합니다. }

cemIxVelCorrMode_Set ( nMapIdx, nVelCorrOpt1, VelCorrOpt2 );

// 설정된 직선 보간 속도 보정 모드를 반환합니다.
cemIxSpeedPattern_Get ( MAP0, @nVelCorrOpt1, @VelCorrOpt2 );
```

NAME

cemIxLine / cemIxLineStart

- 직선보간 상대좌표 이송

INFORMATION
1 Interpolation Motion
! VC++ (6, 7, 8)/VB
BCB/Delphi
: Level 3
K 이송 함수
실제 이송이 진행되는 함수이므로, 사전에 반드시 안전을 확인합니다.

SYNOPSIS

r VT_I4 cemIxLine ([in] VT_I4 MapIndex, [in] VT_PR8 DistList, [in] VT_I4 IsBlocking)

r VT_I4 cemIxLineStart ([in] VT_I4 MapIndex, [in] VT_PR8 DistList)

DESCRIPTION

cemIxLine / cemIxLineStart 함수는 현재 위치로부터의 상대 좌표 직선 보간 이송을 수행합니다.

cemIxLine 함수는 모션이 완료되기 전까지 반환되지 않으며, cemIxLineStart 함수는 모션을 시작시킨 후에 바로 반환됩니다.

PARAMETER

▶ ***MapIndex*** : 맵 번호(Map index). 이 맵 번호를 사용하기 전에 먼저 cemIxMapAxes 함수를 통하여 해당 맵 번호에 유효한 축들이 맵핑 되어 있어야 합니다.

▶ ***DistList*** : 현재 위치로부터의 상대적인 이송 좌표 값(각 축의 이송 거리 값)의 배열 주소. 이 배열의 크기는 cemIxMapAxes 함수를 통하여 맵핑된 축 수와 일치하여야 합니다. 거리의 단위는 “Unit Distance”에 의해 정의되는 논리적 거리를 적용합니다.

▶ ***IsBlocking*** : cemIxLine 함수의 인자이며, 완료될 때까지 기다리는 동안 윈도우 메시지를 블록(Blocking)할 것인지를 결정합니다.

Value	Meaning
0 (CE_FALSE)	윈도우 메시지를 블록(Blocking) 하지 않습니다. 따라서 해당 모션이 진행되는 동안 윈도우 이벤트를 처리합니다.
1 (CE_TRUE)	윈도우 메시지를 블록(Blocking) 합니다. 따라서 해당 모션이 완료되는 동안에는 윈도우 이벤트가 처리되지 않습니다.

RETURN VALUE

Value	Meaning
음수	수행 실패. 자세한 내용은 '에러처리' 편을 참고합니다.
0 (ceERR_NONE)	수행 성공.

SEE ALSO

cemIxLineTo, cemIxLineToStart

REFERENCE

□ 논리적 단위 거리는 cemCfgUnitDist_Set 함수에 의해 결정됩니다.

□ cemIxLineStart 함수를 사용하는 경우에는 cemIxIsDone 함수나 cemIxWaitDone 함수를 사용하여 모션의 완료를 확인할 수 있습니다.

□ cemIxLine 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.

스텝 드라이버는 INP 출력이 없는 경우가 일반적인데, 고객님의 부주의나 잘못된 설정으로 INP 입력에 대한 설정이 Enable 로 되어 있을 경우 INP 입력이 스텝 드라이버를 통해 발생하지 않는 이유 때문에 모션 완료가 되지 않는 경우가 발생할 수 있습니다.

고객 여러분들께서는 스텝 드라이버 사용시에 이점을 주의해 주시기를 부탁 드립니다.

윈도우 이벤트라는 것은 무엇입니까?

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

EXAMPLE

C/C++

```
#include “ceSDK.h”
#include “ceSDKDef.h”

/* 원격 노드(Node ID : 1)의 0, 1 번 축에 대하여 마스터 속도 모드로 상대 좌표 직선 보간 이송을 수행하는
예제입니다. */

long nIxMap = 0;                // 보간 맵 설정
long nNodeID = 1;               // 해당 원격 노드 ID 설정

void OnSetIxConfig ()
{
    cemIxMapAxes ( nIxMap, nNodeID, cemX1_MASK | cemY1_MASK, 0 );

    /* 보간 속도 패턴 설정에서 벡터 모드를 사용하지 않을 경우에는
    각 축에 대한 속도를 미리 설정해 주어야 합니다. */
    cemCfgSpeedPattern_Set ( cemX1, cemSMODE_T, 4000, 20000, 20000 );
    cemCfgSpeedPattern_Set ( cemY1, cemSMODE_T, 5000, 20000, 20000 );

    /* 마스터 속도 모드(벡터 모드를 사용하지 않음)로 사용할 경우에는 속도를
    미리 설정한 기준 속도의 비율로 설정해 줍니다. */
    cemIxSpeedPattern_Set ( nIxMap, CE_FALSE, cemSMODE_T, 100, 80, 80 );
}

void OnIxLine_Move ()
{
    //두 축에 대한 이동 거리를 설정합니다.
    double fDistList[2] = { 13000, 9000 };

    //보간 구동을 시작합니다.
    cemIxLine( nIxMap, fDistList, CE_FALSE );

    //cemIxLineStart() 함수를 사용하는 경우에는 다음과 같이 코드를 작성합니다.
    //cemIxLineStart( MAP0, fDistList );
    //cemIxWaitDone( MAP0, CE_FALSE );
}
```

Visual Basic

```
‘ 원격 노드(Node ID : 1)의 0, 1 번 축에 대하여 마스터 속도 모드로 상대 좌표 직선 보간 이송을 수행하는
‘ 예제입니다.

‘ 맵 번호 MAP0 는 이미 선언되어 있다고 가정합니다

Private Sub OnSetIxConfig ()
    Call cemIxMapAxes( MAP0, 1, cemX1_MASK Or cemY1_MASK, 0 )

    ‘ 보간 속도 패턴 설정에서 벡터 모드를 사용하지 않을 경우에는
```

```
    ‘ 각 축에 대한 속도를 미리 설정해 주어야 합니다.
    Call cemCfgSpeedPattern_Set ( cemX1, cemSMODE_T, 4000, 20000, 20000 )
    Call cemCfgSpeedPattern_Set ( cemY1, cemSMODE_T, 5000, 20000, 20000 )

    ‘ 각 축의 기준 속도의 비율로 마스터 속도 모드를 설정합니다.
    Call cemIxSpeedPattern_Set ( MAP0, CE_FALSE, cemSMODE_T, 100, 80, 80 )

End Sub


Private Sub OnIxLine_Move ()

    Dim fDistList(2) As Double
    fDistList(0) = 13000
    fDistList(1) = 9000

    ‘ 보간 구동을 시작합니다.
    Call cemIxLine ( MAP0, fDistList(0), CE_FALSE )

End Sub
```

Delphi

```
{ 원격 노드(Node ID : 1)의 0, 1 번 축에 대하여 마스터 속도 모드로 상대 좌표 직선 보간 이송을 수행하는
예제입니다. }

// 맵 번호 MAP0 는 이미 선언되어 있다고 가정합니다

procedure OnSetIxConfig ();
begin
    cemIxMapAxes( MAP0, 1, cemX1_MASK or cemY1_MASK, 0 )

    { 보간 속도 패턴 설정에서 벡터 모드를 사용하지 않을 경우에는
    각 축에 대한 속도를 미리 설정해 주어야 합니다. }
    cemCfgSpeedPattern_Set ( cemX1, cemSMODE_T, 4000, 20000, 20000 );
    cemCfgSpeedPattern_Set ( cemY1, cemSMODE_T, 5000, 20000, 20000 );

    // 각 축의 기준 속도의 비율로 마스터 속도 모드를 설정합니다.
    cemIxSpeedPattern_Set ( MAP0, CE_FALSE, cemSMODE_T, 100, 80, 80 );

end;

procedure OnIxLine_Move ();
var
    fDistList : Array[0..1] of Double;

begin
    fDistList[0] := 13000;
    fDistList[1] := 9000;

    // 보간 구동을 시작합니다.
    cemIxLine ( MAP0, @fDistList, CE_FALSE );

end;
```

NAME

cemIxLineTo / cemIxLineToStart

- 직선보간 절대좌표 이송

INFORMATION
1 Interpolation Motion
! VC++ (6, 7, 8)/VB
BCB/Delphi
: Level 3
K 이송 함수
실제 이송이 진행되는 함수이므로, 사전에 반드시 안전을 확인합니다.

SYNOPSIS

r VT_I4 cemIxLineTo ([in] VT_I4 MapIndex, [in] VT_PR8 PosList, [in] VT_I4 IsBlocking)

r VT_I4 cemIxLineToStart ([in] VT_I4 MapIndex, [in] VT_PR8 PosList)

DESCRIPTION

cemIxLineTo / cemIxLineToStart 함수는 절대 좌표 직선 보간 이송을 수행합니다.

cemIxLineTo 함수는 모션이 완료되기 전까지 반환되지 않으며, cemIxLineToStart 함수는 모션을 시작시킨 후에 바로 반환됩니다.

PARAMETER

▶ ***MapIndex*** : 맵 번호(Map index). 이 맵 번호를 사용하기 전에 먼저 cemIxMapAxes 함수를 통하여 해당 맵 번호에 유효한 축들이 맵핑 되어 있어야 합니다.

▶ ***PosList*** : 이송할 목표 절대 좌표 값(각 축의 절대 좌표 값)의 배열 주소. 이 목표 좌표 배열의 크기는 cemIxMapAxes 함수를 통하여 맵핑된 축의 수와 일치하여야 합니다. 거리의 단위는 “Unit Distance”에 의해 정의되는 논리적 거리를 적용합니다.

▶ ***IsBlocking*** : cemIxLineTo 함수의 인자이며, 완료될 때까지 기다리는 동안 윈도우 메시지를 블록(Blocking)할 것인지를 결정합니다.

Value	Meaning
0 (CE_FALSE)	윈도우 메시지를 블록(Blocking) 하지 않습니다. 따라서 해당 모션이 진행되는 동안 윈도우 이벤트를 처리합니다.
1 (CE_TRUE)	윈도우 메시지를 블록(Blocking) 합니다. 따라서 해당 모션이 완료되는 동안에는 윈도우 이벤트가 처리되지 않습니다.

RETURN VALUE

Value	Meaning
음수	수행 실패. 자세한 내용은 '에러처리' 편을 참고합니다.
0 (ceERR_NONE)	수행 성공.

SEE ALSO

cemIxLine, cemIxLineStart

REFERENCE

□ 논리적 단위 거리는 cemCfgUnitDist_Set 함수에 의해 결정됩니다.

□ cemIxLineToStart 함수를 사용하는 경우에는 cemIxIsDone 함수나 cemIxWaitDone 함수를 사용하여 모션의 완료를 확인할 수 있습니다.

□ cemIxLineTo 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.

스텝 드라이버는 INP 출력이 없는 경우가 일반적인데, 고객님의 부주의나 잘못된 설정으로 INP 입력에 대한 설정이 Enable 로 되어 있을 경우 INP 입력이 스텝 드라이버를 통해 발생하지 않는 이유 때문에 모션 완료가 되지 않는 경우가 발생할 수 있습니다.

고객 여러분들께서는 스텝 드라이버 사용시에 이점을 주의해 주시기를 부탁 드립니다.

윈도우 이벤트라는 것은 무엇입니까?

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

EXAMPLE

```
//* cemIxLine / cemIxLineStart 예제를 참고하여 주시기 바랍니다.
```

NAME

cemIxArcA / cemIxArcAStart

- 원호 보간 상대 좌표 이송

(상대적 중심 좌표와 각도)

INFORMATION
1 Interpolation Motion
! VC++ (6, 7, 8)/VB
BCB/Delphi
: Level 3
K 이송 함수
실제 이송이 진행되는 함수이므로, 사전에 반드시 안전을 확인합니다.

SYNOPSIS

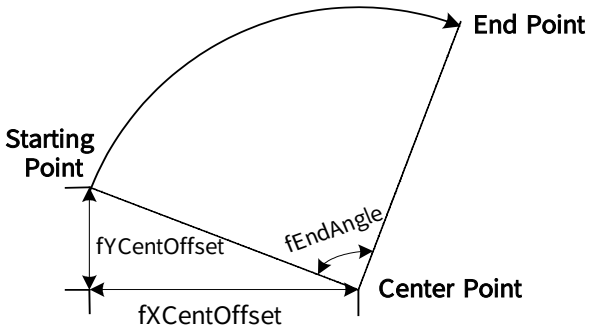
r VT_I4 cemIxArcA ([in] VT_I4 MapIndex, [in] VT_R8 XCentOffset, [in] VT_R8 YCentOffset, [in] VT_R8 EndAngle, [in] VT_I4 IsBlocking)

r VT_I4 cemIxArcAStart ([in] VT_I4 MapIndex, [in] VT_R8 XCentOffset, [in] VT_R8 YCentOffset, [in] VT_R8 EndAngle)

DESCRIPTION

cemIxArcA / cemIxArcAStart 함수는 중심 좌표와 원호의 각도를 매개변수로 하여 원호 보간 이송을 수행합니다. 이때 중심 좌표는 상대 좌표로 표현됩니다. cemIxArcA 함수는 모션이 완료되기 전까지 반환되지 않으며, cemIxArcAStart 함수는 모션을 시작시킨 후에 바로 반환됩니다.

원호 보간은 임의의 두 축에 대해서 적용됩니다. 아래 설명에서는 맵핑된 두 축을 X, Y 축으로 간주하여 설명합니다. 여기서 X 축이라 함은 맵핑된 두 축 중에서 축 번호가 낮은 축을 의미하며 Y 축은 축 번호가 높은 축을 의미합니다. 예를 들어 Z 축과 U 축이 맵핑된 두 축이라면 Z 축이 X 축에 해당하며 U 축이 Y 축에 해당합니다.



PARAMETER

▶ ***MapIndex*** : 맵 번호(Map index). 이 맵 번호를 사용하기 전에 먼저 cemIxMapAxes 함수를 통하여 해당 맵 번호에 유효한 축들이 맵핑 되어 있어야 합니다.

▶ ***XCentOffset*** : 현재 위치(시작 위치)로부터 원의 중심까지 X 축 상대 좌표 값.

▶ ***YCentOffset*** : 현재 위치(시작 위치)로부터 원의 중심까지 Y 축 상대 좌표 값.

▶ ***EndAngle*** : 원호 보간 이송을 완료할 목표 지점의 현재 위치에 대한 각도 값을 Degree(°)값으로 지정합니다. 각도의 부호가 (+)이면 반 시계방향, (-)이면 시계방향으로의 이동을 의미합니다.

▶ ***IsBlocking*** : cemIxArcA 함수의 인자이며, 완료될 때까지 기다리는 동안 윈도우 메시지를 블록(Blocking)할 것인지를 결정합니다. 단, 쓰레드내에서 실행할 때는 이 값을 1(CE_TRUE)로 설정해 주어야 합니다.

Value	Meaning
0 (CE_FALSE)	윈도우 메시지를 블록(Blocking) 하지 않습니다. 따라서 해당 모션이 진행되는 동안 윈도우 이벤트를 처리합니다.
1 (CE_TRUE)	윈도우 메시지를 블록(Blocking) 합니다. 따라서 해당 모션이 완료되는 동안에는 윈도우 이벤트가 처리되지 않습니다.

RETURN VALUE

Value	Meaning
음수	수행 실패. 자세한 내용은 '에러처리' 편을 참고합니다.
0 (ceERR_NONE)	수행 성공.

SEE ALSO

cemIxArcATo, cemIxArcAToStart

REFERENCE

□ cemIxArcAStart 함수를 사용하는 경우에는 cemIxIsDone 함수나 cemIxWaitDone 함수를 사용하여 모션의 완료를 확인할 수 있습니다.

□ cemIxArcA 함수를 사용하는 경우에는 내부적으로 루프를 수행하면서 모션이 완료되기를 기다리는데, 이때 블록 모드(Blocking Mode) 설정에 따라 윈도우 이벤트를 처리하는 방식이 달라집니다. 그러나 일반적으로 윈도우의 작업 쓰레드(Work Thread) 에서는 블록 모드를 사용하여, 함수 내부에서 지연 없이 쓰레드 내부의 작업에 집중할 수 있도록 설정하는 것이 바람직합니다.

□ cemIxArcA() 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.

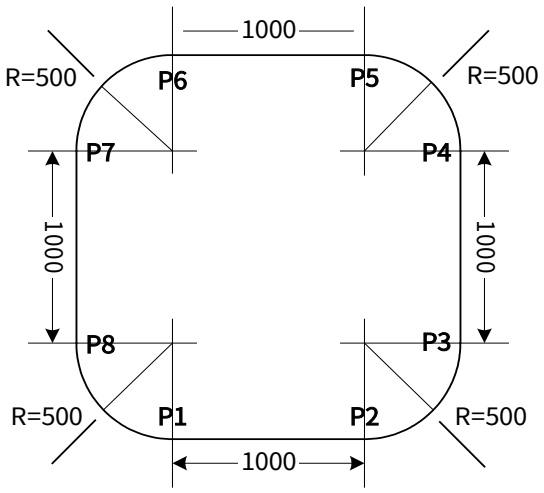
스텝 드라이버는 INP 출력이 없는 경우가 일반적인데, 고객님의 부주의나 잘못된 설정으로 INP 입력에 대한 설정이 Enable 로 되어 있을 경우 INP 입력이 스텝 드라이버를 통해 발생하지 않는 이유 때문에 모션 완료가 되지 않는 경우가 발생할 수 있습니다.

고객 여러분들께서는 스텝 드라이버 사용시에 이점을 주의해 주시기를 부탁 드립니다.

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

EXAMPLE

본 예제는 다음 그림과 같이 직선 보간 이송과 원호 보간 이송을 조합하는 Coordinated Motion 을 수행하는 예제입니다. P1 점으로부터 출발하여 P8 점을 거쳐 다시 P1 으로 복귀하는 작업입니다.



C/C++

```
#include “ceSDK.h”
#include “ceSDKDef.h”

long nIxMap = 0;                // 보간 맵 설정
long nNodeID = 1;               // 해당 원격 노드 ID 설정

void OnSetIxConfig ()
{
    // 0 번 축과 1 번 축을 보간 맵으로 설정합니다.
    cemIxMapAxes ( nIxMap, nNodeID, cemX1_MASK | cemY1_MASK, 0 );

    // 각 축의 속도를 설정합니다.
    cemCfgSpeedPattern_Set ( cemX1, cemSMODE_T, 1000, 5000, 5000 );
    cemCfgSpeedPattern_Set ( cemY1, cemSMODE_T, 1000, 5000, 5000 );
}

void OnIxArcA_Move()
{
    /* 맵에 포함된 축의 보간 마스터 속도를 설정합니다.
    작업 속도 100%, 가속도의 80%, 감속도의 80% 로 설정합니다. */
    cemIxSpeedPattern_Set ( nIxMap, CE_FALSE, cemSMODE_T, 100, 80, 80 );

    double fDistList[2];

    //P1 에서 P2 로 이동합니다.
    fDistList[0] = 1000;     fDistList[1] = 0;
    cemIxLine( nIxMap, fDistList, CE_FALSE );
```

```
    // P2 에서 P3 로 이동합니다.
    cemIxArcA( nIxMap, 0, 500, 90, CE_FALSE );

    // P3 에서 P4 로 이동합니다.
    fDistList[0] = 0;     fDistList[1] = 1000;
    cemIxLine( nIxMap, fDistList, CE_FALSE );

    // P4 에서 P5 로 이동합니다.
    cemIxArcA( nIxMap, -500, 0, 90, CE_FALSE );

    // P5 에서 P6 로 이동합니다.
    fDistList[0] = 1000;     fDistList[1] = 0;
    cemIxLine( nIxMap, fDistList, CE_FALSE );

    // P6 에서 P7 로 이동합니다.
    cemIxArcA( nIxMap, 0, -500, 90, CE_FALSE );

    // P7 에서 P8 로 이동합니다.
    fDistList[0] = 0;     fDistList[1] = -1000;
    cemIxLine( nIxMap, fDistList, CE_FALSE );

    // P8 에서 P1 로 이동합니다.
    cemIxArcA( nIxMap, 500, 0, 90, CE_FALSE );
}
```

Visual Basic

```
Dim nIxMap As Long          ' 보간 맵 번호 설정
Dim nNodeID As Long         ' 해당 원격 노드 ID 설정

nIxMap = 0
nNodeID = 1

Private Sub OnSetIxConfig ()

    ' 0 번 축과 1 번 축을 MAP0 로 설정합니다.
    Call cemIxMapAxes ( nIxMap, nNodeID, cemX1_MASK Or cemY1_MASK, 0 )

    ' 각 축의 속도를 설정합니다.
    Call cemCfgSpeedPattern_Set ( cemX1, cemSMODE_T, 1000, 5000, 5000 )
    Call cemCfgSpeedPattern_Set ( cemY1, cemSMODE_T, 1000, 5000, 5000 )

End Sub


Private Sub OnIxArcA_Move()

    Dim fDistList(2) As Double

    ' 맵에 포함된 축의 보간 마스터 속도를 설정합니다.
    ' 작업 속도 100%, 가속도의 80%, 감속도의 80% 로 설정합니다.
    Call cemIxSpeedPattern_Set ( nIxMap, CE_FALSE, cemSMODE_T, 100, 80, 80 )
```

```
    ‘ P1 에서 P2 로 이동합니다.
    fDistList(0) = 1000
    fDistList(1) = 0
    Call cemIxLine ( nIxMap, fDistList(0), CE_FALSE )

    ‘ P2 에서 P3 로 이동합니다.
    Call cemIxArcA ( nIxMap, 0, 500, 90, CE_FALSE )

    ‘ P3 에서 P4 로 이동합니다.
    fDistList(0) = 0
    fDistList(1) = 1000
    Call cemIxLine ( nIxMap, fDistList(0), CE_FALSE )

    ‘ P4 에서 P5 로 이동합니다.
    Call cemIxArcA ( nIxMap, -500, 0, 90, CE_FALSE )

    ‘ P5 에서 P6 로 이동합니다.
    fDistList(0) = 1000
    fDistList(1) = 0
    Call cemIxLine ( nIxMap, fDistList(0), CE_FALSE )

    ‘ P6 에서 P7 로 이동합니다.
    Call cemIxArcA ( nIxMap, 0, -500, 90, CE_FALSE )

    ‘ P7 에서 P8 로 이동합니다.
    fDistList(0) = 0
    fDistList(1) = -1000
    Call cemIxLine ( nIxMap, fDistList(0), CE_FALSE )

    ‘ P8 에서 P1 로 이동합니다.
    Call cemIxArcA ( nIxMap, 500, 0, 90, CE_FALSE )

End Sub
```

Delphi

```
var
    nIxMap : LongInt;           // 보간 맵 번호 설정
    nNodeID : LongInt;          // 해당 원격 노드 ID 설정
begin
    nIxMap := 0;
    nNodeID := 1;
end;

procedure OnSetIxConfig ();
begin
    // 0 번 축과 1 번 축을 MAP0 로 설정합니다.
    cemIxMapAxes ( nIxMap, nNodeID, cemX1_MASK or cemY1_MASK, 0 );

    // 각 축의 속도를 설정합니다.
    cemCfgSpeedPattern_Set ( cemX1, cemSMODE_T, 1000, 5000, 5000 );
    cemCfgSpeedPattern_Set ( cemY1, cemSMODE_T, 1000, 5000, 5000 );

end;
```

```
procedure OnIxArcA_Move();
var
    fDistList : Array[0..1] of Double;

begin
    { 맵에 포함된 축의 보간 마스터 속도를 설정합니다.
    작업 속도 100%, 가속도의 80%, 감속도의 80% 로 설정합니다. }
    cemIxSpeedPattern_Set ( nIxMap, CE_FALSE, cemSMODE_T, 100, 80, 80 );


    // P1 에서 P2 로 이동합니다.
    fDistList[0] := 1000;
    fDistList[1] := 0;
    cemIxLine ( nIxMap, @fDistList, CE_FALSE );

    // P2 에서 P3 로 이동합니다.
    cemIxArcA ( nIxMap, 0, 500, 90, CE_FALSE );

    // P3 에서 P4 로 이동합니다.
    fDistList[0] := 0;
    fDistList[1] := 1000;
    cemIxLine ( nIxMap, @fDistList, CE_FALSE );

    // P4 에서 P5 로 이동합니다.
    cemIxArcA ( nIxMap, -500, 0, 90, CE_FALSE );

    // P5 에서 P6 로 이동합니다.
    fDistList[0] := 1000;
    fDistList[1] := 0;
    cemIxLine ( nIxMap, @fDistList, CE_FALSE );

    // P6 에서 P7 로 이동합니다.
    cemIxArcA ( nIxMap, 0, -500, 90, CE_FALSE );

    // P7 에서 P8 로 이동합니다.
    fDistList[0] := 0;
    fDistList[1] := -1000;
    cemIxLine ( nIxMap, @fDistList, CE_FALSE );

    // P8 에서 P1 로 이동합니다.
    cemIxArcA ( nIxMap, 500, 0, 90, CE_FALSE );

end;
```

NAME

cemIxArcATo / cemIxArcAToStart

- 원호 보간 절대 좌표 이송

(절대적 중심 좌표와 각도)

INFORMATION	
1	Interpolation Motion
!	VC++ (6, 7, 8)/VB
	BCB/Delphi
:	Level 3
K	이송 함수

실제 이송이 진행되는 함수이므로, 사전에 반드시 안전을 확인합니다

SYNOPSIS

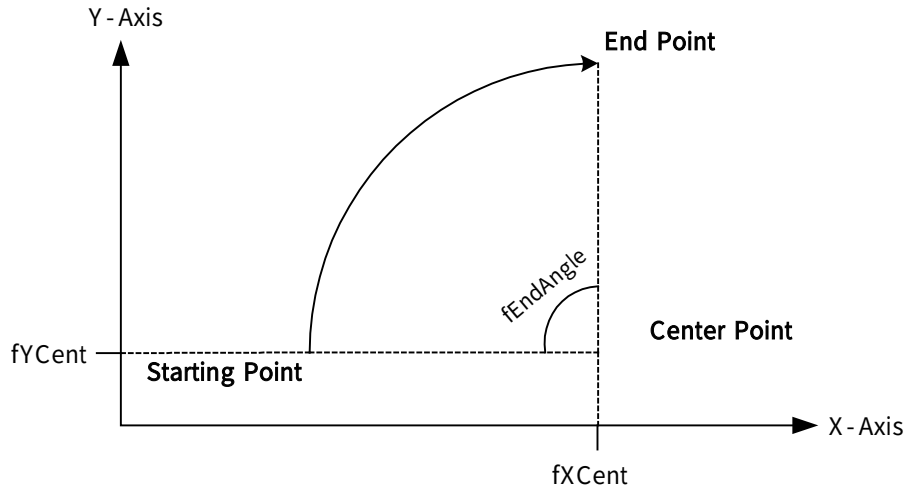
r VT_I4 cemIxArcATo ([in] VT_I4 MapIndex, [in] VT_R8 XCent, [in] VT_R8 YCent, [in] VT_R8 EndAngle, [in] VT_I4 IsBlocking)

r VT_I4 cemIxArcAToStart ([in] VT_I4 MapIndex, [in] VT_R8 XCent, [in] VT_R8 YCent, [in] VT_R8 EndAngle)

DESCRIPTION

cemIxArcATo / cemIxArcAToStart 함수는 중심 좌표와 원호의 각도를 매개변수로 하여 원호 보간 이송을 수행합니다. 이때 중심 좌표는 절대 좌표로 표현됩니다.
cemIxArcATo 함수는 모션이 완료되기 전까지 반환되지 않으며, cemIxArcAToStart 함수는 모션을 시작시킨 후에 바로 반환됩니다.

원호 보간은 임의의 두 축에 대해서 적용됩니다. 아래 설명에서는 맵핑된 두 축을 X, Y 축으로 간주하여 설명합니다. 여기서 X 축이라 함은 맵핑된 두 축 중에서 축 번호가 낮은 축을 의미하며 Y 축은 축 번호가 높은 축을 의미합니다. 예를 들어 Z 축과 U 축이 맵핑된 두 축이라면 Z 축이 X 축에 해당하며 U 축이 Y 축에 해당합니다.



PARAMETER

▶ ***MapIndex*** : 맵 번호(Map index). 이 맵 번호를 사용하기 전에 먼저 cemIxMapAxes 함수를 통하여 해당 맵 번호에 유효한 축들이 맵핑 되어 있어야 합니다.

▶ ***XCent*** : 중심점의 X 축 절대 좌표를 설정합니다.

▶ ***YCent*** : 중심점의 Y 축 절대 좌표를 설정합니다.

▶ ***EndAngle*** : 원호보간 이동을 완료할 목표 지점의 현재 위치에 대한 각도 값을 Degree(°)값으로 지정합니다. 각도의 부호가 (+)이면 반 시계방향, (-)이면 시계방향으로의 이송을 의미합니다.

▶ ***IsBlocking*** : cemIxArcATo 함수의 인자이며, 완료될 때까지 기다리는 동안 윈도우 메시지를 블록(Blocking)할 것인지를 결정합니다. 단, 쓰레드내에서 실행할 때는 이 값을 1(CE_TRUE)로 설정해 주어야 합니다.

Value	Meaning
0 (CE_FALSE)	윈도우 메시지를 블록(Blocking) 하지 않습니다. 따라서 해당 모션이 진행되는 동안 윈도우 이벤트를 처리합니다.
1 (CE_TRUE)	윈도우 메시지를 블록(Blocking) 합니다. 따라서 해당 모션이 완료되는 동안에는 윈도우 이벤트가 처리되지 않습니다.

RETURN VALUE

Value	Meaning
음수	수행 실패. 자세한 내용은 '에러처리' 편을 참고합니다.
0 (ceERR_NONE)	수행 성공.

SEE ALSO

cemIxArcA, cemIxArcAStart

REFERENCE

□ cemIxArcAToStart 함수를 사용하는 경우에는 cemIxIsDone 함수나 cemIxWaitDone 함수를 사용하여 모션의 완료를 확인할 수 있습니다.

□ cemIxArcATo 함수를 사용하는 경우에는 내부적으로 루프를 수행하면서 모션이 완료되기를 기다리는데, 이때 블록 모드(Blocking Mode)설정에 따라 윈도우 이벤트를 처리하는 방식이 달라집니다. 그러나 일반적으로 윈도우의 작업 쓰레드(Work Thread) 에서는 블록 모드를 사용하여, 함수 내부에서 지연 없이 쓰레드 내부의 작업에 집중할 수 있도록 설정하는 것이 바람직합니다.

□ cemIxArcATo 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

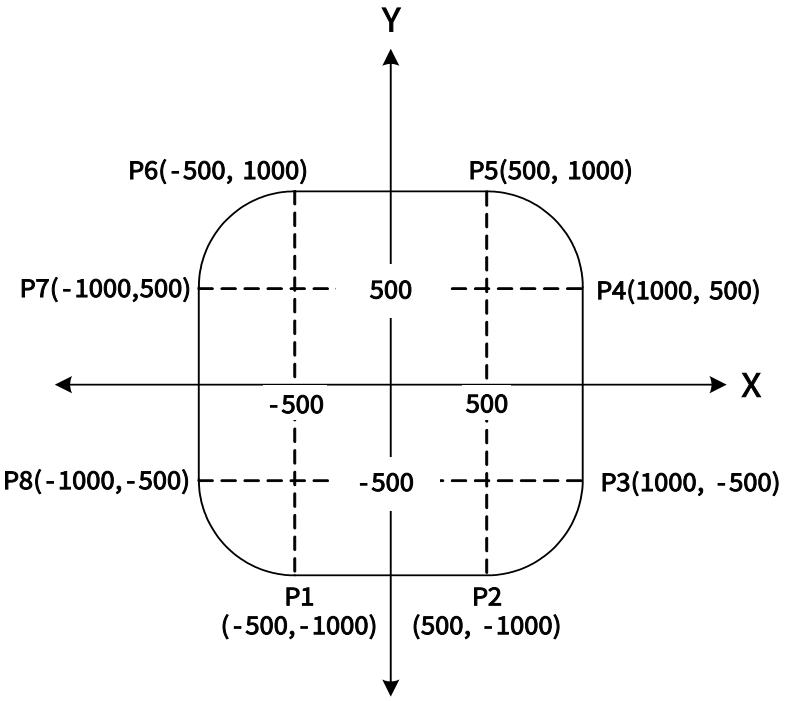
스텝 드라이버는 INP 출력이 없는 경우가 일반적인데, 고객님의 부주의나 잘못된 설정으로 INP 입력에 대한 설정이 Enable 로 되어 있을 경우 INP 입력이 스텝 드라이버를 통해 발생하지 않는 이유 때문에 모션 완료가 되지 않는 경우가 발생할 수 있습니다.

고객 여러분들께서는 스텝 드라이버 사용시에 이점을 주의해 주시기를 부탁 드립니다.

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

EXAMPLE

본 예제는 다음 그림과 같이 직선 보간 이송과 원호 보간 이송을 조합하는 Coordinated Motion 을 수행하는 예제입니다. P1 점으로부터 출발하여 P8 점을 거쳐 다시 P1 으로 복귀하는 작업입니다. 그리고 현재 위치가 P1 의 위치에 있다고 가정합니다.



C/C++

```
#include “ceSDK.h”
#include “ceSDKDef.h”

long nIxMap = 0;              // 보간 맵 설정
long nNodeID = 1;             // 해당 원격 노드 ID 설정

void OnSetIxConfig ()
{
    // 0 번 축과 1 번 축을 보간 맵으로 설정합니다.
    cemIxMapAxes ( nIxMap, nNodeID, cemX1_MASK | cemY1_MASK, 0 );

    // 각 축의 속도를 설정합니다.
    cemCfgSpeedPattern_Set ( cemX1, cemSMODE_T, 1000, 5000, 5000 );
    cemCfgSpeedPattern_Set ( cemY1, cemSMODE_T, 1000, 5000, 5000 );
}


void OnIxArcATo_Move ()
{
    // 맵에 포함된 축의 속도 비율을 설정합니다.
```

```
    cemIxSpeedPattern_Set ( nIxMap, CE_FALSE, cemSMODE_T, 100, 70, 70 );

    double fPosList[2] = {0.0f, 0.0f};

    // P1 에서 P2 로 이동합니다.
    fPosList[0] = 500;     fPosList[1] = -1000;
    cemIxLineTo ( nIxMap, fPosList, CE_FALSE );

    // P2 에서 P3 로 이동합니다.
    cemIxArcATo ( nIxMap, 500, -500, 90, CE_FALSE );

    // P3 에서 P4 로 이동합니다.
    fPosList[0] = 1000;     fPosList[1] = 500;
    cemIxLineTo ( nIxMap, fPosList, CE_FALSE );

    // P4 에서 P5 로 이동합니다.
    cemIxArcATo ( nIxMap, 500, 500, 90, CE_FALSE );

    // P5 에서 P6 로 이동합니다.
    fPosList[0] = -500;     fPosList[1] = 1000;
    cemIxLineTo ( nIxMap, fPosList, CE_FALSE );

    // P6 에서 P7 로 이동합니다.
    cemIxArcATo ( nIxMap, -500, 500, 90, CE_FALSE );

    // P7 에서 P8 로 이동합니다.
    fPosList[0] = -1000;     fPosList[1] = -500;
    cemIxLineTo ( nIxMap, fPosList, CE_FALSE );

    //P8 에서 P1 로 이동합니다.
    cemIxArcATo ( nIxMap, -500, -500, 90, CE_FALSE );
}
```

```
Visual Basic

Dim nIxMap As Long          ' 보간 맵 설정
Dim nNodeID As Long         ' 해당 원격 노드 ID 설정

nIxMap = 0
nNodeId = 1

Private Sub OnSetIxConfig ()

    ' 0 번 축과 1 번 축을 보간 맵으로 설정합니다.
    Call cemIxMapAxes ( nIxMap, nNodeID, cemX1_MASK Or cemY1_MASK, 0 )

    ' 각 축의 속도를 설정합니다.
    Call cemCfgSpeedPattern_Set ( cemX1, cemSMODE_T, 1000, 5000, 5000 )
    Call cemCfgSpeedPattern_Set ( nAxis1, cemSMODE_T, 1000, 5000, 5000 )

End Sub


Private Sub OnIxArcATo_Move ()
```

```
    Dim fPosList(2) As Double

    ‘ 맵에 포함된 축의 속도 비율을 설정합니다.
    Call cemIxSpeedPattern_Set ( nIxMap, CE_FALSE, cemSMODE_T, 100, 70, 70 )

    ‘ P1 에서 P2 로 이동합니다.
    fPosList(0) = 500
    fPosList(1) = -1000
    Call cemIxLineTo ( nIxMap, fPosList(0), CE_FALSE )

    ‘ P2 에서 P3 로 이동합니다.
    Call cemIxArcATo ( nIxMap, 500, -500, 90, CE_FALSE )

    ‘ P3 에서 P4 로 이동합니다.
    fPosList(0) = 1000
    fPosList(1) = 500
    Call cemIxLineTo ( nIxMap, fPosList(0), CE_FALSE )

    ‘ P4 에서 P5 로 이동합니다.
    Call cemIxArcATo ( nIxMap, 500, 500, 90, CE_FALSE )

    ‘ P5 에서 P6 로 이동합니다.
    fPosList(0) = -500
    fPosList(1) = 1000
    Call cemIxLineTo ( nIxMap, fPosList(0), CE_FALSE )

    ‘ P6 에서 P7 로 이동합니다.
    Call cemIxArcATo ( nIxMap, -500, 500, 90, CE_FALSE )

    ‘ P7 에서 P8 로 이동합니다.
    fPosList(0) = -1000
    fPosList(1) = -500
    Call cemIxLineTo ( nIxMap, fPosList(0), CE_FALSE )

    ‘ P8 에서 P1 로 이동합니다.
    Call cemIxArcATo ( nIxMap, -500, -500, 90, CE_FALSE )

End Sub
```

```
Delphi

var
    nIxMap : LongInt;           // 보간 맵 설정
    nNodeID : LongInt;          // 해당 원격 노드 ID 설정
begin
    nIxMap := 0;
    nNodeId := 1;
end;

procedure OnSetIxConfig ();
begin
    // 0 번 축과 1 번 축을 보간 맵으로 설정합니다.
    cemIxMapAxes ( nIxMap, nNodeID, cemX1_MASK or cemY1_MASK, 0 );

    // 각 축의 속도를 설정합니다.
```

```
    cemCfgSpeedPattern_Set ( cemX1, cemSMODE_T, 1000, 5000, 5000 );
    cemCfgSpeedPattern_Set ( cemY1, cemSMODE_T, 1000, 5000, 5000 );

end;


procedure OnIxArcATo_Move ()
ver
    fPosList : Array[0..1] of Double;

begin
    // 맵에 포함된 축의 속도 비율을 설정합니다.
    cemIxSpeedPattern_Set ( nIxMap, CE_FALSE, cemSMODE_T, 100, 70, 70 );

    // P1 에서 P2 로 이동합니다.
    fPosList[0] := 500;
    fPosList[1] := -1000;
    cemIxLineTo ( nIxMap, @fPosList, CE_FALSE );

    // P2 에서 P3 로 이동합니다.
    cemIxArcATo ( nIxMap, 500, -500, 90, CE_FALSE );

    // P3 에서 P4 로 이동합니다.
    fPosList[0] := 1000;
    fPosList[1] := 500;
    cemIxLineTo ( nIxMap, @fPosList, CE_FALSE );

    // P4 에서 P5 로 이동합니다.
    cemIxArcATo ( nIxMap, 500, 500, 90, CE_FALSE );

    // P5 에서 P6 로 이동합니다.
    fPosList[0] := -500;
    fPosList[1] := 1000;
    cemIxLineTo ( nIxMap, @fPosList, CE_FALSE );

    // P6 에서 P7 로 이동합니다.
    cemIxArcATo ( nIxMap, -500, 500, 90, CE_FALSE );

    // P7 에서 P8 로 이동합니다.
    fPosList[0] := -1000;
    fPosList[1] := -500;
    cemIxLineTo ( nIxMap, @fPosList, CE_FALSE );

    // P8 에서 P1 로 이동합니다.
    cemIxArcATo ( nIxMap, -500, -500, 90, CE_FALSE );

end;
```

NAME

cemIxArcP / cemIxArcPStart

- 원호 보간 상대 좌표 이송

(상대적 중심 좌표와 종점 좌표)

INFORMATION
1 Interpolation Motion
! VC++ (6, 7, 8)/VB
BCB/Delphi
: Level 3
K 이송 함수
실제 이송이 진행되는 함수이므로, 사전에 반드시 안전을 확인합니다.

SYNOPSIS

r VT_I4 cemIxArcP ([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)

r VT_I4 cemIxArcPStart ([in] VT_I4 MapIndex, [in] VT_R8 XCentOffset,

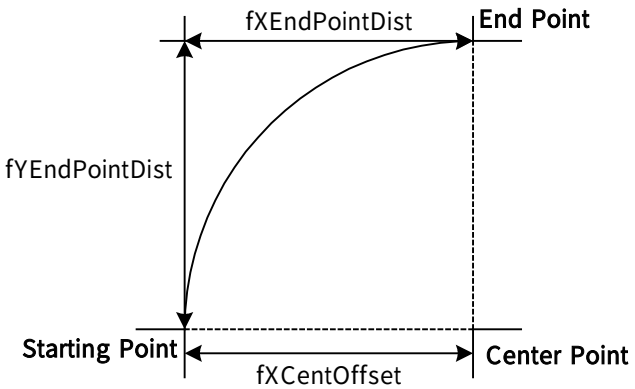
[in] VT_R8 YCentOffset, [in] VT_R8 XEndPointDist, [in] VT_R8 YEndPointDist,

[in] VT_I4 Direction)

DESCRIPTION

cemIxArcP / cemIxArcPStart 함수는 중심 좌표와 종점 좌표를 매개변수로 하여 원호 보간 이송을 수행합니다. 이때 각 좌표는 상대 좌표로 표현됩니다. cemIxArcP 함수는 모션이 완료되기 전까지 반환되지 않으며, cemIxArcPStart 함수는 모션을 시작시킨 후에 바로 반환됩니다.

원호 보간은 임의의 두 축에 대해서 적용됩니다. 아래 설명에서는 맵핑된 두 축을 X, Y 축으로 간주하여 설명합니다. 여기서 X 축이라 함은 맵핑된 두 축 중에서 축 번호가 낮은 축을 의미하며 Y 축은 축 번호가 높은 축을 의미합니다. 예를 들어 Z 축과 U 축이 맵핑된 두 축이라면 Z 축이 X 축에 해당하며 U 축이 Y 축에 해당합니다.



PARAMETER

▶ *MapIndex* : 맵 번호(Map index). 이 맵 번호를 사용하기 전에 먼저 cemIxMapAxes 함수를 통하여 해당 맵 번호에 유효한 축들이 맵핑 되어 있어야 합니다.

▶ *XCentOffset* : 현재 위치(시작 위치)로부터 원의 중심까지 X 축상의 거리를 설정합니다.

▶ *YCentOffset* : 현재 위치(시작 위치)로부터 원의 중심까지 Y 축상의 거리를 설정합니다.

▶ *XEndPointDist* : 원호 보간 이송을 완료할 목표 지점의 현재 위치로부터 X 축상의 거리를 설정합니다.

▶ *YEndPointDist* : 원호 보간 이송을 완료할 목표 지점의 현재 위치로부터 Y 축상의 거리를 설정합니다.

▶ *Direction* : 회전 방향을 지정합니다.

Value	Meaning
0 (cemARC_CW)	시계 방향(CW)으로 회전.
1 (cemARC_CCW)	반 시계 방향(CCW)으로 회전.

▶ *IsBlocking* : cemIxArcP 함수의 인자이며, 완료될 때까지 기다리는 동안 윈도우 메시지를 블록(Blocking)할 것인지를 결정합니다. 단, 쓰레드내에서 실행할 때는 이 값을 1(CE_TRUE)로 설정해 주어야 합니다.

Value	Meaning
0 (CE_FALSE)	윈도우 메시지를 블록(Blocking) 하지 않습니다. 따라서 해당 모션이 진행되는 동안 윈도우 이벤트를 처리합니다.
1 (CE_TRUE)	윈도우 메시지를 블록(Blocking) 합니다. 따라서 해당 모션이 완료되는 동안에는 윈도우 이벤트가 처리되지 않습니다.

RETURN VALUE

Value	Meaning
음수	수행 실패. 자세한 내용은 '에러처리' 편을 참고합니다.
0 (ceERR_NONE)	수행 성공.

SEE ALSO

cemIxArcPTo, cemIxArcPToStart

REFERENCE

□ cemIxArcPStart 함수를 사용하는 경우에는 cemIxIsDone 함수나 cemIxWaitDone 함수를 사용하여 모션의 완료를 확인할 수 있습니다.

□ cemIxArcP 함수를 사용하는 경우에는 내부적으로 루프를 수행하면서 모션이 완료되기를 기다리는데, 이때 블록 모드(Blocking Mode) 설정에 따라 윈도우 이벤트를 처리하는 방식이 달라집니다. 그러나 일반적으로 윈도우의 작업 쓰레드(Work Thread) 에서는 블록 모드를 사용하여, 함수 내부에서 지연 없이 쓰레드 내부의 작업에 집중할 수 있도록 설정하는 것이 바람직합니다.

□ cemIxArcP 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.

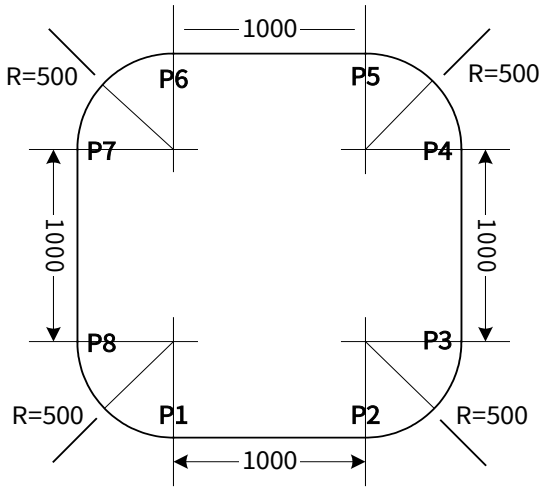
스텝 드라이버는 INP 출력이 없는 경우가 일반적인데, 고객님의 부주의나 잘못된 설정으로 INP 입력에 대한 설정이 Enable 로 되어 있을 경우 INP 입력이 스텝 드라이버를 통해 발생하지 않는 이유 때문에 모션 완료가 되지 않는 경우가 발생할 수 있습니다.

고객 여러분들께서는 스텝 드라이버 사용시에 이점을 주의해 주시기를 부탁 드립니다.

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

EXAMPLE

본 예제는 다음 그림과 같이 직선 보간 이송과 원호 보간 이송을 조합하는 Coordinated Motion 을 수행하는 예제입니다. P1 점으로부터 출발하여 P8 점을 거쳐 다시 P1 으로 복귀하는 작업입니다.



C/C++

```
#include “ceSDK.h”
#include “ceSDKDef.h”

long nIxMap = 0;                // 보간 맵 설정
long nNodeID = 1;               // 해당 원격 노드 ID

void OnSetIxConfig ()
{
    // 0 번 축과 1 번 축을 보간 맵으로 설정합니다.
    cemIxMapAxes ( nIxMap, nNodeID, cemX1_MASK | cemY1_MASK, 0 );

    // 각 축의 속도를 설정합니다.
    cemCfgSpeedPattern_Set ( cemX1, cemSMODE_T, 1000, 5000, 5000 );
    cemCfgSpeedPattern_Set ( cemY1, cemSMODE_T, 1000, 5000, 5000 );
}


void OnIxArcP_Move ()
{
    // 맵에 포함된 축의 속도 비율을 설정합니다.
    cemIxSpeedPattern_Set ( nIxMap, CE_FALSE, cemSMODE_T, 100, 70, 70 );

    double fDistList[2] = {0.0f, 0.0f};
```

```
    // P1 에서 P2 로 이송합니다.
    fDistList[0] = 1000;     fDistList[1] = 0;
    cemIxLine ( nIxMap, fDistList, CE_FALSE );

    // P2 에서 P3 로 이송합니다.
    cemIxArcP ( nIxMap, 0, 500, 500, 500, cemARC_CCW, CE_FALSE );

    // P3 에서 P4 로 이송합니다.
    fDistList[0] = 0;     fDistList[1] = 1000;
    cemIxLine ( nIxMap, fDistList, CE_FALSE );

    // P4 에서 P5 로 이송합니다.
    cemIxArcP ( nIxMap, -500, 0, -500, 500, cemARC_CCW, CE_FALSE );

    // P5 에서 P6 로 이송합니다.
    fDistList[0] = -1000;     fDistList[1] = 0;
    cemIxLine ( nIxMap, fDistList, CE_FALSE );

    // P6 에서 P7 로 이송합니다.
    cemIxArcP ( nIxMap, 0, -500, -500, -500, cemARC_CCW, CE_FALSE );

    // P7 에서 P8 로 이송합니다.
    fDistList[0] = 0;     fDistList[1] = -1000;
    cemIxLine ( nIxMap, fDistList, CE_FALSE );

    // P8 에서 P1 로 이송합니다.
    cemIxArcP ( nIxMap, 500, 0, 500, -500, cemARC_CCW, CE_FALSE );
}
```

```
Visual Basic

Dim nIxMap As Long          ' 보간 맵 설정
Dim nNodeID As Long         ' 해당 원격 노드 ID

nIxMap = 0
nNodeID = 1

Private Sub OnSetIxConfig ()

    ' 0 번 축과 1 번 축을 보간 맵으로 설정합니다.
    Call cemIxMapAxes ( nIxMap, nNodeID, cemX1_MASK Or cemY1_MASK, 0 )

    ' 각 축의 속도를 설정합니다.
    Call cemCfgSpeedPattern_Set ( cemX1, cemSMODE_T, 1000, 5000, 5000 )
    Call cemCfgSpeedPattern_Set ( cemY1, cemSMODE_T, 1000, 5000, 5000 )

End Sub


Private Sub OnIxArcP_Move ()

    Dim fDistList(2) As Double

    ' 맵에 포함된 축의 속도 비율을 설정합니다.
    Call cemIxSpeedPattern_Set ( nIxMap, CE_FALSE, cemSMODE_T, 100, 70, 70 )
```

```
    ‘ P1 에서 P2 로 이송합니다.
    fDistList(0) = 1000
    fDistList(1) = 0
    Call cemIxLine ( nIxMap, fDistList(0), CE_FALSE )

    ‘  P2 에서 P3 로 이송합니다.
    Call cemIxArcP ( nIxMap, 0, 500, 500, 500, cemARC_CCW, CE_FALSE )

    ‘ P3 에서 P4 로 이송합니다.
    fDistList(0) = 0
    fDistList(1) = 1000
    Call cemIxLine ( nIxMap, fDistList(0), CE_FALSE )

    ‘ P4 에서 P5 로 이송합니다.
    Call cemIxArcP ( nIxMap, -500, 0, -500, 500, cemARC_CCW, CE_FALSE )

    ‘ P5 에서 P6 로 이송합니다.
    fDistList(0) = -1000
    fDistList(1) = 0
    Call cemIxLine ( nIxMap, fDistList(0), CE_FALSE )

    ‘ P6 에서 P7 로 이송합니다.
    Call cemIxArcP ( nIxMap, 0, -500, -500, -500, cemARC_CCW, CE_FALSE )

    ‘ P7 에서 P8 로 이송합니다.
    fDistList(0) = 0
    fDistList(1) = -1000
    Call cemIxLine ( nIxMap, fDistList(0), CE_FALSE )

    ‘ P8 에서 P1 로 이송합니다.
    Call cemIxArcP ( nIxMap, 500, 0, 500, -500, cemARC_CCW, CE_FALSE )

End Sub
```

```
Delphi

var
    nIxMap : LongInt;           // 보간 맵 설정
    nNodeID : LongInt;          // 해당 원격 노드 ID 설정
begin
    nIxMap := 0;
    nNodeId := 1;
end;


procedure OnSetIxConfig ();
begin
    // 0 번 축과 1 번 축을 보간 맵으로 설정합니다.
    cemIxMapAxes ( nIxMap, nNodeID, cemX1_MASK or cemY1_MASK, 0 );

    // 각 축의 속도를 설정합니다.
    cemCfgSpeedPattern_Set ( cemX1, cemSMODE_T, 1000, 5000, 5000 );
    cemCfgSpeedPattern_Set ( cemY1, cemSMODE_T, 1000, 5000, 5000 );
```

```
end;


procedure OnIxArcP_Move ();
var
    fDistList : Array[0..1] of Double;

begin
    // 맵에 포함된 축의 속도 비율을 설정합니다.
    cemIxSpeedPattern_Set ( nIxMap, CE_FALSE, cemSMODE_T, 100, 70, 70 );

    // P1 에서 P2 로 이송합니다.
    fDistList[0] := 1000;
    fDistList[1] := 0;
    cemIxLine ( nIxMap, @fDistList, CE_FALSE );

    //  P2 에서 P3 로 이송합니다.
    cemIxArcP ( nIxMap, 0, 500, 500, 500, cemARC_CCW, CE_FALSE );

    // P3 에서 P4 로 이송합니다.
    fDistList[0] := 0;
    fDistList[1] := 1000;
    cemIxLine ( nIxMap, @fDistList, CE_FALSE );

    // P4 에서 P5 로 이송합니다.
    cemIxArcP ( nIxMap, -500, 0, -500, 500, cemARC_CCW, CE_FALSE );

    // P5 에서 P6 로 이송합니다.
    fDistList[0] := -1000;
    fDistList[1] := 0;
    cemIxLine ( nIxMap, @fDistList, CE_FALSE );

    // P6 에서 P7 로 이송합니다.
    cemIxArcP ( nIxMap, 0, -500, -500, -500, cemARC_CCW, CE_FALSE );

    // P7 에서 P8 로 이송합니다.
    fDistList[0] := 0;
    fDistList[1] := -1000;
    cemIxLine ( nIxMap, @fDistList, CE_FALSE );

    // P8 에서 P1 로 이송합니다.
    cemIxArcP ( nIxMap, 500, 0, 500, -500, cemARC_CCW, CE_FALSE );

end;
```

NAME

cemIxArcPTo / cemIxArcPToStart

- 원호 보간 절대 좌표 이송

(절대적 중심 좌표와 종점 좌표)

INFORMATION	
1	Interpolation Motion
!	VC++ (6, 7, 8)/VB
	BCB/Delphi
:	Level 3
K	이송 함수

실제 이송이 진행되는 함수이므로, 사전에 반드시 안전을 확인합니다.

SYNOPSIS

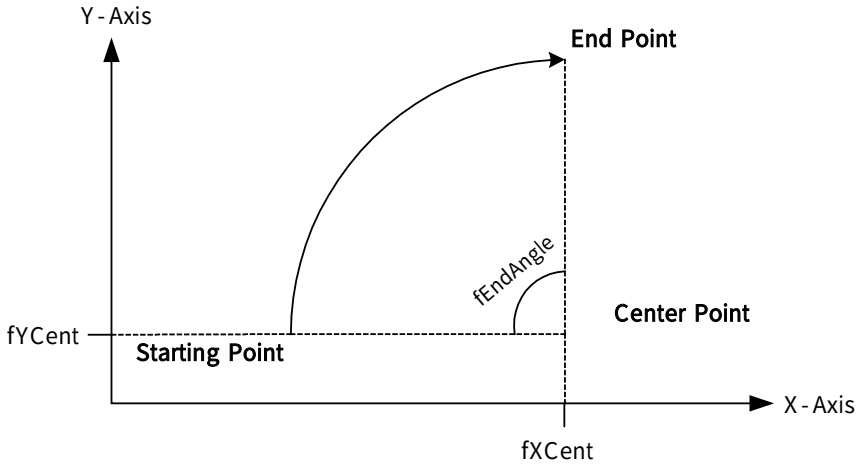
r VT_I4 cemIxArcPTo ([in] VT_I4 MapIndex, [in] VT_R8 XCent, [in] VT_R8 YCent, [in] VT_R8 XEndPos, [in] VT_R8 YEndPos, [in] VT_I4 Direction, [in] VT_I4 IsBlocking)

r VT_I4 cemIxArcPToStart ([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

cemIxArcPTo /cemIxArcPToStart 함수는 중심 좌표와 종점 좌표를 매개변수로 하여 원호 보간 이송을 수행합니다. 이때 각 좌표는 절대 좌표로 표현됩니다.
cemIxArcPTo 함수는 모션이 완료되기 전까지 반환되지 않으며, cemIxArcPToStart 함수는 모션을 시작시킨 후에 바로 반환됩니다.

원호 보간은 임의의 두 축에 대해서 적용됩니다. 아래 설명에서는 맵핑된 두 축을 X, Y 축으로 간주하여 설명합니다. 여기서 X 축이라 함은 맵핑된 두 축 중에서 축 번호가 낮은 축을 의미하며 Y 축은 축 번호가 높은 축을 의미합니다. 예를 들어 Z 축과 U 축이 맵핑된 두 축이라면 Z 축이 X 축에 해당하며 U 축이 Y 축에 해당합니다.



PARAMETER

▶ *MapIndex* : 맵 번호(Map index). 이 맵 번호를 사용하기 전에 먼저 cemIxMapAxes 함수를 통하여 해당 맵 번호에 유효한 축들이 맵핑 되어 있어야 합니다.

▶ *XCent* : 중심점의 X 축 절대 좌표 값을 설정합니다.

▶ *YCent* : 중심점의 Y 축 절대 좌표 값을 설정합니다.

▶ *XEndPos* : 원호 보간 이송을 완료할 목표 지점(End point)의 X 축 절대 좌표 값을 설정합니다.

▶ *YEndPos* : 원호 보간 이송을 완료할 목표 지점(End point)의 Y 축 절대 좌표 값을 설정합니다.

▶ *Direction* : 회전 방향을 설정합니다.

Value	Meaning
0 (cemARC_CW)	시계 방향(CW)으로 회전.
1 (cemARC_CCW)	반 시계 방향(CCW)으로 회전.

▶ *IsBlocking* : cemIxArcPTo 함수의 인자이며, 완료될 때까지 기다리는 동안 윈도우 메시지를 블록(Blocking)할 것인지를 결정합니다. 단, 쓰레드내에서 실행할 때는 이 값을 1(CE_TRUE)로 설정해 주어야 합니다.

Value	Meaning
0 (CE_FALSE)	윈도우 메시지를 블록(Blocking) 하지 않습니다. 따라서 해당 모션이 진행되는 동안 윈도우 이벤트를 처리합니다.
1 (CE_TRUE)	윈도우 메시지를 블록(Blocking) 합니다. 따라서 해당 모션이 완료되는 동안에는 윈도우 이벤트가 처리되지 않습니다.

RETURN VALUE

Value	Meaning
음수	수행 실패. 자세한 내용은 '에러처리' 편을 참고합니다.
0 (ceERR_NONE)	수행 성공.

SEE ALSO

cemIxArcP, cemIxArcPStart

REFERENCE

□ cemIxArcPToStart 함수를 사용하는 경우에는 cemIxIsDone 함수나 cemIxWaitDone 함수를 사용하여 모션의 완료를 확인할 수 있습니다.

□ cemIxArcPTo 함수를 사용하는 경우에는 내부적으로 루프를 수행하면서 모션이 완료되기를 기다리는데, 이때 블록 모드(Blocking Mode) 설정에 따라 윈도우 이벤트를 처리하는 방식이 달라집니다. 그러나 일반적으로 윈도우의 작업 쓰레드(Work Thread) 에서는 블록 모드를 사용하여, 함수 내부에서 지연 없이 쓰레드 내부의 작업에 집중할 수 있도록 설정하는 것이 바람직합니다.

□ cemIxArcPTo 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.

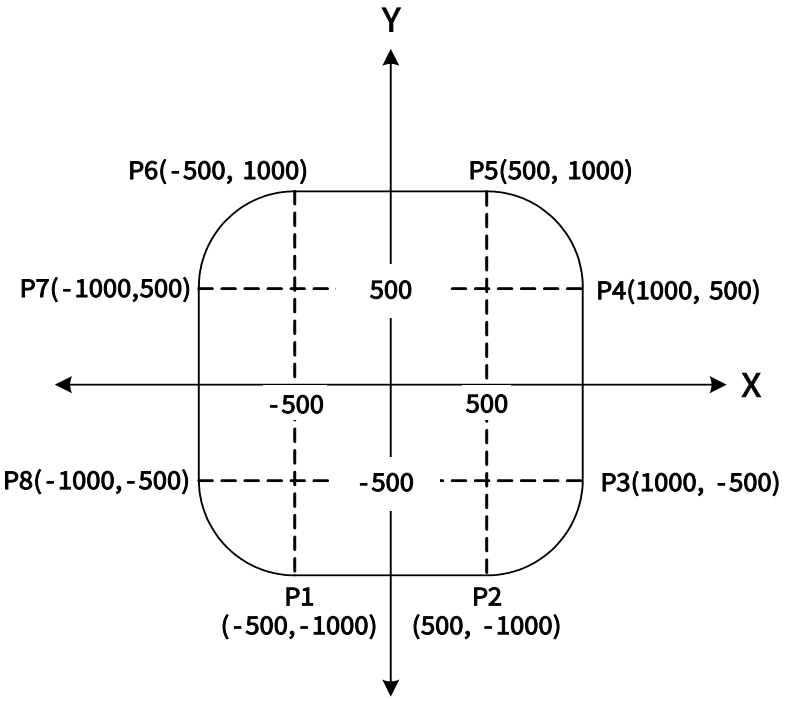
스텝 드라이버는 INP 출력이 없는 경우가 일반적인데, 고객님의 부주의나 잘못된 설정으로 INP 입력에 대한 설정이 Enable 로 되어 있을 경우 INP 입력이 스텝 드라이버를 통해 발생하지 않는 이유 때문에 모션 완료가 되지 않는 경우가 발생할 수 있습니다.

고객 여러분들께서는 스텝 드라이버 사용시에 이점을 주의해 주시기를 부탁 드립니다.

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

EXAMPLE

본 예제는 다음 그림과 같이 직선 보간 이송과 원호 보간 이송을 조합하는 Coordinated Motion 을 수행하는 예제입니다. P1 점으로부터 출발하여 P8 점을 거쳐 다시 P1 으로 복귀하는 작업입니다. 그리고 현재 위치가 P1 의 위치에 있다고 가정합니다.



C/C++

```
#include “ceSDK.h”
#include “ceSDKDef.h”

long nIxMap = 0;              // 보간 맵 설정
long nNodeID = 1;             // 해당 원격 노드 ID

void OnSetIxConfig ()
{
    // 0 번 축과 1 번 축을 보간 맵으로 설정합니다.
    cemIxMapAxes ( nIxMap, nNodeID, cemX1_MASK | cemY1_MASK, 0 );

    //각 축의 속도를 설정합니다.
    cemCfgSpeedPattern_Set ( cemX1, cemSMODE_T, 1000, 5000, 5000 );
    cemCfgSpeedPattern_Set ( cemY1, cemSMODE_T, 1000, 5000, 5000 );
}


void OnIxArcPTo_Move ()
{
    // 맵에 포함된 축의 속도 비율을 설정합니다.
    cemIxSpeedPattern_Set ( nIxMap, CE_FALSE, cemSMODE_T, 100, 70, 70 );
```

```
    double fPosList[2] = {0.0f, 0.0f};

    // P1 에서 P2 로 이동합니다.
    fPosList[0] = 500;     fPosList[1] = -1000;
    cemIxLineTo ( nIxMap, fPosList, CE_FALSE );

    // P2 에서 P3 로 이송합니다.
    cemIxArcPTo ( nIxMap, 500, -500, 1000, -500, cemARC_CCW, CE_FALSE );

    // P3 에서 P4 로 이송합니다.
    fPosList[0] = 1000;     fPosList[1] = 500;
    cemIxLineTo ( nIxMap, fPosList, CE_FALSE );

    // P4 에서 P5 로 이송합니다.
    cemIxArcPTo( nIxMap, 500, 500, 500, 1000, cemARC_CCW, CE_FALSE );

    // P5 에서 P6 로 이송합니다.
    fPosList[0] = -500;     fPosList[1] = 1000;
    cemIxLineTo ( nIxMap, fPosList, CE_FALSE );

    // P6 에서 P7 로 이송합니다.
    cemIxArcPTo ( nIxMap, -500, 500, -1000, 500, cemARC_CCW, CE_FALSE );

    // P7 에서 P8 로 이송합니다.
    fPosList[0] = -1000;     fPosList[1] = -500;
    cemIxLineTo ( nIxMap, fPosList, CE_FALSE );

    // P8 에서 P1 로 이송합니다.
    cemIxArcPTo ( nIxMap, -500, -500, -500, -1000, cemARC_CCW, CE_FALSE );
}
```

Visual Basic

```
Dim nIxMap As Long          ' 보간 맵 설정
Dim nNodeID As Long         ' 해당 원격 노드 ID

nIxMap = 0
nNodeID = 1

Private Sub OnSetIxConfig ()

    ' 0 번 축과 1 번 축을 보간 맵으로 설정합니다.
    Call cemIxMapAxes ( nIxMap, nNodeID, cemX1_MASK Or cemY1_MASK, 0 )

    ' 각 축의 속도를 설정합니다.
    Call cemCfgSpeedPattern_Set ( cemX1, cemSMODE_T, 1000, 5000, 5000 )
    Call cemCfgSpeedPattern_Set ( cemY1, cemSMODE_T, 1000, 5000, 5000 )

End Sub


Private Sub OnIxArcPTo_Move ()

    Dim fPosList(2) As Double
    ' 맵에 포함된 축의 속도 비율을 설정합니다.
```

```
    Call cemIxSpeedPattern_Set ( nIxMap, CE_FALSE, cemSMODE_T, 100, 70, 70 )

    ‘ P1 에서 P2 로 이동합니다.
    fPosList(0) = 500
    fPosList(1) = -1000
    Call cemIxLineTo ( nIxMap, fPosList(0), CE_FALSE )

    ‘ P2 에서 P3 로 이송합니다.
    Call cemIxArcPTo ( nIxMap, 500, -500, 1000, -500, cemARC_CCW, CE_FALSE )

    ‘ P3 에서 P4 로 이송합니다.
    fPosList(0) = 1000
    fPosList(1) = 500
    Call cemIxLineTo ( nIxMap, fPosList(0), CE_FALSE )

    ‘ P4 에서 P5 로 이송합니다.
    Call cemIxArcPTo ( nIxMap, 500, 500, 500, 1000, cemARC_CCW, CE_FALSE )

    ‘ P5 에서 P6 로 이송합니다.
    fPosList(0) = -500
    fPosList(1) = 1000
    Call cemIxLineTo ( nIxMap, fPosList(0), CE_FALSE )

    ‘ P6 에서 P7 로 이송합니다.
    Call cemIxArcPTo ( nIxMap, -500, 500, -1000, 500, cemARC_CCW, CE_FALSE )

    ‘ P7 에서 P8 로 이송합니다.
    fPosList(0) = -1000
    fPosList(1) = -500
    Call cemIxLineTo ( nIxMap, fPosList(0), CE_FALSE )

    ‘ P8 에서 P1 로 이송합니다.
    Call cemIxArcPTo ( nIxMap, -500, -500, -500, -1000, cemARC_CCW, CE_FALSE )

End Sub
```

```
Delphi

var
    nIxMap : LongInt;           // 보간 맵 설정
    nNodeID : LongInt;          // 해당 원격 노드 ID 설정
begin
    nIxMap := 0;
    nNodeId := 1;
end;


procedure OnSetIxConfig ();
begin

    // 0 번 축과 1 번 축을 보간 맵으로 설정합니다.
    cemIxMapAxes ( nIxMap, nNodeID, cemX1_MASK or cemY1_MASK, 0 );;

    // 각 축의 속도를 설정합니다.
    cemCfgSpeedPattern_Set ( cemX1, cemSMODE_T, 1000, 5000, 5000 );
```

```
    cemCfgSpeedPattern_Set ( cemY1, cemSMODE_T, 1000, 5000, 5000 );

end;


procedure OnIxArcPTo_Move ();
var
    fPosList : Array[0..1] of Double;

begin
    // 맵에 포함된 축의 속도 비율을 설정합니다.
    cemIxSpeedPattern_Set ( nIxMap, CE_FALSE, cemSMODE_T, 100, 70, 70 );

    // P1 에서 P2 로 이동합니다.
    fPosList[0] := 500;
    fPosList[1] := -1000;
    cemIxLineTo ( nIxMap, @fPosList, CE_FALSE );

    // P2 에서 P3 로 이송합니다.
    cemIxArcPTo ( nIxMap, 500, -500, 1000, -500, cemARC_CCW, CE_FALSE );

    // P3 에서 P4 로 이송합니다.
    fPosList[0] := 1000;
    fPosList[1] := 500;
    cemIxLineTo ( nIxMap, @fPosList, CE_FALSE );

    // P4 에서 P5 로 이송합니다.
    cemIxArcPTo ( nIxMap, 500, 500, 500, 1000, cemARC_CCW, CE_FALSE );

    // P5 에서 P6 로 이송합니다.
    fPosList[0] := -500;
    fPosList[1] := 1000;
    cemIxLineTo ( nIxMap, @fPosList, CE_FALSE );

    // P6 에서 P7 로 이송합니다.
    cemIxArcPTo ( nIxMap, -500, 500, -1000, 500, cemARC_CCW, CE_FALSE );

    // P7 에서 P8 로 이송합니다.
    fPosList[0] := -1000;
    fPosList[1] := -500;
    cemIxLineTo ( nIxMap, @fPosList, CE_FALSE );

    // P8 에서 P1 로 이송합니다.
    cemIxArcPTo ( nIxMap, -500, -500, -500, -1000, cemARC_CCW, CE_FALSE );

end;
```

NAME

cemIxArc3P / cemIxArc3PStart

- 3 점(Point) 원호 보간 상대 좌표 이송

(현재 좌표와 상대적 두 좌표)

INFORMATION
1 Interpolation Motion
! VC++ (6, 7, 8)/VB
BCB/Delphi
: Level 3
K 이송 함수
실제 이송이 진행되는 함수이므로, 사전에 반드시 안전을 확인합니다.

SYNOPSIS

r VT_I4 cemIxArc3P ([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)

r VT_I4 cemIxArc3PStart ([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

cemIxArc3P / cemIxArc3PStart 함수는 현재 좌표와 상대적 두 좌표를 매개변수로 하여 원호 보간 이송을 수행합니다. 이때 각 좌표는 상대 좌표로 표현됩니다.
cemIxArc3P 함수는 모션이 완료되기 전까지 반환되지 않으며, cemIxArc3PStart 함수는 모션을 시작시킨 후에 바로 반환됩니다.

원호 보간은 임의의 두 축에 대해서 적용됩니다. 아래 설명에서는 맵핑된 두 축을 X, Y 축으로 간주하여 설명합니다. 여기서 X 축이라 함은 맵핑된 두 축 중에서 축 번호가 낮은 축을 의미하며 Y 축은 축 번호가 높은 축을 의미합니다. 예를 들어 Z 축과 U 축이 맵핑된 두 축이라면 Z 축이 X 축에 해당하며 U 축이 Y 축에 해당합니다.

PARAMETER

▶ ***MapIndex*** : 맵 번호(Map index). 이 맵 번호를 사용하기 전에 먼저 cemIxMapAxes 함수를 통하여 해당 맵번호에 유효한 축들이 맵핑 되어 있어야 합니다.

▶ ***P2X*** : 현재 위치(시작 위치)로부터 두 번째 점까지 X 축상의 거리를 설정합니다.

▶ ***P2Y*** : 현재 위치(시작 위치)로부터 두 번째 점까지 Y 축상의 거리를 설정합니다.

▶ ***P3X*** : 두 번째 점부터 세 번째 점까지 X 축상의 거리를 설정합니다.

▶ ***P3Y*** : 두 번째 점부터 세 번째 점까지 Y 축상의 거리를 설정합니다.

▶ ***EndAngle*** : 원호 보간 이송을 완료할 목표 지점의 현재 위치에 대한 각도 값을 Degree(°)값으로 지정합니다. 각도의 부호가 (+)이면 반 시계방향, (-)이면 시계방향으로의 이동을 의미합니다. 단, EndAngle 값을 '0'으로 설정하시면 세번째 점까지 원호 보간 이송을 수행합니다.

▶ ***IsBlocking*** : cemIxArc3P 함수의 인자이며, 완료될 때까지 기다리는 동안 윈도우 메시지를 블록(Blocking)할 것인지를 결정합니다. 단, 쓰레드내에서 실행할 때는 이 값을 1(CE_TRUE)로 설정해 주어야 합니다.

Value	Meaning
0 (CE_FALSE)	윈도우 메시지를 블록(Blocking) 하지 않습니다. 따라서 해당 모션이 진행되는 동안 윈도우 이벤트를 처리합니다.
1 (CE_TRUE)	윈도우 메시지를 블록(Blocking) 합니다. 따라서 해당 모션이 완료되는 동안에는 윈도우 이벤트가 처리되지 않습니다.

RETURN VALUE

Value	Meaning
음수	수행 실패. 자세한 내용은 '에러처리' 편을 참고합니다
0 (ceERR_NONE)	수행 성공

SEE ALSO

cemIxArc3PTo, cemIxArc3PToStart

REFERENCE

□ cemIxArc3PStart 함수를 사용하는 경우에는 cemIxIsDone 함수나 cemIxWaitDone 함수를 사용하여 모션의 완료를 확인할 수 있습니다.

□ cemIxArc3P 함수를 사용하는 경우에는 내부적으로 루프를 수행하면서 모션이 완료되기를 기다리는데, 이때 블록 모드(Blocking Mode) 설정에 따라 윈도우 이벤트를 처리하는 방식이 달라집니다. 그러나 일반적으로 윈도우의 작업 쓰레드(Work Thread) 에서는 블록 모드를 사용하여, 함수 내부에서 지연 없이 쓰레드 내부의 작업에 집중할 수 있도록 설정하는 것이 바람직합니다.

□ cemIxArc3P 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.

스텝 드라이버는 INP 출력이 없는 경우가 일반적인데, 고객님의 부주의나 잘못된 설정으로 INP 입력에 대한 설정이 Enable 로 되어 있을 경우 INP 입력이 스텝 드라이버를 통해 발생하지 않는 이유 때문에 모션 완료가 되지 않는 경우가 발생할 수 있습니다.

고객 여러분들께서는 스텝 드라이버 사용시에 이점을 주의해 주시기를 부탁 드립니다.

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

EXAMPLE

C/C++

```
#include “ceSDK.h”
#include “ceSDKDef.h”

long nIxMap = 0;              // 보간 맵 설정
long nNodeID = 1;             // 해당 원격 노드 ID

void OnSetIxConfig ()
{
    // 0 번 축과 1 번 축을 보간 맵으로 설정합니다.
    cemIxMapAxes ( nIxMap, nNodeID, cemX1_MASK | cemY1_MASK, 0 );

    // 각 축의 속도를 설정합니다.
    cemCfgSpeedPattern_Set ( cemX1, cemSMODE_T, 1000, 5000, 5000 );
    cemCfgSpeedPattern_Set ( cemY1, cemSMODE_T, 1000, 5000, 5000 );
}

void OnIxArc3P_Move ()
{
    // 맵에 포함된 축의 속도 비율을 설정합니다.
    cemIxSpeedPattern_Set ( nIxMap, CE_FALSE, cemSMODE_T, 100, 70, 70 );

    double fX2 = 1000.0f, fY2 = 3000.0f;
    double fX3 = 3000.0f, fY3 = 2000.0f;

    /* 현재 위치, ( 현재 위치 + fX2, 현재 위치 + fY2 ), ( 현재 위치 + fX3, 현재 위치 + fY3 ) 의 세점을
    지나는 원을 그립니다.*/

    cemIxArc3P ( nIxMap, fX2, fY2, fX3, fY3, 360, CE_FALSE );

    // cemIxArc3PStart() 함수를 사용하는 경우에는 다음과 같이 코드를 작성합니다.
    // cemIxArc3PStart ( nIxMap, fX2, fY2, fX3, fY3, 360 );
    // cemIxWaitDone ( nIxMap, CE_FALSE );
}
```

Visual Basic

```
Dim nIxMap As Long          ‘ 보간 맵 설정
Dim nNodeID As Long         ‘ 해당 원격 노드 ID
```

```
nIxMap = 0
nNodeID = 1


Private Sub OnSetIxConfig ()

    ' 0 번 축과 1 번 축을 보간 맵으로 설정합니다.
    Call cemIxMapAxes ( nIxMap, nNodeID, cemX1_MASK Or cemY1_MASK, 0 )

    ' 각 축의 속도를 설정합니다.
    Call cemCfgSpeedPattern_Set ( cemX1, cemSMODE_T, 1000, 5000, 5000 )
    Call cemCfgSpeedPattern_Set ( cemY1, cemSMODE_T, 1000, 5000, 5000 )

End Sub


Private Sub OnIxArc3P_Move ()

    Dim fX2 As Double, fY2 As Double, fX3 As Double, fY3 As Double

    fX2 = 1000
    fY2 = 3000
    fX3 = 3000
    fY3 = 2000


    ' 맵에 포함된 축의 속도 비율을 설정합니다.
    Call cemIxSpeedPattern_Set ( nIxMap, CE_FALSE, cemSMODE_T, 100, 70, 70 )


    ' 현재 위치, ( 현재 위치 + fX2, 현재 위치 + fY2 ), ( 현재 위치 + fX3, 현재 위치 + fY3 ) 의 세점을
    ' 지나는 원을 그립니다.

    Call cemIxArc3P ( nIxMap, fX2, fY2, fX3, fY3, 360, CE_FALSE )

    ' cemIxArc3PStart() 함수를 사용하는 경우에는 다음과 같이 코드를 작성합니다.
    ' cemIxArc3PStart ( nIxMap, fX2, fY2, fX3, fY3, 360 )
    ' cemIxWaitDone ( nIxMap, CE_FALSE )

End Sub
```

```
Delphi

var
    nIxMap : LongInt;           // 보간 맵 설정
    nNodeID : LongInt;          // 해당 원격 노드 ID 설정
begin
    nIxMap := 0;
    nNodeId := 1;
end;


procedure OnSetIxConfig ();

    // 0 번 축과 1 번 축을 보간 맵으로 설정합니다.
    cemIxMapAxes ( nIxMap, nNodeID, cemX1_MASK or cemY1_MASK, 0 );
```

```
    // 각 축의 속도를 설정합니다.
    cemCfgSpeedPattern_Set ( cemX1, cemSMODE_T, 1000, 5000, 5000 );
    cemCfgSpeedPattern_Set ( cemY1, cemSMODE_T, 1000, 5000, 5000 );

end;


procedure OnIxArc3P_Move ();
var
    fX2, fY2, fX3, fY3 : Double;

begin
    fX2 := 1000;
    fY2 := 3000;
    fX3 := 3000;
    fY3 := 2000;


    // 맵에 포함된 축의 속도 비율을 설정합니다.
    cemIxSpeedPattern_Set ( nIxMap, CE_FALSE, cemSMODE_T, 100, 70, 70 );


    { 현재 위치, ( 현재 위치 + fX2, 현재 위치 + fY2 ), ( 현재 위치 + fX3, 현재 위치 + fY3 ) 의 세점을
     지나는 원을 그립니다. }

    cemIxArc3P ( nIxMap, fX2, fY2, fX3, fY3, 360, CE_FALSE );

    { cemIxArc3PStart() 함수를 사용하는 경우에는 다음과 같이 코드를 작성합니다.
     cemIxArc3PStart ( nIxMap, fX2, fY2, fX3, fY3, 360 )
     cemIxWaitDone ( nIxMap, CE_FALSE )
    }

end;
```

NAME

cemIxArc3PTo / cemIxArc3PToStart

- 3 점(Point) 원호 보간 절대 좌표 이송

(현재 좌표와 절대적 두 좌표)

INFORMATION	
1	Interpolation Motion
!	VC++ (6, 7, 8)/VB
	BCB/Delphi
:	Level 3
K	이송 함수

실제 이송이 진행되는 함수이므로, 사전에 반드시 안전을 확인합니다.

SYNOPSIS

r VT_I4 cemIxArc3PTo ([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)

r VT_I4 cemIxArc3PToStart ([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

cemIxArc3PTo / cemIxArc3PToStart 함수는 현재 좌표와 절대적 두 좌표를 매개변수로 하여 원호 보간 이송을 수행합니다. 이때 각 좌표는 절대 좌표로 표현됩니다.
cemIxArc3PTo 함수는 모션이 완료되기 전까지 반환되지 않으며, cemIxArc3PToStart 함수는 모션을 시작시킨 후에 바로 반환됩니 다.

원호 보간은 임의의 두 축에 대해서 적용됩니다. 아래 설명에서는 맵핑된 두 축을 X, Y 축으로 간주하여 설명합니다. 여기서 X 축이라 함은 맵핑된 두 축 중에서 축 번호가 낮은 축을 의미하며 Y 축은 축 번호가 높은 축을 의미합니다. 예를 들어 Z 축과 U 축이 맵핑된 두 축이라면 Z 축이 X 축에 해당하며 U 축이 Y 축에 해당합니다.

PARAMETER

▶ ***MapIndex*** : 맵 번호(Map index). 이 맵 번호를 사용하기 전에 먼저 cemIxMapAxes 함수를 통하여 해당 맵 번호에 유효한 축들이 맵핑 되어 있어야 합니다.

▶ ***P2X*** : 두 번째 점의 X 축 절대 좌표 값을 설정합니다.

▶ ***P2Y*** : 두 번째 점의 Y 축 절대 좌표 값을 설정합니다.

▶ ***P3X*** : 세 번째 점의 X 축 절대 좌표 값을 설정합니다.

▶ ***P3Y*** : 세 번째 점의 Y 축 절대 좌표 값을 설정합니다.

▶ ***EndAngle*** : 원호 보간 이송을 완료할 목표 지점의 현재 위치에 대한 각도 값을 Degree(°)값으로 지정합니다. 각도의 부호가 (+)이면 반 시계방향, (-)이면 시계방향으로의 이동을 의미합니다. 단, EndAngle 값을 '0'으로 설정하시면 세 번째 점까지 원호보간 이송을 수행합니다.

▶ ***IsBlocking*** : cemIxArc3PTo 함수의 인자이며, 완료될 때까지 기다리는 동안 윈도우 메시지를 블록(Blocking)할 것인지를 결정합니다. 단, 쓰레드내에서 실행할 때는 이 값을 1(CE_TRUE)로 설정해 주어야 합니다.

Value	Meaning
0 (CE_FALSE)	윈도우 메시지를 블록(Blocking) 하지 않습니다. 따라서 해당 모션이 진행되는 동안 윈도우 이벤트를 처리합니다.
1 (CE_TRUE)	윈도우 메시지를 블록(Blocking) 합니다. 따라서 해당 모션이 완료되는 동안에는 윈도우 이벤트가 처리되지 않습니다.

RETURN VALUE

Value	Meaning
음수	수행 실패. 자세한 내용은 '에러처리' 편을 참고합니다.
0 (ceERR_NONE)	수행 성공.

SEE ALSO

cemIxArc3P, cemIxArc3PStart

REFERENCE

□ cemIxArc3PToStart 함수를 사용하는 경우에는 cemIxIsDone 함수나 cemIxWaitDone 함수를 사용하여 모션의 완료를 확인할 수 있습니다.

□ cemIxArc3PTo 함수를 사용하는 경우에는 내부적으로 루프를 수행하면서 모션이 완료되기를 기다리는데, 이때 블록 모드(Blocking Mode) 설정에 따라 윈도우 이벤트를 처리하는 방식이 달라집니다. 그러나 일반적으로 윈도우의 작업 쓰레드(Work Thread) 에서는 블록 모드를 사용하여, 함수 내부에서 지연 없이 쓰레드 내부의 작업에 집중할 수 있도록 설정하는 것이 바람직합니다.

□ cemIxArc3PTo 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.

스텝 드라이버는 INP 출력이 없는 경우가 일반적인데, 고객님의 부주의나 잘못된 설정으로 INP 입력에 대한 설정이 Enable 로 되어 있을 경우 INP 입력이 스텝 드라이버를 통해 발생하지 않는 이유 때문에 모션 완료가 되지 않는 경우가 발생할 수 있습니다.

고객 여러분들께서는 스텝 드라이버 사용시에 이점을 주의해 주시기를 부탁 드립니다.

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

EXAMPLE

C/C++

```
#include “ceSDK.h”
#include “ceSDKDef.h”

long nIxMap = 0;              // 보간 맵 설정
long nNodeID = 1;             // 해당 원격 노드 ID

void OnSetIxConfig ()
{
    // 0 번 축과 1 번 축을 보간 맵으로 설정합니다.
    cemIxMapAxes ( nIxMap, nNodeID, cemX1_MASK | cemY1_MASK, 0 );

    // 각 축의 속도를 설정합니다.
    cemCfgSpeedPattern_Set ( cemX1, cemSMODE_T, 1000, 5000, 5000 );
    cemCfgSpeedPattern_Set ( cemY1, cemSMODE_T, 1000, 5000, 5000 );
}


void OnIxArc3PTo_Move ()
{
    // 맵에 포함된 축의 속도 비율을 설정합니다.
    cemIxSpeedPattern_Set ( nIxMap, CE_FALSE, cemSMODE_T, 100, 70, 70 );

    double fX2 = 0.0f, fY2 = 3000.0f;
    double fX3 = 3000.0f, fY3 = 0.0f;

    // 현재 위치, (fX2, fY2), (fX3, fY3) 세점을 지나는 원을 그립니다.
    cemIxArc3PTo ( MAP0, fX2, fY2, fX3, fY3, 360, CE_FALSE );

    // cemIxArc3PToStart() 함수를 사용하는 경우에는 다음과 같이 코드를 작성합니다.
    // cemIxArc3PToStart( MAP0, fX2, fY2, fX3, fY3, 360 );
    // cemIxWaitDone( MAP0, CE_FALSE );
}
```

Visual Basic

```
Dim nIxMap As Long          ‘ 보간 맵 설정
Dim nNodeID As Long         ‘ 해당 원격 노드 ID

nIxMap = 0
nNodeID = 1
```

```
Private Sub OnSetIxConfig ()

    ‘ 0 번 축과 1 번 축을 보간 맵으로 설정합니다.
    Call cemIxMapAxes ( nIxMap, nNodeID, cemX1_MASK Or cemY1_MASK, 0 )

    ‘ 각 축의 속도를 설정합니다.
    Call cemCfgSpeedPattern_Set ( cemX1, cemSMODE_T, 1000, 5000, 5000 )
    Call cemCfgSpeedPattern_Set ( cemY1, cemSMODE_T, 1000, 5000, 5000 )

End Sub


Private Sub OnIxArc3PTo_Move ()

    Dim fX2 As Double, fY2 As Double, fX3 As Double, fY3 As Double

    fX2 = 0
    fY2 = 3000
    fX3 = 3000
    fY3 = 0

    ‘ 맵에 포함된 축의 속도 비율을 설정합니다.
    Call cemIxSpeedPattern_Set ( nIxMap, CE_FALSE, cemSMODE_T, 100, 70, 70 )

    ‘ 현재 위치, (fX2, fY2), (fX3, fY3) 세점을 지나는 원을 그립니다.
    Call cemIxArc3PTo ( MAP0, fX2, fY2, fX3, fY3, 360, CE_FALSE )

    ‘ cemIxArc3PToStart() 함수를 사용하는 경우에는 다음과 같이 코드를 작성합니다.
    ‘ cemIxArc3PToStart( MAP0, fX2, fY2, fX3, fY3, 360 )
    ‘ cemIxWaitDone( MAP0, CE_FALSE )

End Sub
```

```
Delphi

var
    nIxMap : LongInt;           // 보간 맵 설정
    nNodeID : LongInt;          // 해당 원격 노드 ID 설정
begin
    nIxMap := 0;
    nNodeId := 1;
end;


procedure OnSetIxConfig ();
begin
    // 0 번 축과 1 번 축을 보간 맵으로 설정합니다.
    cemIxMapAxes ( nIxMap, nNodeID, cemX1_MASK or cemY1_MASK, 0 );

    // 각 축의 속도를 설정합니다.
    cemCfgSpeedPattern_Set ( cemX1, cemSMODE_T, 1000, 5000, 5000 );
    cemCfgSpeedPattern_Set ( cemY1, cemSMODE_T, 1000, 5000, 5000 );

end;
```

```
procedure OnIxArc3PTo_Move ();
var
    fX2, fY2, fX3, fY3 : Double;

    fX2 := 0;
    fY2 := 3000;
    fX3 := 3000;
    fY3 := 0;

    // 맵에 포함된 축의 속도 비율을 설정합니다.
    cemIxSpeedPattern_Set ( nIxMap, CE_FALSE, cemSMODE_T, 100, 70, 70 );

    // 현재 위치, (fX2, fY2), (fX3, fY3) 세점을 지나는 원을 그립니다.
    cemIxArc3PTo ( MAP0, fX2, fY2, fX3, fY3, 360, CE_FALSE );

    { cemIxArc3PToStart() 함수를 사용하는 경우에는 다음과 같이 코드를 작성합니다.
     cemIxArc3PToStart( MAP0, fX2, fY2, fX3, fY3, 360 )
     cemIxWaitDone( MAP0, CE_FALSE )
    }

end;
```

NAME

cemIxStop / cemIxStopEmg

- 보간 이송 정지

- 보간 비상 정지

INFORMATION

1 Interpolation Motion

! VC++ (6, 7, 8)/VB

BCB/Delphi

: Level 3

K 정지 함수

고속 이송시에 비상 정지를 주의하십시오. 기구물의 손상이나 안전사고에 원인이 될 수 있습니다.

SYNOPSIS

r VT_I4 cemIxStop ([in] VT_I4 MapIndex)

r VT_I4 cemIxStopEmg ([in] VT_I4 MapIndex)

DESCRIPTION

cemIxStop / cemIxStopEmg 함수는 지정한 보간 맵에 대한 보간 작업을 정지합니다.

정지 시에 cemIxStop 함수를 사용하면 감속 후 정지하며, cemIxStopEmg 를 사용하면 감속 없이 즉시 정지를 수행합니다.

PARAMETER

▶ ***MapIndex*** : 맵 번호(Map index). 이 맵 번호를 사용하기 전에 먼저 cemIxMapAxes 함수를 통하여 해당 맵 번호에 유효한 축들이 맵핑 되어 있어야 합니다.

RETURN VALUE

Value	Meaning
음수	수행 실패. 자세한 내용은 '에러처리' 편을 참고합니다.
0 (ceERR_NONE)	수행 성공.

EXAMPLE

C/C++

```
#include “ceSDK.h”
#include “ceSDKDef.h”


void OnIxMove ()
{
    long nIxMap = 0;            // 보간 맵 설정
    long nNodeID = 1;           // 해당 원격 노드 ID 설정

    cemIxMapAxes ( nIxMap, nNodeID, cemX1_MASK | cemY1_MASK, 0 );

    double fDistList[2] = { 10000.0f, 20000.0f };

    // 직선 보간 이송을 시작합니다.
    cemIxLineStart ( nIxMap , fDistList );
    Sleep (1000);

    // 보간 이송 축들을 감속 후 정지합니다.
    cemIxStop ( nIxMap );

    // cemIxStopEmg() 함수를 사용할 경우 감속 없이 즉시 정지합니다.
    // cemIxStopEmg ( nIxMap );
}
```

Visual Basic

```
Private Sub OnIxMove ()

    Dim nIxMap As Long        ‘ 보간 맵 설정
    Dim nNodeID As Long       ‘ 해당 원격 노드 ID 설정
    Dim fDistList(2) As Double

    nIxMap = 0
    nNodeID = 1

    fDistList(0) = 10000
    fDistList(1) = 20000

    Call cemIxMapAxes ( nIxMap, nNodeID, cemX1_MASK Or cemY1_MASK, 0 )

    ‘ 직선 보간 이송을 시작합니다.
    Call cemIxLineStart ( nIxMap , fDistList(0) )

    Sleep ( 1000 )

    ‘ 보간 이송 축들을 감속 후 정지합니다.
    Call cemIxStop ( nIxMap )

    ‘ cemIxStopEmg() 함수를 사용할 경우 감속 없이 즉시 정지합니다.
    ‘ cemIxStopEmg ( nIxMap )
```

```
End Sub
```

Delphi

```
procedure OnIxMove ();
var
    nIxMap : LongInt;                    // 보간 맵 설정
    nNodeID : LongInt;                   // 해당 원격 노드 ID 설정
    fDistList : Array[0..1] of Double;

begin
    nIxMap := 0;
    nNodeID := 1;

    fDistList[0] := 10000;
    fDistList[1] := 20000;

    cemIxMapAxes ( nIxMap, nNodeID, cemX1_MASK or cemY1_MASK, 0 );

    // 직선 보간 이송을 시작합니다.
    cemIxLineStart ( nIxMap , @fDistList );

    Sleep ( 1000 );

    // 보간 이송 축들을 감속 후 정지합니다.
    cemIxStop ( nIxMap );

    { cemIxStopEmg() 함수를 사용할 경우 감속 없이 즉시 정지합니다.
     cemIxStopEmg ( nIxMap );
    }

end;
```

NAME

cemIxIsDone

- 보간 모션 완료 확인

INFORMATION	
1	Interpolation Motion
!	VC++ (6, 7, 8)/VB
	BCB/Delphi
:	Level 3
J	위험 요소 없음

SYNOPSIS

r VT_I4 cemIxIsDone ([in] VT_I4 MapIndex, [out] VT_PI4 IsDone)

DESCRIPTION

지정한 보간 맵에 해당하는 보간 작업이 완료됐는지를 확인합니다.

PARAMETER

▶ ***MapIndex*** : 맵 번호(Map index). 이 맵 번호를 사용하기 전에 먼저 cemIxMapAxes 함수를 통하여 해당 맵 번호에 유효한 축들이 맵핑 되어 있어야 합니다.

▶ ***IsDone*** : 이 매개를 통해 모션 작업이 완료되었는지를 판단할 수 있습니다.

Value	Meaning
0 (CE_FALSE)	모션작업이 완료되지 않음.
1 (CE_TRUE)	모션작업이 완료됨.

RETURN VALUE

Value	Meaning
음수	수행 실패. 자세한 내용은 '에러처리' 편을 참고합니다.
0 (ceERR_NONE)	수행 성공.

SEE ALSO

cemIxWaitDone

EXAMPLE

C/C++

```
#include “ceSDK.h”
#include “ceSDKDef.h”

void OnIxMove ()
{
    long nIxMap = 0;            // 보간 맵 설정
    long nNodeID = 1;           // 해당 원격 노드 ID

    cemIxMapAxes ( nIxMap, nNodeID, cemX1_MASK | cemY1_MASK, 0 );

    double fDistList[2] = { 10000.0f, 20000.0f };

    // 직선 보간 이송을 시작합니다.
    cemIxLineStart ( nIxMap , fDistList );

    long nIsDone = CE_FALSE;

    while( !nIsDone )
    {
            // 직선 보간 이송 완료 여부를 확인합니다.
            cemIxIsDone( MAP0, &nIsDone );
}
```

Visual Basic

```
Private Sub OnIxMove ()

    Dim nIxMap As Long      ‘ 보간 맵 설정
    Dim nNodeID As Long     ‘ 해당 원격 노드 ID
    Dim nIsDone As Long     ‘ 직선 보간 이송 완료 정보.
    Dim fDistList(2) As Double

    nIxMap = 0
    nNodeID = 1

    fDistList(0) = 10000
    fDistList(1) = 20000

    Call cemIxMapAxes ( nIxMap, nNodeID, cemX1_MASK Or cemY1_MASK, 0 )

    ‘ 직선 보간 이송을 시작합니다.
    Call cemIxLineStart ( nIxMap , fDistList(0) )

    nIsDone = CE_FALSE
    While ( nIsDone = CE_FALSE )
            cemIxIsDone ( nIxMap, nIsDone )   ‘ 직선 보간 이송 완료 여부를 확인합니다.
    Wend

End Sub
```

Delphi

```
procedure OnIxMove ();
var
    nIxMap : LongInt;                   // 보간 맵 설정
    nNodeID : LongInt                   // 해당 원격 노드 ID
    nIsDone : LongInt                   // 직선 보간 이송 완료 정보.
    fDistList : Array[0..1] of Double;

begin
    nIxMap := 0;
    nNodeID := 1;

    fDistList[0] := 10000;
    fDistList[1] := 20000;

    cemIxMapAxes ( nIxMap, nNodeID, cemX1_MASK or cemY1_MASK, 0 );

    // 직선 보간 이송을 시작합니다.
    cemIxLineStart ( nIxMap , @fDistList );

    nIsDone := CE_FALSE;

    while nIsDone = CE_FALSE do
    begin
            cemIxIsDone ( nIxMap, @nIsDone );       // 직선 보간 이송 완료 여부를 확인합니다.
    end;

end;
```

NAME

cemIxWaitDone

- 보간 모션 완료 대기

INFORMATION	
1	Interpolation Motion
!	VC++ (6, 7, 8)/VB
	BCB/Delphi
:	Level 3
J	위험 요소 없음

SYNOPSIS

r VT_I4 cemIxWaitDone ([in] VT_I4 MapIndex, [in] VT_I4 IsBlocking)

DESCRIPTION

지정한 보간 맵에 해당하는 보간 작업 이송이 완료될 때까지 기다립니다.

PARAMETER

▶*MapIndex* : 맵 번호(Map index). 이 맵 번호를 사용하기 전에 먼저 cemIxMapAxes 함수를 통하여 해당 맵 번호에 유효한 축들이 맵핑 되어 있어야 합니다.

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

Value	Meaning
0 (CE_FALSE)	윈도우 메시지를 블록(Blocking) 하지 않습니다. 따라서 해당 모션이 진행되는 동안 윈도우 이벤트를 처리합니다.
1 (CE_TRUE)	윈도우 메시지를 블록(Blocking) 합니다. 따라서 해당 모션이 완료되는 동안에는 윈도우 이벤트가 처리되지 않습니다.

RETURN VALUE

Value	Meaning
음수	수행 실패. 자세한 내용은 '에러처리' 편을 참고합니다
0 (ceERR_NONE)	수행 성공

SEE ALSO

cemIxIsDone

REFERENCE

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

□ 스텝 드라이버를 사용 중인 고객님들께서는 다음을 참조해 주십시오.
스텝 드라이버는 INP 출력이 없는 경우가 일반적인데, 고객님의 부주의나 잘못된 설정으로 INP 입력에 대한 설정이 Enable 로 되어 있을 경우 INP 입력이 스텝 드라이버를 통해 발생하지 않는 이유 때문에 모션 완료가 되지 않는 경우가 발생할 수 있습니다. 고객 여러분들께서는 스텝 드라이버 사용시에 이점을 주의해 주시기를 부탁 드립니다.

□ 서보 드라이버의 LSP, LSN 신호를 사용 중인 고객님께서는 다음을 참조해 주십시오.
서보드라이버의 입력 신호 중 하나인 EL(End of Limit) 신호는 저희 ㈜ 커미조아 모션 컨트롤러뿐만 아니라 서보드라이버에도 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 신호를 서보 드라이버 측에 전달하기 위한 용도로 사용됩니다.

그러나, 모션 소프트웨어에서 INP 설정이 되어 있는 경우 EL 신호가 검출 된 후에 일부 서보 드라이버에서는 진행 방향에서 정지 한 후 더 이상 움직이지 않는 상황이 발생하며, 이 상황에서 INP 신호가 출력되지 않아, 모션 이송이 완료되지 못하고, 명시적으로 STOP 명령을 통해 강제적으로 모션을 종료해야 되는 현상이 발생할 수 있습니다. 이 현상은 EL 모드를 통해 원점 복귀를 하는 상황에서도 발생할 수 있습니다.

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

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

EXAMPLE

C/C++

```
#include “ceSDK.h”
#include “ceSDKDef.h”

void OnIxMove ()
{
    long nIxMap = 0;            // 보간 맵 설정
    long nNodeID = 1;           // 해당 원격 노드 ID

    cemIxMapAxes ( nIxMap, nNodeID, cemX1_MASK | cemY1_MASK, 0 );

    double fDistList[2] = { 10000.0f, 20000.0f };

    // 직선 보간 이송을 시작합니다.
    if ( cemIxLineStart ( MAP0 , fDistList ) == ceERR_NONE )
    {
            // 모션이 완료 시까지 기다립니다.
            cemIxWaitDone ( nIxMap, CE_FALSE );
    }
}
```

Visual Basic

```
Private Sub OnIxMove ()

    Dim nIxMap As Long        ‘ 보간 맵 설정
    Dim nNodeID As Long       ‘ 해당 원격 노드 ID
    Dim nIsDone As Long       ‘ 직선 보간 이송 완료 정보.
    Dim fDistList(2) As Double

    nIxMap = 0
    nNodeID = 1

    fDistList(0) = 10000
    fDistList(1) = 20000

    Call cemIxMapAxes ( nIxMap, nNodeID, cemX1_MASK Or cemY1_MASK, 0 )

    ‘ 직선 보간 이송을 시작합니다.
    If cemIxLineStart ( MAP0 , fDistList(0) ) = ceERR_NONE Then
            Call cemIxWaitDone ( nIxMap, CE_FALSE ) ‘ 모션이 완료 시까지 기다립니다.
    End If

End Sub
```

Delphi

```
procedure OnIxMove ();
```

```
var
    nIxMap : LongInt;                   // 보간 맵 설정
    nNodeID : LongInt;                  // 해당 원격 노드 ID
    nIsDone : LongInt;                  // 직선 보간 이송 완료 정보.
    fDistList : Arrary[0..1] of Double;

begin
    nIxMap := 0;
    nNodeID := 1;

    fDistList[0] := 10000;
    fDistList[1] := 20000;

    cemIxMapAxes ( nIxMap, nNodeID, cemX1_MASK or cemY1_MASK, 0 );

    // 직선 보간 이송을 시작합니다.
    if cemIxLineStart ( MAP0 , @fDistList ) = ceERR_NONE then
    begin
            cemIxWaitDone ( nIxMap, CE_FALSE );     // 모션이 완료 시까지 기다립니다.
    end;

end;
```

7.3 원점복귀(Home Return)

이 단원에서는 원점 복귀(Home Return)에 관련된 함수들을 소개합니다. 원점 복귀는 모션제어의 대상이 되는 구조물이 원점 위치로 자동 복귀하도록 하는 작업입니다. 원점 복귀 작업이 완료되면 Command Counter, Feedback Counter, Deviation Counter 는 자동으로 0(Zero)로 초기화됩니다.

원점 복귀 작업을 수행하기 위해서는 ORG(HOME), EZ 및 EL 신호가 참조되는데 이 신호들의 의미 및 작용은 다음과 같습니다.

□ ORG (HOME) 신호

ORG 신호는 구조물이 원점에 복귀했는지를 센서로부터 입력 받는 신호입니다. 일반적으로는 근접 센서와 같은 센서들을 이용하여 원점 복귀 여부를 감지하게 됩니다. 이 신호는 'ORG' 단자를 통하여 입력되어야 합니다.

□ EZ 신호

엔코더의 Z 상(제로 펄스) 신호를 의미합니다. 이 신호는 원점 복귀 모드에 따라 ORG 신호 또는 EL 신호와 함께 사용되어 보다 정밀한 원점복귀 작업을 수행할 수 있도록 해줍니다. 이 신호는 'EZ+' 단자와 'EZ-' 단자를 통하여 입력되어야 합니다.

□ EL 신호

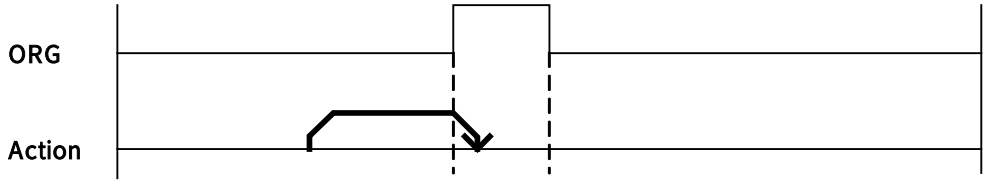
기계적인 리밋(Limit) 신호를 의미합니다. 이 신호는 일반적으로 구조물이 움직일 수 있는 한계점을 감지하기 위해 사용되나 원점 복귀 모드에 따라 ORG 신호의 대용으로도 사용될 수 있습니다. EL 신호는 (+)방향 리밋 신호와 (-)방향 리밋 두 가지 신호가 있습니다. (+)방향 리밋 신호는 '+EL' 단자, 그리고 (-)방향 리밋 신호는 '-EL' 단자를 통하여 입력되어야 합니다.

7.3.1 원점 복귀 모드 안내

㈜커미조아 모션컨트롤러는 다음과 같이 13 가지의 다양한 원점 복귀 모드를 제공합니다. 각 원점 복귀 모드에 따른 동작 방식은 다음 설명과 같습니다. 아래의 그림은 모두 속도모드를 Trapezoidal 모드로 설정한 상태임을 가정하여 그려 진 것이며, 만일 Constant 속도 모드로 설정된 경우에는 감속이 없이 즉시 정지하게 됩니다.

□ MODE 0 : ORG ON => Stop

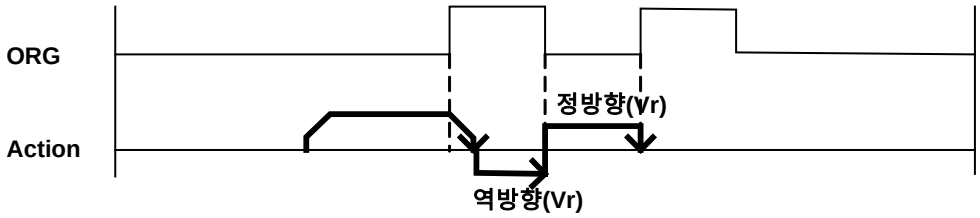
MODE 0 에서는 ORG 신호가 OFF 에서 ON 으로 바뀌는 순간에 모션을 감속 후 정지하고 원점 복귀 작업을 종료합니다.



□ MODE 1 : ORG ON => Stop => Back (Vr) => ORG OFF => Forward(Vr) => ORG ON => Stop

MODE 1 에서는 ORG 신호가 OFF 에서 ON 으로 바뀌는 순간에 모션을 감속 후 정지한 후 ORG 신호가 OFF 가 될 때까지 Vr(Reverse Speed)의 속도로 역방향 회전을 수행합니다. ORG 신호가 OFF 되면 다시 Vr 의 속도로 정방향 회전을 수행하다가 ORG 신호가 다시 ON 되는 순간에 원점 복귀작업을 종료합니다.

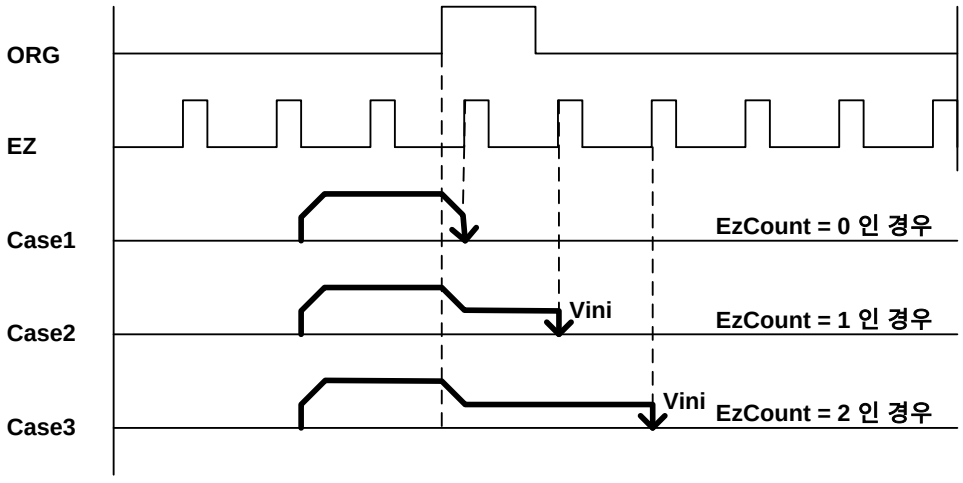
이 모드를 사용할 때 주의할 것은 처음 ORG 신호가 ON 으로 변경되어 감속 후 정지하는 도중에 ORG 센서의 ON 으로 유지되는 시간이 짧아서 이미 OFF 상태로 변경되면 역방향 이송을 생략하고 최종적으로 원점 센서를 찾으려는 단계로 넘어갑니다. 따라서 이러한 경우에는 (-)Limit 위치까지 이송하게 됩니다. 이러한 경우에도 자동으로 다시 탈출하여 정상적인 원점복귀 작업을 다시 수행하지만 의도하지 않게 원점복귀 시간이 길어질 수 있습니다. 이를 방지하는 방법은 구조적으로 ORG 신호가 ON 으로 유지되는 시간을 길게 해주거나 또는 원점 복귀 시의 작업 속도를 낮추거나 감속도를 크게 하여 감속 시 이동되는 거리를 짧게 해주면 됩니다.



Vr : Reverse Speed를 의미합니다.

□ MODE 2 : ORG ON => Slow Down (Vini) => Stop on EZ Count

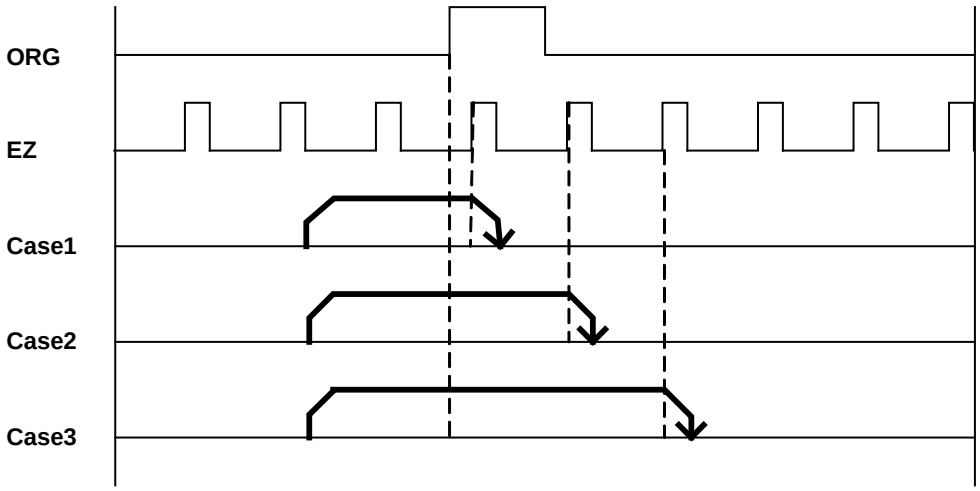
MODE 2 에서는 ORG 신호가 OFF 에서 ON 으로 바뀌는 순간에 모션을 감속한 후 초기속도 값으로 모션을 진행하다가 EZ 신호에 따라 원점 복귀 작업을 종료합니다. cemHomeConfig_Set 함수를 통하여 미리 설정된 EzCount 값에 따라 종료하는 시점은 다음과 같이 달라집니다.



Vini : 초기속도를 의미하며, cnmSxOptIniSpeed_Set 함수 참조

□ MODE 3 : ORG => Stop on EZ Count

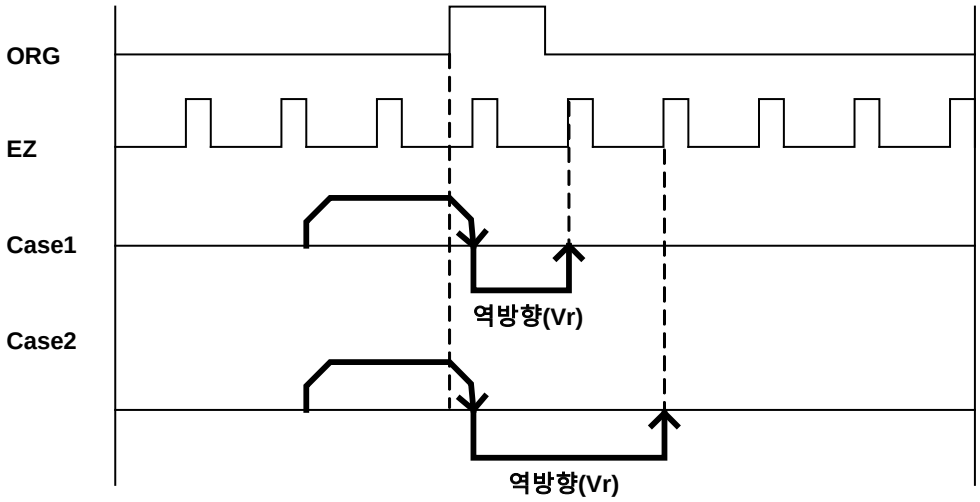
MODE 3 에서는 ORG 신호가 OFF 에서 ON 으로 바뀌고 난 후 발생하는 EZ 신호에 따라 감속 후 정지합니다. cemHomeConfig_Set 함수를 통하여 미리 설정된 EzCount 값에 따라 감속을 시작하는 시점은 다음과 같이 달라집니다.



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

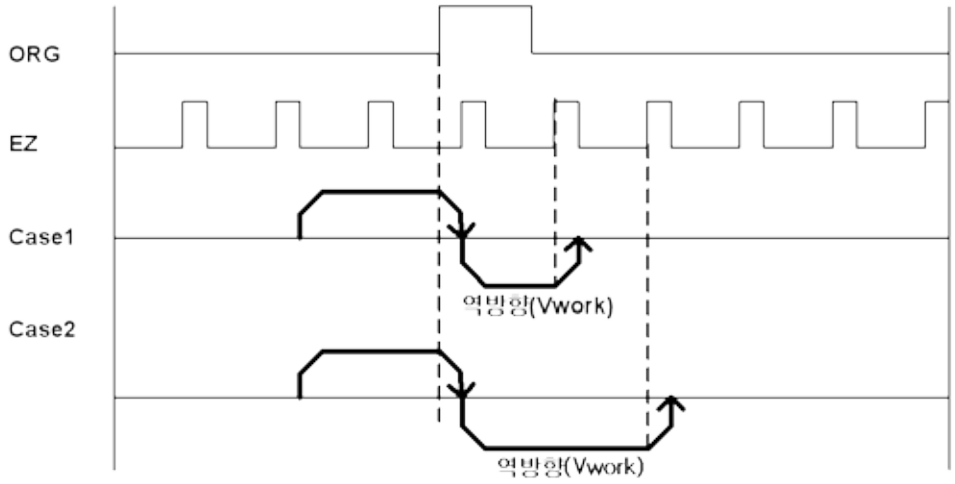
MODE 4 에서는 ORG 신호가 OFF 에서 ON 으로 바뀌는 순간 감속 후 정지합니다. 그리고 Vr 의 속도로 역방향 회전한 후 EZ 신호에 따라 원점 복귀 작업을 종료합니다.



Case1 : EzCount = 1 인 경우
Case2 : EzCount = 2 인 경우
Vr : Reverse Speed

□ MODE 5 : ORG ON => Stop => Back (Vwork) => Stop on EZ Count

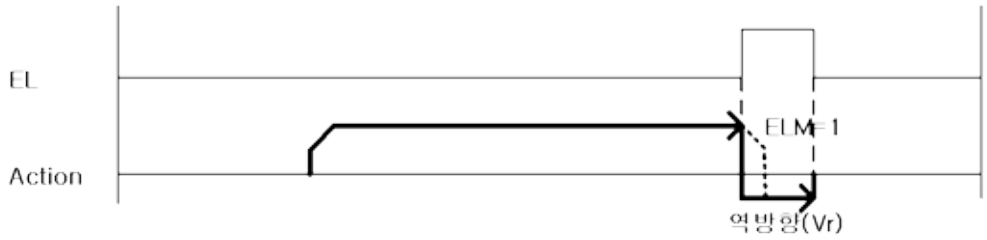
MODE 5 에서는 ORG 신호가 OFF 에서 ON 으로 바뀌는 순간 감속 후 정지합니다. 그리고 작업 속도까지 가속하여 역방향 회전한 후 EZ 신호에 따라 감속 후 원점 복귀 작업을 종료합니다.



Case1 : EzCount = 1 인 경우, cnmHomeConfig_Set 함수 참조
Case2 : EzCount = 2 인 경우, cnmHomeConfig_Set 함수 참조

□ MODE 6 : EL ON => Stop => Back (Vr) => EL OFF => Stop

MODE 6 에서는 EL 신호가 ON 으로 바뀌는 순간 즉시 정지 또는 ELM=1 인 경우에 감속 후 정지합니다. 그리고 반대 방향으로 Vr 속도로 회전하다가 EL 신호가 OFF 되는 순간에 복귀작업을 종료합니다. 여기서 ELM=1 은 EL 의 “Stop mode”가 “감속 후 정지” 모드로 설정됨을 의미합니다.

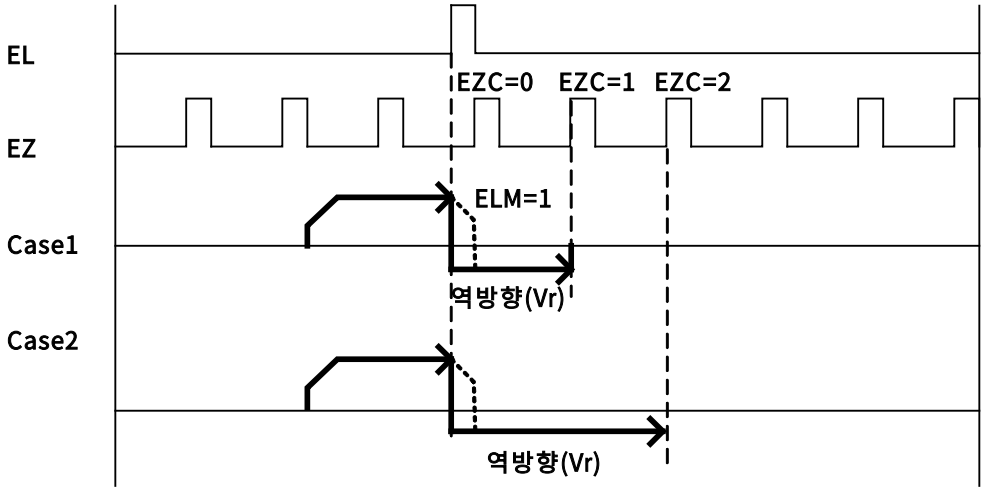


Vr : Reverse Speed

cemCfgMioProperty_Set 함수를 통해, EL 신호가 ON 으로 변경되는 순간 모션을 즉시 정지를 할 것인지 감속 후 정지를 할 것인지를 설정할 수 있습니다.
자세한 설명은 cemCfgMioProperty_Set 함수를 통해 확인해 주시기 바랍니다.

□ MODE 7 : EL ON => Stop => Back (Vr) => Stop on EZ count

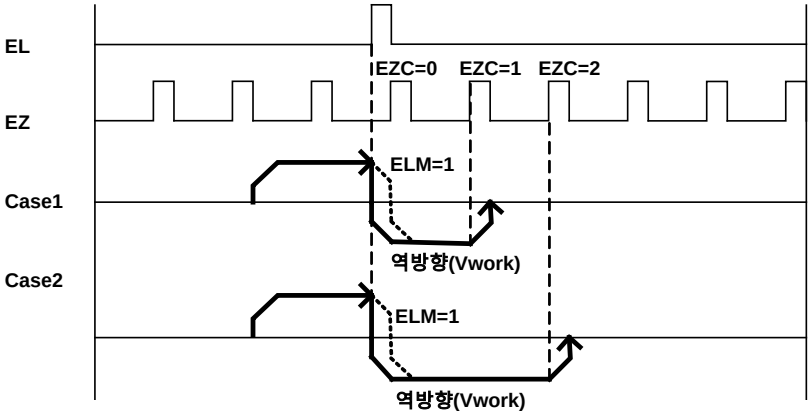
MODE 7 에서는 EL 신호가 ON 으로 바뀌는 순간 즉시 정지 또는 ELM=1 인 경우에 감속 후 정지합니다. 그리고 반대 방향으로 Vr 속도로 회전하다가 EzCount 에 따라 원점 복귀작업을 종료합니다. 여기서 ELM=1 은 EL 의 “Stop mode”가 “감속 후 정지” 모드로 설정됨을 의미합니다.



Case1 : EzCount = 1 인 경우
Case2 : EzCount = 2 인 경우
Vr : Reverse Speed

□ MODE 8 : EL ON => Stop => Back (Vwork) => Stop on EZ count

MODE 8 에서는 EL 신호가 ON 으로 바뀌는 순간 즉시 정지 또는 ELM=1 인 경우에 감속 후 정지합니다. 그리고 작업 속도까지 가속하여 반대 방향으로 회전한 후 EZ 신호에 따라 감속 후 원점 복귀 작업을 종료합니다. 여기서 ELM=1 은 EL 의 “Stop mode”가 “감속 후 정지” 모드로 설정됨을 의미합니다.



Case1 : EzCount = 1 인 경우
Case2 : EzCount = 2 인 경우
Vwork : 작업속도를 의미하며 cnmHomeSpeedPattern_Set 함수 참조

□ MODE 9 : MODE0 => Operate till dev. counter 0

MODE 9 에서는 MODE 0 과 동일한 원점 복귀 작업을 수행한 후, 편차 카운터(Deviation Counter)가 '0'이 되도록 모션을 다시 취한 후에 원점복귀 작업을 종료합니다.

□ MODE 10 : MODE3 => Operate till dev. counter 0

MODE 10 에서는 MODE 3 과 동일한 원점 복귀 작업을 수행한 후, 편차 카운터(Deviation Counter)가 '0'이 되도록 모션을 다시 취한 후에 원점 복귀 작업을 종료합니다.

□ MODE 11 : MODE5 => Operate till dev. counter 0

MODE 11 에서는 MODE 5 과 동일한 원점 복귀 작업을 수행한 후, 편차 카운터(Deviation Counter)가 '0'이 되도록 모션을 다시 취한 후에 원점 복귀 작업을 종료합니다.

□ MODE 12 : MODE8 > Operate till dev. counter 0

MODE 12 에서는 MODE 8 과 동일한 원점 복귀 작업을 수행한 후, 편차 카운터(Deviation Counter)가 '0'이 되도록 모션을 다시 취한 후에 원점 복귀 작업을 종료합니다.

7.3.2 자동 원점 검색 기능에 대하여

일반적으로 기구물의 위치는 원점 센서를 기준으로 (+) 방향의 위치에 있으며, 따라서 (-) 방향으로 원점복귀를 수행하면 됩니다. 하지만 원점복귀를 시작하는 시점에 기구물이 이미 원점 센서가 ON 이 되어 있는 위치에 있거나 원점 센서와 (-)EL 센서 사이에 위치한 경우에는 원점 센서를 기준으로 (+) 방향의 위치까지 탈출한 후에 정상적인 원점복귀 작업을 수행하여야 합니다. 이러한 기능을 "자동 원점 검색" 기능이라 하며, ㈜커미조아의 모션컨트롤러는 이 기능을 자동으로 수행 해주므로 기구물의 위치에 관계없이 원점복귀 작업을 수행할 수 있습니다.

기구물과 각 센서들의 위치에 따른 원점복귀 절차는 다음과 같이 약간씩 다릅니다. 단, 원점복귀 작업의 방향을 음의 방향으로 한 경우에 대한 것입니다. 만일 원점복귀를 양의 방향으로 한 경우에는 –EL 신호 대신 +EL 신호가 사용됩니다.

□ CASE 1. 원점 센서를 기준으로 양의 방향 위치에서 원점복귀를 시작한 경우

정상적인 원점복귀 작업을 수행합니다.

□ CASE 2. 원점 센서가 ON 인 상태에서 원점복귀를 시작한 경우

이러한 경우에는 먼저 원점 탈출 거리(Escape distance) 만큼 양의 방향으로 이동한 후 다시 정상적인 원점복귀 작업을 수행합니다. 원점 탈출 거리만큼 탈출하였어도 원점 센서가 ON 상태이면 원점 탈출 거리만큼 탈출하는 작업을 반복하므로 탈출 거리가 짧아도 원점을 탈출하는 데는 아무 문제가 없습니다.

□ CASE 3. 원점 센서를 기준으로 음의 방향 위치에서 원점복귀를 시작한 경우

이러한 경우에는 먼저 음의 방향으로 이동을 시작합니다. 그리고 –EL(Negative Limit) 센서가 ON 되면 정지 후 양의 방향으로 이동합니다. 원점 센서가 ON 되면 다시 정지 후 CASE 2 에서와 같이 원점 탈출 거리만큼 양의 방향으로 이동한 후 다시 정상적인 원점복귀 작업을 수행합니다.

[그림 7-3]는 원점복귀 모드를 0, 속도 패턴을 사다리꼴 방식으로 하고 원점복귀 작업을 수행할 때 위의 각 경우에 대하여 동작하는 방식을 도식적으로 나타낸 것입니다.

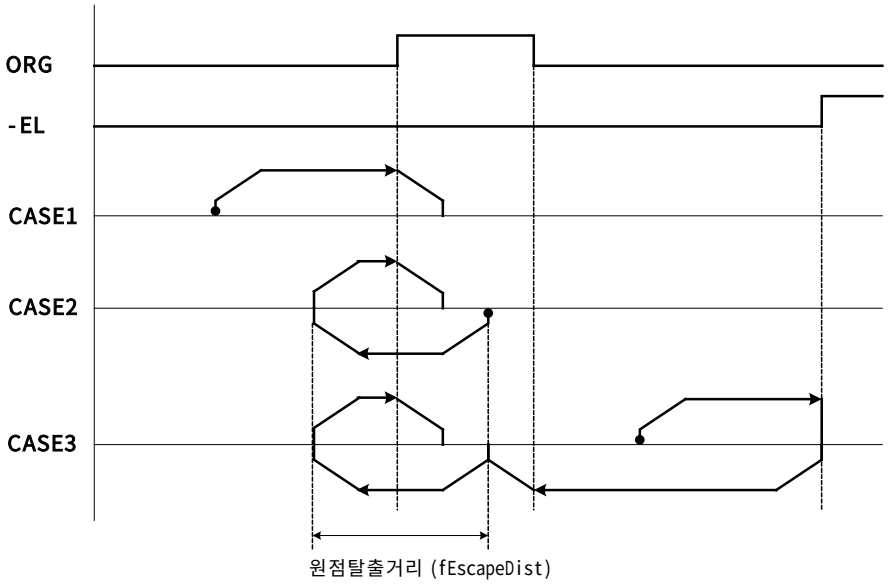


그림 7-3 기구물과 센서의 위치에 따른 원점복귀 작업

7.3.3 함수 요약

"원점 복귀"와 관련된 함수들은 다음과 같습니다.

Summary of Functions
r VT_I4 cemHomeConfig_Set ([in] VT_I4 Axis, [in] VT_I4 HomeMode, [in] VT_I4 Dir, [in] VT_I4 EzCount, [in] VT_R8 EscDist, [in] VT_R8 Offset) 원점 복귀 환경 설정을 구성합니다. 원점 복귀 모드와 Z 상 검출 횟수, 자동 원점 탈출 거리, 원점 복귀 완료 후 추가 이송 거리등을 설정할 수 있습니다.
r VT_I4 cemHomeConfig_Get ([in] VT_I4 Axis, [out] VT_PI4 HomeMode, [out] VT_PI4 Dir, [out] VT_PI4 EzCount, [out] VT_PR8 EscDist, [out] VT_PR8 Offset) 원점 복귀 환경 설정을 반환합니다. 원점 복귀 모드와 Z 상 검출 횟수, 자동 원점 탈출 거리, 원점 복귀 완료 후 추가 이송 거리등이 반환됩니다.
r VT_I4 cemHomePosClrMode_Set ([in] VT_I4 Axis, [in] VT_I4 PosClrMode) 원점 복귀 완료 후 발생하는 모션 컨트롤러와 서보 드라이버간의 제어 편차에 의해 발생한 입력 펄스 (Feedback Pulse) 처리에 대한 환경을 구성합니다.
r VT_I4 cemHomePosClrMode_Get ([in] VT_I4 Axis, [out] VT_PI4 PosClrMode) 원점 복귀 완료 후 발생하는 모션 컨트롤러와 서보 드라이버간의 제어 편차에 의해 발생한 입력 펄스 처리에 대한 설정 상태를 반환합니다.
r VT_I4 cemHomeSpeedPattern_Set ([in] VT_I4 Axis, [in] VT_I4 SpeedMode, [in] VT_R8 Vel, [in] VT_R8 Accel, [in] VT_R8 Decel, [in] VT_R8 RevVel) 원점 복귀 속도 환경을 설정합니다. 이 속도는 원점 복귀 시에만 사용되는 속도이며, 일반 모션 속도와 개별 적인 원점 복귀 속도를 지정할 수 있습니다.
r VT_I4 cemHomeSpeedPattern_Get ([in] VT_I4 Axis, [out] VT_PI4 SpeedMode, [out] VT_PR8 Vel, [out] VT_PR8 Accel, [out] VT_PR8 Decel, [out] VT_PR8 RevVel) 원점 복귀 속도 환경을 반환합니다. 이 속도는 원점 복귀 시에만 사용되는 속도이며, 일반 모션 속도와 개별 적인 원점 복귀 속도를 반환합니다.
r VT_I4 cemHomeMove ([in] VT_I4 Axis, [in] VT_I4 IsBlocking) 원점 복귀 환경 설정을 바탕으로 원점 복귀 이송 작업을 수행합니다. 이 함수는 원점 복귀 이송이 완료 되기 전까지 반환되지 않습니다.
r VT_I4 cemHomeMoveStart ([in] VT_I4 Axis) 원점 복귀 환경 설정을 바탕으로 원점 복귀 이송 작업을 수행합니다. 이 함수는 원점 복귀 이송 시작 후 바로 반환됩니다.
VT_I4 cemHomeSuccess_Get ([in] VT_I4 Axis, [out] VT_PI4 IsSuccess) 이전에 실행된 원점 복귀 완료 상태를 확인합니다. 원점 복귀 완료 상태는 이전에 실행된 원점복귀 상태 이며, 모션 운영 시스템의 하드웨어적인 전원이 차단되지 않는다는 조건하 에서 영구히 보존됩니다.
VT_I4 cemHomeSuccess_Set ([in] VT_I4 Axis, [in] VT_I4 IsSuccess) 원점 복귀 완료 상태를 강제로 설정합니다. 설정된 원점 보귀 완료 상태는 모션 운영 시스템의 하드웨어적인 전원이 차단되지 않는다는 조건 하에서는 영구히 보존됩니다.
r VT_I4 cemHomeIsBusy ([in] VT_I4 Axis, [out] VT_PI4 IsBusy) 원점 복귀 이송 상태를 반환합니다. 이 함수를 통해 원점 복귀가 진행 중인지를 판단할 수 있습니다.
r VT_I4 cemHomeWaitDone ([in] VT_I4 Axis, [in] VT_I4 IsBlocking) 원점 복귀 완료 시점까지 대기합니다.

7.3.4 함수 설명

NAME

cemHomeConfig_Set / cemHomeConfig_Get

- 원점 복귀 환경 설정 및 설정 값 확인

INFORMATION
1 Home Return
! VC++ (6, 7, 8)/VB
BCB/Delphi
: Level 4
K 다소 주의

원점 복귀 함수의 전체 환경 설정을 하는 함수입니다. 매개변수 및 관련 내용을 반드시 숙지합니다.

SYNOPSIS

r VT_I4 cemHomeConfig_Set ([in] VT_I4 Axis, [in] VT_I4 HomeMode, [in] VT_I4 Dir, [in] VT_I4 EzCount, [in] VT_R8 EscDist, [in] VT_R8 Offset)

r VT_I4 cemHomeConfig_Get ([in] VT_I4 Axis, [out] VT_PI4 HomeMode, [out] VT_PI4 Dir, [out] VT_PI4 EzCount, [out] VT_PR8 EscDist, [out] VT_PR8 Offset)

DESCRIPTION

cemHomeConfig_Set / cemHomeConfig_Get 함수는 원점 복귀에 관련된 여러 가지 환경을 설정 혹은 설정된 환경을 반환합니다.

PARAMETER

▶ ***Axis*** : 축 번호. 통합 축으로 관리되는 축 번호를 의미하며, 상수 값으로 0 (Zero Based) 이상, (최대 통합 축 개수 - 1) 이하의 값을 축 번호로 설정할 수 있습니다.

▶ ***HomeMode*** : 원점복귀 모드 번호를 설정 혹은 반환합니다. 앞서 설명한 바와 같이 ㈜커미조아 모션 컨트롤러는 13 가지(0 ~ 12)의 다양한 원점복귀 모드를 제공합니다.

▶ ***Dir*** : 원점복귀 진행 방향을 설정 혹은 반환합니다.

Value	Meaning
0 (cemDIR_N)	(-) 방향 => Negative direction.
1 (cemDIR_P)	(+) 방향 => Positive direction.

▶ ***EzCount*** : 이 값은 ORG 신호 또는 EL 신호가 ON 이 된 후 실제로 원점 복귀 작업을 완료하는데 필요한 EZ Count 값을 0 ~ 15 사이의 값으로 설정 혹은 반환합니다. 이 값의 참조 여부는 원점 복귀 모드에 따라서 다릅니다. EzCount 0 은 1 회 카운트를 의미합니다.

▶ ***EscDist*** : 원점 탈출 거리를 설정 혹은 반환합니다. 원점 탈출 거리는 '자동 원점 검색 기능' 에 사용되며, 최소 '1' 이상의 값으로 설정해야 정상 적인 '자동 원점 검색' 기능을 통해 원점 복귀 이송을 수행합니다. 거리의 단위는 논리적 단위 거리를 사용합니다.

▶ ***Offset*** : 원점 복귀 완료 위치에서 일정 거리 이상을 상대 이송할 필요가 있을 경우, 그 값을 설정 혹은 반환합니다. 이것은 원점 복귀 완료 위치를 기준으로 추가 모션 이송을 의미합니다.

RETURN VALUE

Value	Meaning
음수	수행 실패. 자세한 내용은 '에러처리' 편을 참고합니다.
0 (ceERR_NONE)	수행 성공.

EXAMPLE

C/C++

```
#include “ceSDK.h”
#include “ceSDKDef.h”


void OnSetHomeConfig ()
{
    long nHomeMode = 0;             // 원점 복귀 모드 설정. 0 ~ 12 번 모드가 있습니다.
    long nHomeDir = cemDIR_N;       // 원점 복귀 방향. cemDIR_N: (-) 방향, cemDIR_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 ( cemHomeConfig_Set ( cemX1, nHomeMode, nHomeDir, nEzCount, fEscDist, fOffset ) != ceERR_NONE )
    {
            OutputDebugString ( “cemHomeConfig_Get has been failed” );
    }

    // cemHomeConf_Get() 함수로 설정되어있는 원점 복귀 환경 설정 정보를 반환합니다.
    // cemHomeConfig_Get ( cemX1, &nHomeMode, &nHomeDir, &nEzCount, &fEscDist, &fOffset );
}
```

Visual Basic

```
Private Sub OnSetHomeConfig ()

    Dim nHMode As Long              ‘ 원점 복귀 모드 설정.
    Dim nHomeDir As Long            ‘ 원점 복귀 방향.
    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
    nHmDir = cemDIR_N               ‘ 원점 복귀 (-)방향.
    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 cemHomeConfig_Set ( cemX1, nHomeMode, nHmDir, nEzCount, fEscDist, fOffset ) <> ceERR_NONE Then
            MsgBox ( “cemHomeConfig_Get 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.
    nHmDir := cemDIR_N;      // 원점 복귀 방향. cemDIR_N: (-) 방향, cemDIR_P: (+) 방향
    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 cemHomeConfig_Set ( cemX1, nHomeMode, nHmDir, nEzCount, fEscDist, fOffset ) <> ceERR_NONE then
    begin
            ShowMessage ( 'cemHomeConfig_Get has been failed' );
    end;

end;
```

NAME

cemHomePosClrMode_Set /

cemHomePosClrMode_Get

- 원점 복귀 완료 후 위치 소거 모드 설정 및 설정 값 확인

INFORMATION	
1	Home Return
!	VC++ (6, 7, 8)/VB
	BCB/Delphi
:	Level 4
K	다소 주의

원점 복귀 후 발생할 수 있는 일정량의 위치 편차는 정밀한 모션 제어에서 매우 중요한 부분입니다.

SYNOPSIS

r VT_I4 cemHomePosClrMode_Set ([in] VT_I4 Axis, [in] VT_I4 PosClrMode)

r VT_I4 cemHomePosClrMode_Get ([in] VT_I4 Axis, [out] VT_PI4 PosClrMode)

DESCRIPTION

cemHomePosClrMode_Set / cemHomePosClrMode_Get 함수는 원점 복귀 이송 완료 후에 발생하는 모션 컨트롤러와 서보 드라이버간의 제어 편차에 의해 발생하는 입력 펄스(Feedback pulse) 처리 방법을 설정 및 설정 상태를 반환합니다.

PARAMETER

▶ ***Axis*** : 축 번호. 통합 축으로 관리되는 축 번호를 의미하며, 상수 값으로 0 (Zero Based) 이상, (최대 통합 축 개수 - 1) 이하의 값을 축 번호로 설정할 수 있습니다.

▶ ***PosClrMode*** : 원점복귀가 완료된 후에 Command 및 Feedback 위치가 클리어 되는 모드를 설정 혹은 반환합니다. PosClrMode 는 다음과 같습니다.

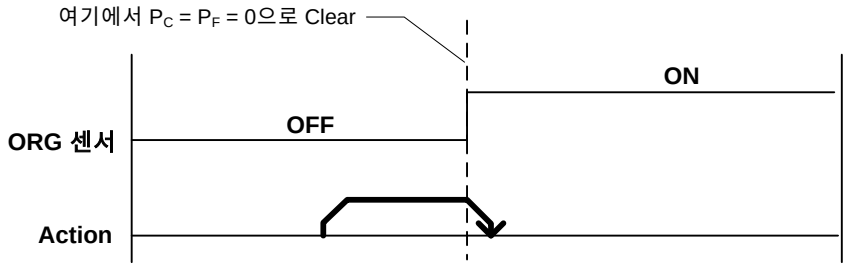
Value	Meaning
-1 (cemHPCM_DISABLE)	원점 복귀 클리어 모드를 사용하지 않습니다.
0 (cemHPCM_M0)	최종 완료 조건에 해당하는 외부 하드웨어 신호(ORG/EL/EZ)가 입력되는 순간에 Command 및 Feedback 의 위치를 '0'으로 소거합니다. 일정량의 Feedback 위치 편차를 보입니다.
1 (cemHPCM_M1)	원점 복귀 모드에 상관없이 하드웨어 신호의 입력 상태와 더불어 소프트웨어 적인 모션 완료 확인 동작을 포함한, 전체적인 원점복귀 작업이 모두 완료된 후에 Command 와 Feedback 위치를 모두 '0'으로 소거합니다. 일정량의 Feedback 위치 편차를 보입니다.
2 (cemHPCM_M2)	1 차적으로는 cemHPCM_M0 일 때와 동일하게 동작합니다. 단, 원점 복귀가 완료된 후에 Feedback 위치와 동일한 값으로 Command 위치를 설정합니다. 이를 통해 서보 드라이버에서 실제 이송한 위치에 대한 양을 Command 위치에 반영시켜서, 다음 모션 동작을 수행할 수 있도록 합니다.

RETURN VALUE

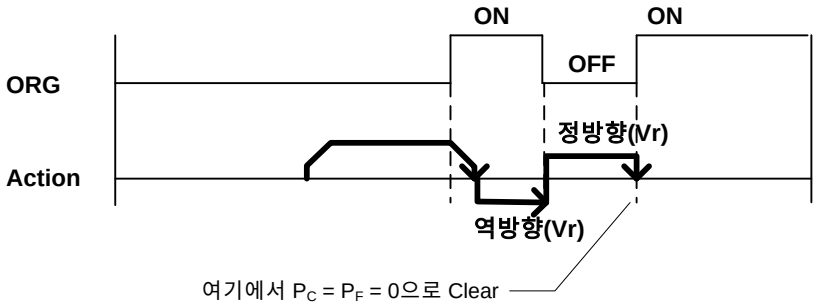
Value	Meaning
음수	수행 실패. 자세한 내용은 '에러처리' 편을 참고합니다.
0 (ceERR_NONE)	수행 성공.

REFERENCE

□ PosClrMode 가 0(cemHPCM_M0) 또는 2(cemHPCM_M2)로 하면 최종 단계에서 감속 없이 정지하는 원점복귀 모드(1, 2, 4, 6, 7)에서는 Command 위치가 0 인 상태에서 원점복귀가 완료되고, 나머지 모드에서는 감속을 시작 할 때 위치가 0 으로 클리어 되며, 감속 하는 동안에 이송한 양만큼 위치가 증가 또는 감소하게 됩니다. 아래의 두 그림은 최종 단계에서 감속을 하는 원점복귀 모드 0 번과 즉시 정지를 하는 모드 1 번에서의 동작을 예로 든 것입니다.



[원점복귀 모드 0에서의 동작]



[원점복귀 모드 1에서의 동작]

□ 서보 모터를 사용하는 경우에는 서보 드라이버의 제어 지연 시간 때문에 원점 복귀 완료 후 모터의 실제 위치는 약간의 편차가 발생할 수 있으며, 이는 원점복귀의 오차를 유발하게 됩니다. 그런데 PosClrMode 가 0(cemHPCM_M0) 또는 1(cemHPCM_M1)인 경우에는 모션컨트롤러의 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 모드가 2(cemHPCM_M2)입니다.

□ 스텝 모터를 사용하는 경우에는 cemHPCM_M2 를 사용하면 안됩니다.

서보 모터를 사용하는 경우에는 cemHPCM_M2 로 하였을 때 가장 정확한 원점 복귀 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. 경우에 따라서는 원점 복귀 완료 후에 절대 좌표 '0'으로 이송하는 명령이 필요할 수 있습니다.

EXAMPLE

C/C++

```
#include “ceSDK.h”
#include “ceSDKDef.h”

void OnSetHomePosClrMode ()
{
    long nPosClrMode = 0;      // 원점 복귀 위치 소거 모드 정보.

    /* 원점 복귀 완료 후 위치 소거 모드 */
    // -1 (cemHPCM_DISABLE) : 원점 복귀 위치 소거 모드 사용 않함.
    // 0 (cemHPCM_M0) : 원점 복귀 완료 HW 신호 ON 시 위치 소거.
    // 1 (cemHPCM_M1) : HW 신호를 포함한 전체 원점 복귀 작업 완료 후 위치 소거.
    // 2 (cemHPCM_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 (cemHomePosClrMode_Get ( cemX1, &nPosClrMode ) == ceERR_NONE )
    {
            if ( nPosClrMode != cemHPCM_M2 )
            {
                    cemHomePosClrMode_Set ( cemX1, cemHPCM_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 cemHomePosClrMode_Get ( cemX1, nPosClrMode ) = ceERR_NONE Then

            If  nPosClrMode <> cemHPCM_M2 Then
                    Call cemHomePosClrMode_Set ( cemX1, cemHPCM_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 cemHomePosClrMode_Get ( cemX1, @nPosClrMode ) = ceERR_NONE then
    begin
            if  nPosClrMode <> cemHPCM_M2 then
            begin
                    cemHomePosClrMode_Set ( cemX1, cemHPCM_M2 );
            end;
    end;

end;
```

NAME

cemHomeSpeedPattern_Set /

cemHomeSpeedPattern_Get

- 원점 복귀 속도 설정 및 반환

INFORMATION
1 Home Return
! VC++ (6, 7, 8)/VB
BCB/Delphi
: Level 4
K 다소 주의
속도설정은 논리적인 속도 단위가 적용됩니다.

SYNOPSIS

r VT_I4 cemHomeSpeedPattern_Set ([in] VT_I4 Axis, [in] VT_I4 SpeedMode, [in] VT_R8 Vel, [in] VT_R8 Accel, [in] VT_R8 Decel, [in] VT_R8 RevVel)

r VT_I4 cemHomeSpeedPattern_Get ([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

cemHomeSpeedPattern_Set / cemHomeSpeedPattern_Get 함수는 설정된 축의 원점 복귀 속도 모드와 작업 속도 및 가속도, 감속도, 역방향 속도를 설정 및 설정 상태를 반환합니다.

PARAMETER

▶ ***Axis*** : 축 번호. 통합 축으로 관리되는 축 번호를 의미하며, 상수 값으로 0 (Zero Based) 이상, (최대 통합 축 개수 – 1) 이하의 값을 축 번호로 설정할 수 있습니다.

▶ ***SpeedMode*** : cemHomeSpeedPattern_Set 함수의 인자이며, 원점복귀 시의 S-Curve 형 또는 선형 가감속, 가감속이 없는 모드등, 원점 복귀 속도 모드를 설정 혹은 반환합니다.

Value	Meaning
0 (cemSMODE_C)	CONSTANT 속도모드 => 가감속을 수행하지 않습니다.
1 (cemSMODE_T)	TRAPEZOIDAL 속도모드 => 사다리꼴 가감속을 수행합니다.
2 (cemSMODE_S)	S-CURVE 속도모드 => S-CURVE 가감속을 수행합니다.
-1 (cemSMODE_KEEP)	이전 속도 모드를 유지합니다.

▶ ***SpeedMode*** : cemHomeSpeedPattern_Get 함수의 인자이며, 설정된 속도 모드를 반환합니다.

Value	Meaning
0 (cemSMODE_C)	CONSTANT 속도모드 => 가감속을 수행하지 않습니다.
1 (cemSMODE_T)	TRAPEZOIDAL 속도모드 => 사다리꼴 가감속을 수행합니다.
2 (cemSMODE_S)	S-CURVE 속도모드 => S-CURVE 가감속을 수행합니다.

▶ ***Vel*** : 원점 복귀 작업 속도를 설정 혹은 반환합니다. 기호로는 Vwork 로 표기합니다.

▶ ***Accel*** : 원점 복귀 가속도를 설정 혹은 반환합니다.

▶ ***Decel*** : 원점 복귀 감속도를 설정 혹은 반환합니다.

▶ ***RevVel*** : Reverse Speed 를 설정 혹은 반환합니다. 원점 복귀 모드에 따라 Reverse Speed 를 필요로 하는 모드가 있습니다. 기호로는 Vr 로 표기합니다.

RETURN VALUE

Value	Meaning
음수	수행 실패. 자세한 내용은 '에러처리' 편을 참고합니다.
0 (ceERR_NONE)	수행 성공.

EXAMPLE

C/C++

```
#include “ceSDK.h”
#include “ceSDKDef.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.*/
     cemHomeSpeedPattern_Set (      cemX1,          // 대상 축 선택.
                                    cemSMODE_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.
    cemHomeSpeedPattern_Get ( cemX1, &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, fHec As Double, fRevVal As Double

    ‘ 0 번축의 속도 패턴을 S-Curve 로 설정하고,
    ‘ 작업 속도를 2000, 가속도를 10000, 감속도를 10000, Vr 을 10 으로 설정합니다.
     Call cemHomeSpeedPattern_Set ( cemX1, cemSMODE_S, 2000, 10000, 10000 )

    ‘ 설정되어있는 속도 패턴 및 작업속도, 가속도, 감속도를 반환합니다.
    Call cemHomeSpeedPattern_Get ( cemX1, nHSpdMode, fHVel, fHAcc, fHDec, fHRevVal );

End Sub
```

Delphi

```
procedure OnSetHomeSpeedPattern ();
var
    nHSpdMode : LongInt;                     // 원점 복귀 속도 환경 설정 정보.
    fHVel, fHAcc, fDec : Double;
```

```
begin
    { 0 번축의 속도 패턴을 S-Curve 로 설정하고,
     작업속도를 2000, 가속도를 10000, 감속도를 10000, Vr 을 10 으로 설정합니다. }
     cemHomeSpeedPattern_Set ( cemX1, cemSMODE_S, 2000, 10000, 10000, 10 );

    // 설정되어있는 속도 패턴 및 작업속도, 가속도, 감속도를 반환합니다.
    cemHomeSpeedPattern_Get ( cemX1, @nHSpdMode, @fHVel, @fHAcc, @fHDec, @fRevVal );

end;
```

NAME

cemHomeMove / cemHomeMoveStart

- 단축 원점 복귀 이송

INFORMATION	
1	Home Return
!	VC++ (6, 7, 8)/VB
	BCB/Delphi
:	Level 4
K	다소 위험

실제 이송이 진행되는 함수이므로, 사전에 반드시 안전을 확인합니다.

SYNOPSIS

r VT_I4 cemHomeMove ([in] VT_I4 Axis, [in] VT_I4 IsBlocking)

r VT_I4 cemHomeMoveStart ([in] VT_I4 Axis)

DESCRIPTION

cemHomeMove / cemHomeMoveStart 함수는 원점 복귀 작업을 수행합니다.

cemHomeMove 함수는 원점 복귀 이송이 완료되기 전까지 반환되지 않으며, cemHomeMoveStart 함수는 원점 복귀 이송을 시작시킨 후에 바로 반환됩니다.

PARAMETER

▶ *Axis* : 축 번호. 통합 축으로 관리되는 축 번호를 의미하며, 상수 값으로 0 (Zero Based) 이상, (최대 통합 축 개수 – 1) 이하의 값을 축 번호로 설정할 수 있습니다.

▶ *IsBlocking* : cemHomeMove 함수의 인자이며, 완료될 때까지 기다리는 동안 윈도우 메시지를 블록(Blocking)할 것인지를 결정합니다. 단, 쓰레드내에서 실행할 때는 이 값을 1(CE_TRUE)로 설정해 주어야 합니다.

Value	Meaning
0 (CE_FALSE)	윈도우 메시지를 블록(Blocking) 하지 않습니다. 따라서 해당 모션이 진행되는 동안 윈도우 이벤트를 처리합니다.
1 (CE_TRUE)	윈도우 메시지를 블록(Blocking) 합니다. 따라서 해당 모션이 완료되는 동안에는 윈도우 이벤트가 처리되지 않습니다.

RETURN VALUE

Value	Meaning
음수	수행 실패. 자세한 내용은 '에러처리' 편을 참고합니다.
0 (ceERR_NONE)	수행 성공.

REFERENCE

□ cemHomeMoveStart 함수를 사용하는 경우에는 cemHomeIsBusy, cemHomeWaitDone, cemHomeSuccess_Get 함수를 사용하여 모션의 완료를 확인할 수 있습니다.

□ cemHomeMove 함수를 사용하는 경우에는 내부적으로 루프를 수행하면서 모션이 완료되기를 기다리는데, 이때 블록 모드(Blocking Mode) 설정에 따라 윈도우 이벤트를 처리하는 방식이 달라집니다. 그러나 일반적으로 윈도우의 작업 쓰레드(Work Thread) 에서는 블록 모드를 사용하여, 함수 내부에서 지연 없이 쓰레드 내부의 작업에 집중할 수 있도록 설정하는 것이 바람직합니다.

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

□ 스텝 드라이버를 사용 중인 고객님들께서는 다음을 참조해 주십시오.
스텝 드라이버는 INP 출력이 없는 경우가 일반적인데, 고객님의 부주의나 잘못된 설정으로 INP 입력에 대한 설정이 Enable 로 되어 있을 경우 INP 입력이 스텝 드라이버를 통해 발생하지 않는 이유 때문에 모션 완료가 되지 않는 경우가 발생할 수 있습니다. 고객 여러분들께서는 스텝 드라이버 사용시에 이점을 주의해 주시기를 부탁 드립니다.

□ 서보 드라이버의 LSP, LSN 신호를 사용 중인 고객님께서는 다음을 참조해 주십시오.
서보 드라이버의 입력 신호 중 하나인 EL(End of Limit) 신호는 저희 ㈜ 커미조아 모션 컨트롤러뿐만 아니라 서보 드라이버에도 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 신호를 서보 드라이버 측에 전달하기 위한 용도로 사용됩니다.

그러나, 모션 소프트웨어에서 INP 설정이 되어 있는 경우 EL 신호가 검출 된 후에 일부 서보 드라이버에서는 진행 방향에서 정지 한 후 더 이상 움직이지 않는 상황이 발생하며, 이 상황에서 INP 신호가 출력되지 않아, 모션 이송이 완료되지 못하고, 명시적으로 STOP 명령을 통해 강제적으로 모션을 종료해야 되는 현상이 발생할 수 있습니다. 이 현상은 EL 모드를 통해 원점 복귀를 하는 상황에서도 발생할 수 있습니다.

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

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

EXAMPLE

C/C++

```
#include “ceSDK.h”
#include “ceSDKDef.h”

/************************************************************************
* 원점 복귀 환경을 설정합니다.
************************************************************************/
void OnSetHomeConfig ()
{
    /* 원점 복귀 환경을 원점 복귀 모드 = 0, 방향 = (-), Ez Count = 0,
    원점 탈출 거리 = 10, Offset = 0 으로 설정합니다. */
    cemHomeConfig_Set ( cemX1, 0, cemDIR_N, 0, 10, 0 );
    cemHomeConfig_Set ( cemY1, 0, cemDIR_N, 0, 10, 0 );

    /* 원점 복귀 속도 환경을 설정합니다. */
    cemHomeSpeedPattern_Set ( cemX1, cemSMODE_S, 1000, 10000, 10000, 10 );
    cemHomeSpeedPattern_Set ( cemY1, cemSMODE_S, 1000, 10000, 10000, 10 );
}


/************************************************************************
* 원점 복귀 이송을 수행합니다.
************************************************************************/
void OnHomeReturn ()
{
    long nIsHomming = CE_TRUE;       // 원점 복귀 진행 상태 정보

    /* cemHomeIsBusy() 명령을 통한 원점 복귀 완료 체크 */
    if ( cemHomeMoveStart ( cemX1 ) == ceERR_NONE )
    {
            While ( nIsHomming )
            {
                    cemHomeIsBusy ( cemX1, &nIsHomming );
                    // 0 (CE_FALSE) : 원점 복귀 이송 작업이 진행중이지 않습니다.
                    // 1 (CE_TRUE) : 원점 복귀 이송 작업이 진행중입니다.
            }
    }

    // cemHomeWaitDone() 명령을 통한 원점 복귀 완료 체크.
    if ( cemHomeMoveStart ( cemY1 ) == ceERR_NONE )
    {
            cemHomeWaitDone ( cemY1, CE_FALSE ); // 원점 복귀 완료 시까지 대기합니다.
    }

    // cemHomeMoveStart(), cemHomeWaitDone() 을 아래 코드로 대체 할 수 있습니다.
    // cemHomeMove ( cemY1, CE_FALSE );
}


/************************************************************************
* 원점 복귀 성공 여부를 확인합니다.
************************************************************************/
void OnGetHomeSuccess ()
```

```
{
    long nIsSuccess;

    // 원점 복귀 성공 여부를 확인합니다.
    cemHomeSuccess_Get ( cemX1, &nIsSuccess );

    if ( nIsSuccess == CE_TRUE )
    {
            // 원점 복귀가 성공했음을 의미합니다.
    }
}
```

Visual Basic

```
' ============================================
' 원점 복귀 환경을 설정합니다.
'============================================
Private Sub OnSetHomeConfig ()

    ' 원점 복귀 환경을 원점 복귀 모드 = 0, 방향 = (-), Ez Count = 0,
    ' 원점 탈출 거리 = 10, Offset = 0 으로 설정합니다.
    Call cemHomeConfig_Set ( cemX1, 0, cemDIR_N, 0, 10, 0 )
    Call cemHomeConfig_Set ( cemY1, 0, cemDIR_N, 0, 10, 0 )

    ' 원점 복귀 속도 환경을 설정합니다.
    Call cemHomeSpeedPattern_Set ( cemX1, cemSMODE_S, 1000, 10000, 10000, 10 )
    Call cemHomeSpeedPattern_Set ( cemY1, cemSMODE_S, 1000, 10000, 10000, 10 )

End Sub


' ============================================
' 원점 복귀 이송을 수행합니다.
' ============================================
Private OnHomeReturn ()

    Dim nIsHomming As Long              ' 원점 복귀 진행 상태 정보
    nIsHomming = CE_TRUE

    ' cemHomeIsBusy() 명령을 통한 원점 복귀 완료 체크
    If cemHomeMoveStart ( cemX1 ) = ceERR_NONE Then

            While ( nIsHomming = CE_TRUE )
                    ' 0 (CE_FALSE) : 원점 복귀 이송 작업이 진행중이지 않습니다.
                    ' 1 (CE_TRUE) : 원점 복귀 이송 작업이 진행중입니다.
                    Call cemHomeIsBusy ( cemX1, nIsHomming )
            Wend
    End If

    ' cemHomeWaitDone() 명령을 통한 원점 복귀 완료 체크.
    If cemHomeMoveStart ( cemY1 ) = ceERR_NONE Then
            Call cemHomeWaitDone ( cemY1, CE_FALSE )        ' 원점 복귀 완료 시까지 대기합니다.
    End If
```

```
    ' cemHomeMoveStart(), cemHomeWaitDone() 을 아래 코드로 대체 할 수 있습니다.
    ' cemHomeMove( cemY1, CE_FALSE )

End Sub


' ================================================
' 원점 복귀 성공 여부를 확인합니다.
' ================================================
Private Sub OnGetHomeSuccess ()

    Dim nIsSuccess As Long              ' 원점 복귀 성공 여부 정보.

    ' 원점 복귀 성공 여부를 확인합니다.
    Call cemHomeSuccess_Get ( cemX1, nIsSuccess )

    ' 원점 복귀 성공 메시지를 표시합니다.
    If nIsSuccess = CE_TRUE Then
            MsgBox ( "Home return success" )
    End If

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. }
    cemHomeConfig_Set ( cemX1, 0, cemDIR_N, 0, 10, 0 );
    cemHomeConfig_Set ( cemY1, 0, cemDIR_N, 0, 10, 0 );

    // 원점 복귀 속도 환경을 설정합니다.
    cemHomeSpeedPattern_Set ( cemX1, cemSMODE_S, 1000, 10000, 10000, 10 );
    cemHomeSpeedPattern_Set ( cemY1, cemSMODE_S, 1000, 10000, 10000, 10 );

end;


// ***************************************************************************
// 원점 복귀 이송을 수행합니다.
// ***************************************************************************
procedure OnHomeReturn ();
var
    nIsHomming : LongInt                // 원점 복귀 진행 상태 정보

begin
    // cemHomeIsBusy() 명령을 통한 원점 복귀 완료 체크
    if cemHomeMoveStart ( cemX1 ) = ceERR_NONE then
    begin
            nIsHomming := CE_TRUE;
            while nIsHomming = CE_TRUE do
```

```
        begin
            { 0 (CE_FALSE) : 원점 복귀 이송 작업이 진행중이지 않습니다.
             1 (CE_TRUE) : 원점 복귀 이송 작업이 진행중입니다. }
            cemHomeIsBusy ( cemX1, @nIsHomming );
        end;
    end;

    // cemHomeWaitDone() 명령을 통한 원점 복귀 완료 체크.
    if cemHomeMoveStart ( cemY1 ) = ceERR_NONE then
    begin
        cemHomeWaitDone ( cemY1, CE_FALSE ); // 원점 복귀 완료 시까지 대기합니다.
    end

    // cemHomeMoveStart(), cemHomeWaitDone() 을 아래 코드로 대체 할 수 있습니다.
    // cemHomeMove( cemY1, CE_FALSE );

end;


// ********************************************************************
// 원점 복귀 성공 여부를 확인합니다.
// ********************************************************************
procedure OnGetHomeSuccess ();
var
    nIsSuccess : LongInt                    // 원점 복귀 성공 여부 정보.

begin
    // 원점 복귀 성공 여부를 확인합니다.
    cemHomeSuccess_Get ( cemX1, @nIsSuccess );

    // 원점 복귀 성공 메시지를 표시합니다.
    if nIsSuccess = CE_TRUE then
    begin
        ShowMessage ( 'Home return success' );
    end;

end;
```

NAME

cemHomeSuccess_Get / cemHomeSuccess_Set

- 이전 원점 복귀 완료 확인 및 설정

INFORMATION
1 Home Return
! VC++ (6, 7, 8)/VB
BCB/Delphi
: Level 4
J 위험 요소 없음

원점 복귀 성공 여부는 응용 프로그램의 종료와 관계없이 유지됩니다.

SYNOPSIS

VT_I4 cemHomeSuccess_Get ([in] VT_I4 Axis, [out] VT_PI4 IsSuccess)

VT_I4 cemHomeSuccess_Set ([in] VT_I4 Axis, [in] VT_I4 IsSuccess)

DESCRIPTION

cemHomeSuccess_Get 함수는 이 함수가 호출되기 이전에 원점 복귀 이송이 성공적으로 완료되었는지 확인하는 함수입니다.

cemHomeSuccess_Set 함수는 원점 복귀의 성공 여부에 대한 상태를 강제로 설정하는 함수입니다. 일반적으로는 원점 복귀 완료 상태는 원점 복귀의 실제 수행에 의해서 설정됩니다. 그러나 필요한 경우에 강제로 설정할 수 있습니다.

PARAMETER

▶ ***Axis*** : 축 번호. 통합 축으로 관리되는 축 번호를 의미하며, 상수 값으로 0 (Zero Based) 이상, (최대 통합 축 개수 - 1) 이하의 값을 축 번호로 설정할 수 있습니다.

▶ ***IsSuccess*** : cemHomeSuccess_Get 함수의 인자이며, 이 함수가 호출된 시점을 기준으로 이전에 원점복귀가 성공적으로 완료된 상태인지를 알려 주는 매개변수입니다.

Value	Meaning
0 (CE_FALSE)	지정한 축의 원점 복귀 이송이 진행 중이거나 혹은 비정상적으로 완료되었음을 나타냅니다.
1 (CE_TRUE)	지정한 축의 원점 복귀 이송이 정상적으로 완료된 상태입니다.

▶ ***IsSuccess*** : cemHomeSuccess_Set 함수의 인자이며, 원점 복귀의 성공 여부에 대한 상태를 강제로 설정합니다.

Value	Meaning
0 (CE_FALSE)	지정한 축을 원점 복귀 이송이 진행 중인 상태 혹은 비정상적으로 완료된 상태로 설정합니다.
1 (CE_TRUE)	지정한 축을 원점 복귀 이송이 정상적으로 완료된 상태로 설정합니다.

RETURN VALUE

Value	Meaning
음수	수행 실패. 자세한 내용은 '에러처리' 편을 참고합니다.
0 (ceERR_NONE)	수행 성공.

REFERENCE

□ 원점복귀의 성공 여부에 대한 상태는 응용 프로그램이 종료되어도 그대로 유지됩니다. 따라서 다시 응용 프로그램이 시작되면 이전에 원점복귀를 정상적으로 수행했었는지를 알 수가 있습니다. 단, 제어 시스템의 하드웨어적인 전원이 차단되거나 재 시작(Rebooting) 되면 원점 복귀가 완료되지 않은 상태로 설정됩니다. 따라서 cemHomeSuccess_Get 함수의 이러한 특성을 활용하면 프로그램이 종료되었다가 다시 실행될 때 이전의 원점복귀 수행 여부를 확인할 수가 없어서 매번 원점복귀를 수행해야 했던 불편을 보완할 수 있습니다.

□ IsSuccess 매개변수가 FALSE 인 경우는 원점복귀가 진행 중인 경우를 의미할 수도 있고 비정상적으로 종료되었음을 의미할 수도 있습니다. 따라서 cemHomeMoveStart 함수를 사용한 경우에는 먼저 cemHomeIsBusy 함수나 cemSxWaitDone 함수를 선행하여 완료를 확인한 후에 cemHomeSuccess_Get 를 사용하여 성공 여부를 확인하는 것이 정석입니다.

□ 이전에 원점복귀가 성공적으로 수행되었더라도 해당 축의 원점복귀를 다시 시작하면 원점복귀가 완료되지 않은 상태로 설정됩니다.

EXAMPLE

```
//* cemHomeMove / cemHomeMoveStart 예제를 참고하여 주시기 바랍니다.
```

NAME

cemHomeIsBusy

- 원점 복귀 모션 진행 상태 확인

INFORMATION	
1	Home Return
!	VC++ (6, 7, 8)/VB
	BCB/Delphi
:	Level 4
J	위험 요소 없음

SYNOPSIS

r VT_I4 cemHomeIsBusy ([in] VT_I4 Axis, [out] VT_PI4 IsBusy)

DESCRIPTION

지정한 축이 현재 원점 복귀를 진행 중인지를 IsBusy 버퍼를 통하여 반환합니다.

PARAMETER

▶ ***Axis*** : 축 번호. 통합 축으로 관리되는 축 번호를 의미하며, 상수 값으로 0 (Zero Based) 이상, (최대 통합 축 개수 – 1) 이하의 값을 축 번호로 설정할 수 있습니다.

▶ ***IsBusy*** : 현재 원점 복귀가 진행 중인지를 알려 주는 값을 반환 받을 버퍼. 이 값에 반환되는 값의 의미는 다음과 같습니다.

Value	Meaning
0 (CE_FALSE)	지정한 축은 현재 원점복귀가 진행 중이지 않습니다.
1 (CE_TRUE)	지정한 축은 현재 원점복귀를 진행하고 있습니다.

RETURN VALUE

Value	Meaning
음수	수행 실패. 자세한 내용은 '에러처리' 편을 참고합니다.
0 (ceERR_NONE)	수행 성공.

REFERENCE

□ ceSDK 라이브러리에서는 cemSxIsDone 과 같이 일반적으로 모션이 진행 중이냐 아니면 정지해 있느냐를 확인할 때, 진행 중(Busy)을 확인하기 보다는 완료(Done)를 확인하는 방식을 채택합니다. 그러나 원점복귀에서는 cemHomeSuccess_Get 함수와 혼동될 소지가 있어서 cemHomeIsDone 함수 대신에 cemHomeIsBusy 함수를 제공하게 되었습니다.

□ cemHomeIsBusy 함수의 IsBusy 버퍼에 0(CE_FALSE) 값이 반환되면 원점복귀가 완료되었음을 의미하지만 성공 여부는 알 수가 없습니다. 예를 들어 원점복귀 진행 중에 Limit 또는 Alarm 등과 같은 에러에 의해서 정지되었거나, Stop 함수에 의해서 강제로 정지되었을 때도 IsBusy 에는 0(CE_FALSE)

값이 반환됩니다. 따라서 cemHomeIsBusy 함수를 이용하여 원점 복귀 이송이 완료되었음을 확인한 후에는 cemHomeSuccess_Get 함수를 사용하여 원점복귀의 성공 여부를 확인하여 각각의 상황에 대한 처리를 해주는 것이 바람직합니다.

EXAMPLE

```
//* cemHomeMove / cemHomeMoveStart 예제를 참고하여 주시기 바랍니다.
```

NAME

cemHomeWaitDone

- 원점 복귀 완료 확인

INFORMATION	
1	Home Return
!	VC++ (6, 7, 8)/VB
	BCB/Delphi
:	Level 4
J	위험 요소 없음

SYNOPSIS

r VT_I4 cemHomeWaitDone ([in] VT_I4 Axis, [in] VT_I4 IsBlocking)

DESCRIPTION

지정한 축의 원점 복귀 이송 작업의 완료 시점까지 기다립니다.

PARAMETER

▶ *Axis* : 축 번호. 통합 축으로 관리되는 축 번호를 의미하며, 상수 값으로 0 (Zero Based) 이상, (최대 통합 축 개수 - 1) 이하의 값을 축 번호로 설정할 수 있습니다.

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

Value	Meaning
0 (CE_FALSE)	윈도우 메시지를 블록(Blocking) 하지 않습니다. 따라서 해당 모션이 진행되는 동안 윈도우 이벤트를 처리합니다.
1 (CE_TRUE)	윈도우 메시지를 블록(Blocking) 합니다. 따라서 해당 모션이 완료되는 동안에는 윈도우 이벤트가 처리되지 않습니다.

RETURN VALUE

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

□ 스텝 드라이버를 사용 중인 고객님들께서는 다음을 참조해 주십시요.
스텝 드라이버는 INP 출력이 없는 경우가 일반적인데, 고객님의 부주의나 잘못된 설정으로 INP 입력에 대한 설정이 Enable 로 되어 있을 경우 INP 입력이 스텝 드라이버를 통해 발생하지 않는 이유 때문에

모션 완료가 되지 않는 경우가 발생할 수 있습니다. 고객 여러분들께서는 스텝 드라이버 사용시에 이점을 주의해주시기를 부탁드립니다.

□ 서보 드라이버의 LSP, LSN 신호를 사용 중인 고객님께서는 다음을 참조해 주십시요.
서보 드라이버의 입력 신호 중 하나인 EL(End of Limit) 신호는 저희 ㈜ 커미조아 모션 모듈 뿐만 아니라 서보드라이버에도 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 신호를 서보 드라이버 측에 전달하기 위한 용도로 사용됩니다.

그러나, 모션 소프트웨어에서 INP 설정이 되어 있는 경우 EL 신호가 검출 된 후에 일부 서보 드라이버에서는 진행 방향에서 정지한 후 더 이상 움직이지 않는 상황이 발생하며, 이 상황에서 INP 신호가 출력되지 않아, 모션 이송이 완료되지 못하고, 명시적으로 STOP 명령을 통해 강제적으로 모션을 종료해야 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

```
//* cemHomeMove / cemHomeMoveStart 예제를 참고하여 주시기 바랍니다.
```

Chapter

8

Advanced Motion Control

커미조아의 모션 제어는 기본 모션 제어 기능에만 국한되지 않습니다. 더 높은 기능과 더 안정적인 기능이 커미조아의 제품을 대변해 주고 있습니다. 고급 모션 제어에서는 그 기능에 있어 다양한 기능을 표현하고 있지만, 기능의 응용에서 비로소 진정한 고급 모션제어를 구현하실 수 있습니다.

이 단원에서는 고급 모션 제어에 관련된 함수들을 소개합니다. 고급 모션 제어는 속도 및 위치 오버라이딩 및 Master / Slave 동기 제어 등의 기능으로 구성되어 있습니다. 속도 및 위치 오버라이딩 기능을 통해 모션이 진행되고 있는 중에 작업 속도 및 목표 좌표를 변경할 수 있으며, Master / Slave 동기 제어는 두개의 모터 축을 하나의 모터처럼 완전한 동기 제어를 가능하게 합니다.

8 고급 모션 제어 편

8.1 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 MoveStart 나 MoveToStart 와 같이 In-Position 모션을 수행하고 있는 중에 목표 거리 또는 목표 좌표를 수정하는 것을 의미합니다.

일반적인 모션 구동에서 많이 요구되는 기능은 아니지만, 그 기능면에 있어 타사의 다른 오버라이드 기능보다 월등한 기능과 성능, 그리고 정확성을 제공하고 있습니다.

8.1.1 함수 요약

㈜ 커미조아의 속도 및 위치 오버라이딩에 관련된 함수는 다음과 같습니다.

Summary of Functions
r VT_I4 cemOverrideSpeedSet ([in] VT_I4 Axis) 단축 모션 작업이 진행되고 있는 중에 속도를 변경합니다.
r VT_I4 cemOverrideMove ([in] VT_I4 Axis, [in] VT_R8 NewDistance, [in] VT_I4 IsHardApply, [out] VT_PI4 AppliedState) 단축 상대 좌표 이송 모션에 대하여, 상대 좌표상의 목표 논리 거리 값을 수정합니다.
r VT_I4 cemOverrideMoveTo ([in] VT_I4 Axis, [in] VT_R8 NewPosition, [in] VT_I4 IsHardApply, [out] VT_PI4 AppliedState) 단축 절대 좌표 이송 모션에 대하여, 절대 좌표상의 목표 논리 거리 값을 수정합니다.

참고적으로, 속도 및 위치 오버라이딩 함수는 모션이 종료된 시점에서 수행되는 함수가 아닌, 모션의 이송 중에 적용되는 함수이기 때문에, 이전에 수행된 모션 명령이 cemSxMove 나 cemSxMoveTo 함수같은 모션 이송의 목표 위치에 대한 완료를 동반하여 반환되는 함수에는 적용되지 않습니다. 따라서 오버라이딩 함수의 이전 함수는 cemSxMoveStart 나 cemSxMoveToStart 함수와 같이 이송 명령이 설정된 후에 응용 프로그램의 제어를 즉시 반환 받고, 속도 및 위치 오버라이딩을 수행하게 됩니다.

8.1.2 함수 설명

NAME

cemOverrideSpeedSet

- 단축 속도 오버라이딩

INFORMATION
1 Overriding
! VC++ (6, 7, 8)/VB
! BCB/Delphi
: Level 5
K 다소 주의

함수의 호출 전에 속도 지령 함수를 통해 변경하고자 하는 속도를 설정합니다.

SYNOPSIS

r VT_I4 cemOverrideSpeedSet ([in] VT_I4 Axis)

DESCRIPTION

cemOverrideSpeedSet 함수는 단축 모션이 진행되고 있는 중에 속도를 변경 하고자 할 때 사용하는 함수입니다.
속도를 오버라이딩 하기 위해서는 먼저 속도 환경을 설정하는 함수(cemCfgSpeedPattern_Set 또는 cemSxSpeed_Set)를 통하여 변경하고자 하는 속도 또는 가속도 값을 설정하고나서 속도 오버라이드 함수를 수행해야 합니다.

PARAMETER

▶ ***Axis*** : 축 번호. 통합 축으로 관리되는 축 번호를 의미하며, 상수 값으로 0 (Zero Based) 이상, (최대 통합 축 개수 - 1) 이하의 값을 축 번호로 설정할 수 있습니다.

RETURN VALUE

Value	Meaning
음수	수행 실패. 자세한 내용은 '에러처리' 편을 참고합니다
0 (ceERR_NONE)	수행 성공

REFERENCE

□ 보간 작업을 수행하는 경우에는 속도 오버라이딩을 사용할 수 없습니다.

EXAMPLE

C/C++

```
#include “ceSDK.h”
#include “ceSDKDef.h”

/* 다음 예제는 단축 속도 이송 수행 중에 속도 오버라이드 함수를 통해
속도를 변경하는 예제 입니다. */

void OnMove ()
{
    // 속도 환경을 설정합니다.
    cemCfgSpeedPattern_Set ( cemX1, cemSMODE_S, 1000, 10000, 10000 );

    // 단축 속도 이송을 수행합니다.
    cemSxMoveStart ( cemX1, cemDIR_P );
}

void OnOverrideSpeedHigh ()
{
    // 속도 오버라이드 할 축 및 속도를 지정합니다.
    cemCfgSpeedPattern_Set ( cemX1, cemSMODE_KEEP, 2000, 10000, 10000 );
    // 또는 cemSxSpeedPattern_Set ( cemX1, cemSMODE_KEEP, 200, 100, 100 );

    // 진행 중인 이송 작업에 대해 속도를 오버라이드 합니다.
    cemOverrideSpeedSet ( cemX1 );
}


void OnOverrideSpeedLow ()
{
    // 속도 오버라이드 할 축 및 속도를 지정합니다.
    cemCfgSpeedPattern_Set ( cemX1, cemSMODE_KEEP, 500, 10000, 10000 );
    // 또는 cemSxSpeedPattern_Set ( cemX1, cemSMODE_KEEP, 50, 100, 100 );

    // 진행 중인 이송 작업에 대해 속도를 오버라이드 합니다.
    cemOverrideSpeedSet ( cemX1 );
}
```

Visual Basic

```
‘ 다음 예제는 단축 속도 이송 수행 중에 속도 오버라이드 함수를 통해
‘ 속도를 변경하는 예제 입니다.

Private Sub OnMove ()

    ‘ 속도 환경을 설정합니다.
    Call cemCfgSpeedPattern_Set ( cemX1, cemSMODE_S, 1000, 10000, 10000 )

    ‘ 단축 속도 이송을 수행합니다.
    Call cemSxMoveStart ( cemX1, cemDIR_P )
```

```
End Sub

Private Sub OnOverrideSpeedHigh ()

    ‘ 속도 오버라이드 할 축 및 속도를 지정합니다.
    Call cemCfgSpeedPattern_Set ( cemX1, cemSMODE_KEEP, 2000, 10000, 10000 )
    ‘ 또는 cemSxSpeedPattern_Set ( cemX1, cemSMODE_KEEP, 200, 100, 100 )

    ‘ 진행 중인 이송 작업에 대해 속도를 오버라이드 합니다.
    Call cemOverrideSpeedSet ( cemX1 )

End Sub


Private Sub OnOverrideSpeedLow ()

    ‘ 속도 오버라이드 할 축 및 속도를 지정합니다.
    Call cemCfgSpeedPattern_Set ( cemX1, cemSMODE_KEEP, 500, 10000, 10000 )
    ‘ 또는 cemSxSpeedPattern_Set ( cemX1, cemSMODE_KEEP, 50, 100, 100 )

    ‘진행 중인 이송 작업에 대해 속도를 오버라이드 합니다.
    Call cemOverrideSpeedSet ( cemX1 )

End Sub
```

```
Delphi

{ 다음 예제는 단축 속도 이송 수행 중에 속도 오버라이드 함수를 통해
 속도를 변경하는 예제 입니다. }

procedure OnMove ();
begin
    // 속도 환경을 설정합니다.
    cemCfgSpeedPattern_Set ( cemX1, cemSMODE_S, 1000, 10000, 10000 );

    // 단축 속도 이송을 수행합니다.
    cemSxMoveStart ( cemX1, cemDIR_P );
end;


procedure OnOverrideSpeedHigh ()
begin
    // 속도 오버라이드 할 축 및 속도를 지정합니다.
    cemCfgSpeedPattern_Set ( cemX1, cemSMODE_KEEP, 2000, 10000, 10000 );
    // 또는 cemSxSpeedPattern_Set ( cemX1, cemSMODE_KEEP, 200, 100, 100 );

    // 진행 중인 이송 작업에 대해 속도를 오버라이드 합니다.
    cemOverrideSpeedSet ( cemX1 );

end;


procedure OnOverrideSpeedLow ();
begin
```

```
    // 속도 오버라이드 할 축 및 속도를 지정합니다.
    cemCfgSpeedPattern_Set ( cemX1, cemSMODE_KEEP, 500, 10000, 10000 );
    // 또는 cemSxSpeedPattern_Set ( cemX1, cemSMODE_KEEP, 50, 100, 100 );

    //진행 중인 이송 작업에 대해 속도를 오버라이드 합니다.
    cemOverrideSpeedSet ( cemX1 );

end;
```

NAME

cemOverrideMove

- 단축 상대 위치 오버라이드 이송

INFORMATION
1 Overriding
! VC++ (6, 7, 8)/VB
BCB/Delphi
: Level 5
K 다소 위험
실제 이송이 진행되는 함수이므로, 사전에 반드시 안전을 확인합니다.

SYNOPSIS

r VT_I4 cemOverrideMove ([in] VT_I4 Axis, [in] VT_R8 NewDistance, [in] VT_I4 IsHardApply, [out] VT_PI4 AppliedState)

DESCRIPTION

cemSxMoveStart 이송 함수를 통하여 수행되는 상대 좌표 In-position 모션에 대하여 목표 상대 좌표 값, 즉 목표 거리 값을 오버라이딩 합니다.

PARAMETER

▶ ***Axis*** : 축 번호. 통합 축으로 관리되는 축 번호를 의미하며, 상수 값으로 0 (Zero Based) 이상, (최대 통합 축 개수 - 1) 이하의 값을 축 번호로 설정할 수 있습니다.

▶ ***NewDistance*** : 새로운 목표 거리 값을 지정합니다. 이 값의 기준 위치는 오버라이드 하고자 하는 대상이 되는 cemSxMoveStart 이송 작업에서 사용한 기준점과 같습니다. 즉, 새로운 목표 거리는 cemSxMoveStart 함수를 실행하기 바로 직전의 위치를 기준으로 계산하여야 합니다.

▶ ***IsHardApply*** : 위치 오버라이드 불가 시점에서도 오버라이드 설정을 강제로 적용합니다.

▶ ***AppliedState*** : cemOverrideMove() 함수의 적용 성공/실패 여부를 반환 합니다.

Value	Meaning
0 (CE_FALSE)	모션 에러가 발생하였거나 이미 이송이 완료되어 위치 오버라이드가 적용되지 않음.
1 (CE_TRUE)	위치 오버라이드가 적용됨.

RETURN VALUE

Value	Meaning
음수	수행 실패. 자세한 내용은 '에러처리' 편을 참고합니다.
0 (ceERR_NONE)	수행 성공.

SEE ALSO

cemSxMoveStart

REFERENCE

□ 위치 오버라이드를 수행하려는 시점에 이미 이송이 완료되어 버린 경우에는 위치 오버라이드는 무시되고 AppliedState 매개변수는 '0' 을 반환합니다. 따라서 사용자는 AppliedState 매개변수의 반환값이 '0'인 경우에는 이미 이송이 완료되어 오버라이드가 적용되지 않은 것으로 인지하여야 하며, 그럼에도 불구하고 목표 좌표를 수정해야 하는 경우에는 cemSxMove 또는 cemSxMoveTo 함수를 추가적으로 수행해야 합니다. 이러한 경우에는 오버라이드라는 개념 보다는 추가 이송의 개념을 의미합니다.

EXAMPLE

C/C++

```
#include "ceSDK.h"
#include "ceSDKDef.h"

/* 다음 예제는 모션이 정지해 있는 경우 단축 이송 명령을 수행하고,
이송 중인 경우에는 목표 위치를 오버라이드 합니다. */

/* 'nIsAbsMode' 는 상대 좌표 이송 / 절대 좌표 이송 설정 변수라 가정합니다.
0 (CE_FALSE) : 상대 좌표 이송 / 1 (CE_TRUE) : 절대 좌표 이송 */


void OnOverrideMove ()
{
    long nAppliedState;         // 위치 오버라이드 수행 여부 정보
    long nIsStopped;            // 이송 상태 확인

    cemSxIsDone ( cemX1, &nIsStopped );
    // 0 (CE_FALSE ) : 모션 이송 중
    // 1 (CE_TRUE ) : 모션 정지 상태

    if ( nIsStopped == CE_TRUE )        // 정지 상태 이므로 단축 이송을 수행합니다.
    {
            if ( nIsAbsMode == CE_TRUE )
            {
                    cemSxMoveToStart ( cemX1, 10000 );
            }
            else
            {
                    cemSxMoveStart ( cemX1, 10000 );
            }
    }
    else    // 이송중이므로 목표 위치를 오버라이드 합니다.
    {
            /* IsHardApply 인자를 CE_TRUE 로 할 경우에는 모션이 종료된 상태에서도 강제로 이송시킵니다. */
            if ( nIsAbsMode == CE_TRUE )
            {
                    cemOverrideMoveTo ( cemX1, 20000, CE_TRUE, &nAppliedState );
```

```
            }
            else
            {
                cemOverrideMove ( cemX1, 20000, CE_TRUE, &nAppliedState );
            }
    }
}
```

Visual Basic

```
' 다음 예제는 모션이 정지해 있는 경우 단축 이송 명령을 수행하고,
' 이송 중인 경우에는 목표 위치를 오버라이드 합니다.

' 'nIsAbsMode' 는 상대 좌표 이송 / 절대 좌표 이송 설정 변수라 가정합니다.
' 0 (CE_FALSE) : 상대 좌표 이송 / 1 (CE_TRUE) : 절대 좌표 이송


Private Sub OnOverrideMove ()

    Dim nAppliedState As Long          ' 위치 오버라이드 수행 여부 정보
    Dim nIsStopped As Long             ' 이송 상태 확인

    Call cemSxIsDone ( cemX1, nIsStopped )
    ' 0 (CE_FALSE ) : 모션 이송 중
    ' 1 (CE_TRUE ) : 모션 정지 상태

    If nIsStopped = CE_TRUE Then     ' 정지 상태 이므로 단축 이송을 수행합니다.
        If nIsAbsMode = CE_TRUE Then
            Call cemSxMoveToStart ( cemX1, 10000 )
        Else
            Call cemSxMoveStart ( cemX1, 10000 )
        End If

    Else     ' 이송중이므로 목표 위치를 오버라이드 합니다.

        ' IsHardApply 인자를 CE_TRUE 로 할 경우에는 모션이 종료된 상태에서도 강제로 이동시킵니다.
        If nIsAbsMode = CE_TRUE Then
            Call cemOverrideMoveTo ( cemX1, 20000, CE_TRUE, nAppliedState )
        Else
            Call cemOverrideMove ( cemX1, 20000, CE_TRUE, nAppliedState )
        End If
    End If

End Sub
```

Delphi

```
{ 다음 예제는 모션이 정지해 있는 경우 단축 이송 명령을 수행하고,
 이송 중인 경우에는 목표 위치를 오버라이드 합니다. }

{ 'nIsAbsMode' 는 상대 좌표 이송 / 절대 좌표 이송 설정 변수라 가정합니다.
```

```
 0 (CE_FALSE) : 상대 좌표 이송 / 1 (CE_TRUE) : 절대 좌표 이송 }


procedure OnOverrideMove ();
var
    nAppliedState : LongInt;              // 위치 오버라이드 수행 여부 정보
    nIsStopped : LongInt;                 // 이송 상태 확인

    cemSxIsDone ( cemX1, @nIsStopped );
    // 0 (CE_FALSE ) : 모션 이송 중
    // 1 (CE_TRUE ) : 모션 정지 상태

    if nIsStopped = CE_TRUE then      // 정지 상태 이므로 단축 이송을 수행합니다.
    begin
            if nIsAbsMode = CE_TRUE then
            begin
                    cemSxMoveToStart ( cemX1, 10000 );
            end;

            else
            begin
                     cemSxMoveStart ( cemX1, 10000 );
            end;
    end;

    else      // 이송중이므로 목표 위치를 오버라이드 합니다.
    begin
            // IsHardApply 인자를 CE_TRUE 로 할 경우에는 모션이 종료된 상태에서도 강제로 이동시킵니다.
            if nIsAbsMode = CE_TRUE then
            begin
                    cemOverrideMoveTo ( cemX1, 20000, CE_TRUE, @nAppliedState );
            end;

            else
            begin
                    cemOverrideMove ( cemX1, 20000, CE_TRUE, @nAppliedState );
            end;
    end;

end;
```

NAME

cemOverrideMoveTo

- 단축 절대 위치 오버라이드 이송

INFORMATION
1 Overriding
! VC++ (6, 7, 8)/VB
BCB/Delphi
: Level 5
K 다소 위험
실제 이송이 진행되는 함수이므로, 사전에 반드시 안전을 확인합니다.

SYNOPSIS

r VT_I4 cemOverrideMoveTo ([in] VT_I4 Axis, [in] VT_R8 NewPosition,

[in] VT_I4 IsHardApply, [out] VT_PI4 AppliedState)

DESCRIPTION

cemSxMoveToStart 이송 함수를 통하여 수행되는 절대 좌표 In-position 모션에 대하여 목표 절대 좌표를 오버라이딩 하는 함수입니다.

PARAMETER

▶ *Axis* : 축 번호. 통합 축으로 관리되는 축 번호를 의미하며, 상수 값으로 0 (Zero Based) 이상, (최대 통합 축 개수 - 1) 이하의 값을 축 번호로 설정할 수 있습니다.

▶ *NewPosition* : 새로운 목표 절대 좌표를 지정합니다.

▶ *IsHardApply* : 위치 오버라이드 불가 시점에서도 오버라이드 설정을 강제로 적용합니다.

▶ *AppliedState* : cemOverrideMoveTo 함수의 적용 성공/실패 여부를 반환 합니다.

Value	Meaning
0 (CE_FALSE)	모션 에러가 발생하였거나 이미 이송이 완료되어 위치 오버라이드가 적용되지 않음.
1 (CE_TRUE)	위치 오버라이드가 적용됨.

RETURN VALUE

Value	Meaning
음수	수행 실패. 자세한 내용은 '에러처리' 편을 참고합니다.
0 (ceERR_NONE)	수행 성공.

SEE ALSO

cemSxMoveToStart

REFERENCE

□ 위치 오버라이드를 수행하려는 시점에 이미 이송이 완료되어 버린 경우에는 위치 오버라이드는 무시되고 AppliedState 매개변수는 '0' 을 반환합니다. 따라서 사용자는 AppliedState 매개변수의 반환값이 '0'인 경우에는 이미 이송이 완료되어 오버라이드가 적용되지 않은 것으로 인지하여야 하며, 그럼에도 불구하고 목표 좌표를 수정해야 하는 경우에는 cemSxMove 또는 cemSxMoveTo 함수를 추가적으로 수행해야 합니다. 이러한 경우에는 오버라이드라는 개념 보다는 추가 이송의 개념을 의미합니다.

EXAMPLE

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

8.2 Master/Slave 동기 제어

Master/Slave 동기 제어는 두 개의 모터 축이 하나의 모터처럼 완전한 동기 제어가 필요한 시스템에 사용되는 기능입니다.

㈜커미조아 모션컨트롤러에서 제공하는 Master/Slave 동기 제어 기능은 Manual Pulsar 모드를 응용한 것입니다. 이 기능을 사용하면 두 축을 하나의 축처럼 제어하는 것이 가능합니다.

Master/Slave 동기 제어 모드 상태에서 Slave 축은 자동적으로 Master 축과 동기되어서 동작하므로 사용자는 Master 축만 제어하면 됩니다. 이것은 Master/Slave 축을 하나의 가상축으로 하여 제어의 편리성을 제공합니다.

8.2.1 함수 요약

Master/Slave 동기 제어에 관련된 함수들은 다음과 같습니다.

Summary of Functions
r VT_I4 cemMsRegisterSlave ([in] VT_I4 Axis, [in] VT_R8 MaxSpeed, [in] VT_I4 IsInverse) 대상 모션 축에 대해서, Master/Slave 동기 구동의 Slave 축으로 등록합니다. 해당 축의 이전 축은 묵시적으로 Master 축이 되며, Slave 축은 Master 축과 동기 되어 구동됩니다.
r VT_I4 cemMsUnregisterSlave ([in] VT_I4 Axis) 대상 모션 축에 대해서, Master/Slave 동기 구동의 Slave 축으로서의 등록을 해제합니다.
r VT_I4 cemMsCheckSlaveState ([in] VT_I4 SlaveAxis, [out] VT_PI4 SlaveState) 대상 모션 축에 대해서, Master/Slave 동기 구동의 Slave 축으로 정상 등록되어 있는지에 대한 여부를 반환합니다.
r VT_I4 cemMsMasterAxis_Get ([in] VT_I4 SlaveAxis, [out] VT_PI4 MasterAxis) 대상 모션 축에 대해서, Master/Slave 동기 구동의 Master 축을 조회하여, 해당 Slave 축에 대한 Master 축의 번호를 반환합니다.

8.2.2 함수 설명

NAME

cemMsRegisterSlave / cemMsUnregisterSlave

- 동기 제어 대상 축으로 설정 및 설정 해제

INFORMATION	
1	Master / Slave
!	VC++ (6, 7, 8)/VB
	BCB/Delphi
:	Level 5
J	위험 요소 없음

SYNOPSIS

r VT_I4 cemMsRegisterSlave ([in] VT_I4 Axis, [in] VT_R8 MaxSpeed, [in] VT_I4 IsInverse)

r VT_I4 cemMsUnregisterSlave ([in] VT_I4 Axis)

DESCRIPTION

cemMsRegisterSlave 함수는 지정한 축을 Slave 축으로 등록합니다. 이때 마스터 축은 자동으로 결정되며, 축 번호상 앞의 축이 자동적으로 Master 축으로 됩니다.

cemMsUnregisterSlave 함수는 Slave 축으로 등록된 상태를 해제 합니다.

PARAMETER

▶ ***Axis*** : Slave 축으로 설정 혹은 설정 해제할 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.

▶ ***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 구동 방향과 Master 축의 구동 방향을 반대로 할 것인지를 결정합니다.

Value	Meaning
0 (CE_FALSE)	Master 축과 Slave 축의 구동 방향을 동일하게 함.
1 (CE_TRUE)	Master 축과 Slave 축의 구동 방향을 반대로 함.

RETURN VALUE

Value	Meaning
음수	수행 실패. 자세한 내용은 ‘에러처리’ 편을 참고합니다.
0 (ceERR_NONE)	수행 성공.

REFERENCE

□ 이 함수가 수행되면 지정한 축은 Manual Pulsar 모드로 전환되며, 지정한 축의 PA/PB 입력 핀에 Master 축의 Command 펄스 신호가 자동으로 입력되고, 따라서 두 축이 동기 제어됩니다.

□ Master 축의 Command 출력 모드는 반드시 “CW & CCW 모드 (출력모드 4 ~ 5 번)”로 설정되어 있어야 합니다. Master 축의 Command 출력 모드가 “Pulse & Direction 모드 (출력모드 0 ~ 3 번)”로 설정된 경우에는 Slave 축이 한쪽 방향으로만 구동되게 됩니다. Command 출력 모드의 설정에 관해서는 cemCfgOutMode_Set 함수 설명 편을 참조하시기 바랍니다.

□ Master 축의 Command 출력 모드가 “Pulse & Direction 모드 (출력모드 0~3)” 로 설정된 경우에는 Master / Slave 기능을 양방향에서 원활하게 사용할 수 없습니다.

□ 이 함수가 성공적으로 수행되면 모션 동작 상태 반환(cemStReadMotionState) 함수의 반환 값이 8 (cemMST_WAIT_PLSR)이 됩니다. 이를 확인하면 지정한 축이 Slave 모드로 전환되었는지를 확인할 수 있습니다.

□ Slave 축은 INP 입력이 자동으로 Disable 됩니다. 따라서 이 함수를 호출한 이후에 INP 입력을 Enable 시키는 환경 설정 명령을 수행해서는 안됩니다.

□ Slave 모드가 해제되면 해당 축의 INP 입력모드는 cemMsRegisterSlave 함수가 호출되기 이전의 설정으로 복원됩니다.

EXAMPLE

C/C++

```
#include “ceSDK.h”
#include “ceSDKDef.h”

/*****************************************************************************
* Master/Slave 동기 구동을 하기 위해 슬레이브 축을 등록 합니다.
*****************************************************************************/
void OnSetMsRegisterSalve ()
{
    long nRetVal;

    // 1 번 축을 슬레이브 축으로 등록합니다. 자동으로 마스터 축은 0(cemX1)번 축이 됩니다.
    nRetVal = cemMsRegisterSlave (    cemY1,          // 슬레이브 축 설정
                                      655350,         // 슬레이브 축 최대 구동 속도
                                                      // 마스터 축의 최대 속도와 같거나 커야 합니다.
                                      CE_FALSE        // 마스터 축과의 구동 방향 설정
                                      );
    if ( nRetVal == ceERR_NONE )
    {
            // 슬레이브 축의 등록 상태를 확인 합니다.
            long nSlaveState;
            cemMsCheckSlaveState ( cemY1, &nSlaveState );

            if ( nSlaveState != CE_TRUE )
            {
                    OutputDebugString ( “Slave axis registered failed” );
            }

            // 슬레이브 축의 마스터 축을 확인합니다.
            long nMasterAxis;
            cemMsMasterAxis_Get ( cemY1, &nMasterAxis );

            if ( nMasterAxis != cemX1 )
            {
                    OutputDebugString ( “Slave axis registered failed” );
            }
    }
}


/*****************************************************************************
* 마스터 축에 대해 단축 이송 작업을 수행합니다. 슬레이브 축이 등록
* 되어 있으므로 마스터 축 제어 시 슬레이브 축이 함께 제어됩니다.
*****************************************************************************/
void OnMove ()
{
    // 마스터 축 속도 환경을 설정합니다.
    cemCfgSpeedPattern_Set ( cemX1, cemSMODE_S, 5000, 30000, 30000 );

    // 마스터 축 단축 이송 수행 시 슬레이브 축이 함께 이송합니다.
    cemSxMove ( cemX1, 15000, CE_FALSE );
}
```

Visual Basic

```
' ================================================
' Master/Slave 동기 구동을 하기 위해 슬레이브 축을 등록 합니다.
' ================================================
Private Sub OnSetMsRegisterSalve ()

    Dim nSlaveState As Long          ' 슬레이브 축 등록 상태 정보.
    Dim nMasterAxis As Long          ' 마스터 축 정보.

    ' 1 번 축을 슬레이브 축으로 등록합니다. 자동으로 마스터 축은 0(cemX1)번 축이 됩니다.
    If cemMsRegisterSlave ( cemY1, 655350, CE_FALSE ) = ceERR_NONE Then

        ' 슬레이브 축의 등록 상태를 확인 합니다.
        Call cemMsCheckSlaveState ( cemY1, nSlaveState )

        If nSlaveState <> CE_TRUE Then
            MsgBox ( "Slave axis registered failed" )
        End If

        ' 슬레이브 축의 마스터 축을 확인합니다.
        Call cemMsMasterAxis_Get ( cemY1, nMasterAxis )

        If nMasterAxis <> cemX1 Then
            MsgBox ( "Slave axis registered failed" )
        End If
    End If

End Sub


' ================================================
' 마스터 축에 대해 단축 이송 작업을 수행합니다. 슬레이브 축이 등록
' 되어 있으므로 마스터 축 제어 시 슬레이브 축이 함께 제어됩니다.
' ================================================
Private Sub OnMove ()

    ' 마스터 축 속도 환경을 설정합니다.
    Call cemCfgSpeedPattern_Set ( cemX1, cemSMODE_S, 5000, 30000, 30000 )

    ' 마스터 축 단축 이송 수행 시 슬레이브 축이 함께 이송합니다.
    Call cemSxMove ( cemX1, 15000, CE_FALSE )

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(cemX1)번 축이 됩니다.
    if cemMsRegisterSlave ( cemY1, 655350, CE_FALSE ) = ceERR_NONE then
    begin
            // 슬레이브 축의 등록 상태를 확인 합니다.
            cemMsCheckSlaveState ( cemY1, @nSlaveState );

            if nSlaveState <> CE_TRUE then
            begin
                    ShowMessage ( 'Slave axis registered failed' );
            end;

            // 슬레이브 축의 마스터 축을 확인합니다.
            cemMsMasterAxis_Get ( cemY1, @nMasterAxis );

            if nMasterAxis <> cemX1 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.
    cemCfgSpeedPattern_Set ( cemX1, cemSMODE_S, 5000, 30000, 30000 );

    // 마스터 축 단축 이송 수행 시 슬레이브 축이 함께 이송합니다.
    cemSxMove ( cemX1, 15000, CE_FALSE );

end;
```

NAME

cemMsCheckSlaveState

- 동기 제어 대상(Slave) 축 등록 상태 확인

INFORMATION	
1	Master / Slave
!	VC++ (6, 7, 8)/VB
	BCB/Delphi
:	Level 5
J	위험 요소 없음

SYNOPSIS

r VT_I4 cemMsCheckSlaveState ([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 확인할 대상 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.

▶ ***SlaveState*** : 해당 축의 Slave 축 등록 상태를 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 시켜 주어야 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.
0 (ceERR_NONE)	수행 성공.

EXAMPLE

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

NAME

cemMsMasterAxis_Get

- 동기 제어 대상(Slave)축의 Master 축 번호 반환

INFORMATION	
1	Master / Slave
!	VC++ (6, 7, 8)/VB
	BCB/Delphi
:	Level 5
J	위험 요소 없음

SYNOPSIS

r VT_I4 cemMsMasterAxis_Get ([in] VT_I4 SlaveAxis, [out] VT_PI4 MasterAxis)

DESCRIPTION

대상 모션 축에 대해서, Master / Slave 동기 구동의 Master 축을 조회하여, 해당 Slave 축의 Master 축으로 동작하는 축의 번호를 반환합니다.

PARAMETER

▶ ***SlaveAxis*** : Slave 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.

▶ ***MasterAxis*** : 이 매개변수를 통하여 Slave 축의 Master 축 번호를 반환합니다.

RETURN VALUE

Value	Meaning
음수	수행 실패. 자세한 내용은 '에러처리' 편을 참고합니다.
0 (ceERR_NONE)	수행 성공.

REFERENCE

□ Master/Slave 기능을 사용할 때 Master 축 번호는 Slave 축에 따라서 자동으로 결정되면, 축 번호상 앞의 축이 자동적으로 Master 축으로 설정됩니다.

EXAMPLE

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

Chapter

9

Input signals related to motion control by external signal

모션 제어에서의 외부 신호 입력은 다양한 모션 제어로 응용될 수 있습니다. MPG 와 같은 펄스 출력 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 모션 제어 함수들을 소개합니다. Manual Pulsar 모드는 PA 입력 신호와 PB 입력 신호에 동기하여 모터를 제어하는 모드입니다. PA/PB 를 통하여 입력되는 신호는 지연시간 없이 Command 로 전달됩니다. 외부 신호에 의해 동기되는 모션제어는 다양한 외부 신호 구성을 통해 모션 제어의 동작의 속성을 정의할 수 있습니다.

9 외부 신호 동기 제어 편

9.1 Manual Pulsar (PA/PB) 모드 모션 제어

이 단원에서는 Manual Pulsar 모드 모션에 관련된 함수들을 소개합니다. Manual Pulsar 모드는 PA 입력 신호와 PB 입력 신호에 동기 하여 모터를 제어하는 모드입니다. PA/PB 를 통하여 입력되는 신호는 지연 시간 없이 Command 로 전달됩니다.

Manual Pulsar 모드를 사용하면 로터리 엔코더(Rotary Encoder)와 같은 장치로부터 PA/PB 단자를 통하여 Plus 펄스와 Minus 펄스(CW/CCW) 또는 90° 위상 차를 갖는 A/B Phase 펄스를 입력 받아 수동으로 모션을 제어할 수 있습니다. 또는 다른 모터나 외부 장치와 동기 제어를 하는 데에도 유용하게 사용될 수 있습니다. 특히 Manual Pulsar 모드는 ㈜커미조아 모션 컨트롤러의 “Master/ Slave 모션제어” 기능에 응용됩니다.

PA/PB 단자에 입력되는 신호의 형태는 Pulsar 입력 모드 설정에 따라 달라지며 이는 Pulsar 입력 모드 환경 설정 함수인 cemPlsrInMode_Set 함수에 의해 설정됩니다.

Manual Pulsar 모드 모션은 Velocity Motion 은 물론 상대 좌표/절대 좌표 In-Position 모션에도 적용 가능합니다. 그러나 Coordinated Motion 에는 적용할 수 없습니다.

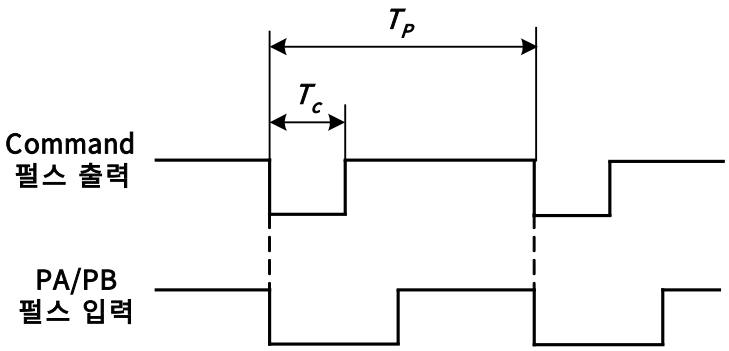
□ Manual Pulsar 모드 모션에서의 속도는 입력 펄스의 주파수에 의해 결정됩니다. 따라서 설정되어 있는 속도 모드나 가속도 및 감속도는 무시되며, Manual Pulsar 입력 신호의 최대 주파수는 cemCfgSpeedPattern_Set 함수에서 설정하는 작업속도에 의해서 결정됩니다.

Manual Pulsar 모션을 수행하기 전에 cemCfgSpeedPattern_Set 함수를 이용하여 작업 속도, 즉 입력 신호의 최대속도를 적절한 값으로 설정하여야 합니다. 설정한 최대 속도를 초과하는 PA/PB 신호가 입력되면 오동작하게 됩니다. 주의할 것은 작업 속도는 논리 속도로 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/PB 로 입력될 수 있는 펄스의 최대 주파수(PPS) **Vwork** : 작업 속도 **fu** : Unit speed 값

□ cemCfgSpeedPattern_Set 함수에서 설정하는 작업 속도는 PA/PB 단자로 입력될 수 있는 펄스의 최대 주파수를 제한하는 동시에 해당 축의 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 펄스 폭은 (1000000/(500000*2)) = 1 μsec 이상이어야 합니다. (최대 주파수에서 실제 입력 펄스의 시간은 ON/OFF 를 포함한 것이므로, Command 펄스의 펄스 폭에 대한 값은 실제 최대 주파수의 2 배가 되어야 합니다) 이러한 경우에, 만일 cemCfgSpeedPattern_Set 함수에서 설정하는 작업속도를 500 KPPS 보다 큰 값으로 설정하면 펄스 폭이 짧아서 Command 펄스를 서보 드라이버가 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 신호 연결 방법은 별도로 제공되는 HW 매뉴얼의 “MPG 입력 신호 (PA, PB, MPGG)” 단원을 참조하시기 바랍니다.

□ Manual Pulsar 구동 명령이 성공적으로 수행되면 cemStReadMotionState 함수의 반환 값이 8 (cemMST_WAIT_PLSR)이 됩니다. 이를 확인하면 Manual Pulsar 구동 명령 시작 여부를 확인할 수 있습니다

9.1.1 함수 요약

Manual Pulsar 모드 모션에 관련된 함수들은 다음과 같습니다.

Summary of Functions
r VT_I4 cemPlsrInMode_Set ([in] VT_I4 Channel, [in] VT_I4 InputMode, [in] VT_I4 IsInverse) 대상 모션 축에 대해서, Manual Pulsar 입력 신호에 대한 환경을 설정합니다.
r VT_I4 cemPlsrInMode_Get ([in] VT_I4 Channel, [out] VT_PI4 InputMode, [out] VT_PI4 IsInverse) 대상 모션 축에 대해서, Manual Pulsar 입력 신호에 대한 환경 설정을 반환합니다.
r VT_I4 cemPlsrGain_Set ([in] VT_I4 Channel, [in] VT_I4 GainFactor, [in] VT_I4 DivFactor) 대상 모션 축에 대해서, Manual Pulsar 의 PA/PB 입력 펄스 대비 지령 펄스(Command Pulse)의 출력 펄스 수에 대한 비율을 설정합니다.
r VT_I4 cemPlsrGain_Get ([in] VT_I4 Channel, [out] VT_PI4 GainFactor, [out] VT_PI4 DivFactor) 대상 모션 축에 대해서, Manual Pulsar 의 PA/PB 입력 펄스 대비 지령 펄스의 출력 펄스 수에 대한 비율을 반환합니다.
r VT_I4 cemPlsrHomeMoveStart ([in] VT_I4 Channel, [in] VT_I4 HomeType) 대상 모션 축에 대해서, Manual Pulsar 입력 신호에 의한 원점 복귀 이송을 수행합니다. 이 함수는 모션 시작 후 바로 반환됩니다.
r VT_I4 cemPlsrMove ([in] VT_I4 Channel, [in] VT_R8 Distance, [in] VT_I4 IsBlocking) 대상 모션 축에 대해서, Manual Pulsar 입력 신호에 맞추어 지정한 논리적 거리를 통해 상대 좌표 이송을 수행합니다. 이 함수는 모션이 완료되기 전까지 반환되지 않습니다.
r VT_I4 cemPlsrMoveStart ([in] VT_I4 Channel, [in] VT_R8 Distance) 대상 모션 축에 대해서, Manual Pulsar 입력 신호에 맞추어 지정한 논리적 거리를 통해 상대 좌표 이송을 수행합니다. 이 함수는 모션 시작 후 바로 반환됩니다.
r VT_I4 cemPlsrMoveTo ([in] VT_I4 Channel, [in] VT_R8 Position, [in] VT_I4 IsBlocking) 대상 모션 축에 대해서, Manual Pulsar 입력 신호에 맞추어 지정한 논리적 거리를 통해 절대 좌표 이송을 수행합니다. 이 함수는 모션이 완료되기 전까지 반환되지 않습니다.
r VT_I4 cemPlsrMoveToStart ([in] VT_I4 Channel, [in] VT_R8 Position) 대상 모션 축에 대해서, Manual Pulsar 입력 신호에 맞추어 지정한 논리적 거리를 통해 절대 좌표 이송을 수행합니다. 이 함수는 모션 시작 후 바로 반환됩니다.
r VT_I4 cemPlsrVMoveStart ([in] VT_I4 Channel) 대상 모션 축에 대해서, Manual Pulsar 입력 신호에 맞추어 연속 속도 이송을 시작합니다. 이 함수는 모션 시작 후 바로 반환됩니다.

9.1.2 함수 설명

NAME

cemPlsrInMode_Set / cemPlsrInMode_Get

- Pulsar 입력 신호 환경설정 및 설정 값 반환

INFORMATION
1 Manual Pulsar
! VC++ (6, 7, 8)/VB
BCB/Delphi
: Level 5
K 다소 주의
Manual Pulsar 기능을 사용하는 모션 축에서는 내부적인 하드웨어 구조상으로 Master/Slave 기능을 사용할 수 없습니다.

SYNOPSIS

r VT_I4 cemPlsrInMode_Set ([in] VT_I4 Channel, [in] VT_I4 InputMode, [in] VT_I4 IsInverse)

r VT_I4 cemPlsrInMode_Get ([in] VT_I4 Channel, [out] VT_PI4 InputMode,

[out] VT_PI4 IsInverse)

DESCRIPTION

cemPlsrInMode_Set 함수는 Pulsar 입력 신호에 대한 환경을 설정하며, cemPlsrInMode_Get 함수는 Pulsar 입력 신호의 환경 설정 상태를 반환합니다.

설정되는 Pulsar 입력 신호 설정은 크게 A 상과 B 상을 검출하는 모드와 CW/CCW 신호 검출 모드가 있습니다. 신호의 성격에 맞게 입력 신호의 환경 설정을 해주시기 바랍니다.

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.

▶ *InputMode* : PA 와 PB 입력 단자를 통하여 입력되는 Pulsar 입력 신호의 입력 모드를 설정 혹은 설정 상태를 반환합니다. 설정 가능한 입력 모드는 다음과 같습니다.

Value	Meaning
0 (cemIMODE_AB1X)	1X A/B (1 채배 Phase type 입력 모드)
1 (cemIMODE_AB2X)	2X A/B (2 채배 Phase type 입력 모드)
2 (cemIMODE_AB4X)	4X A/B (4 채배 Phase type 입력 모드)
3 (cemIMODE_CWCCW)	CW/CCW (PA – Plus direction move, PB – Minus direction move)

▶ *IsInverse* : Pulsar 입력 신호에 의해 결정되는 방향(Direction)을 모션에 반대로 적용할 지 여부를 설정 혹은 설정 상태를 반환 합니다. 의미는 다음과 같습니다.

Value	Meaning
0 (CE_FALSE)	Pulsar 입력 신호가 나타내는 방향과 모션의 방향 일치
1 (CE_TRUE)	Pulsar 입력 신호가 나타내는 방향과 모션의 방향을 반대로 적용

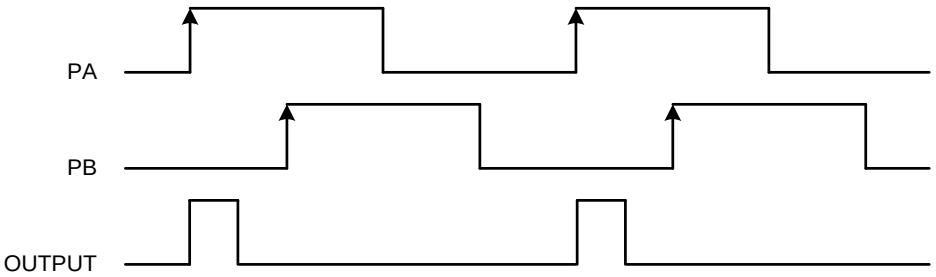
RETURN VALUE

Value	Meaning
음수	수행 실패. 자세한 내용은 '에러처리' 편을 참고합니다.
0 (ceERR_NONE)	수행 성공.

REFERENCE

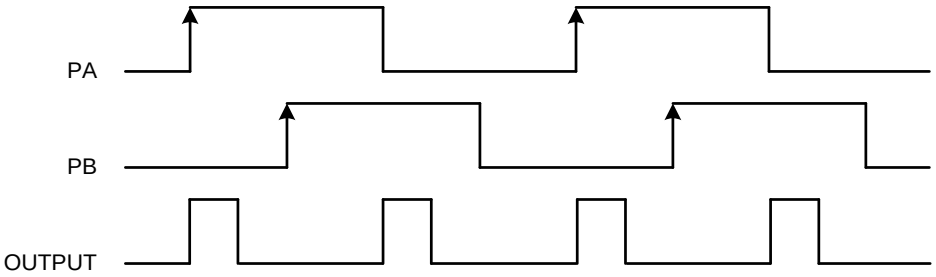
□ InputMode 설정에 따라서 PA/PB 입력 신호와 Command 출력 펄스의 관계는 다음과 같습니다.

• InputMode = cemIMODE_AB1X 일 때



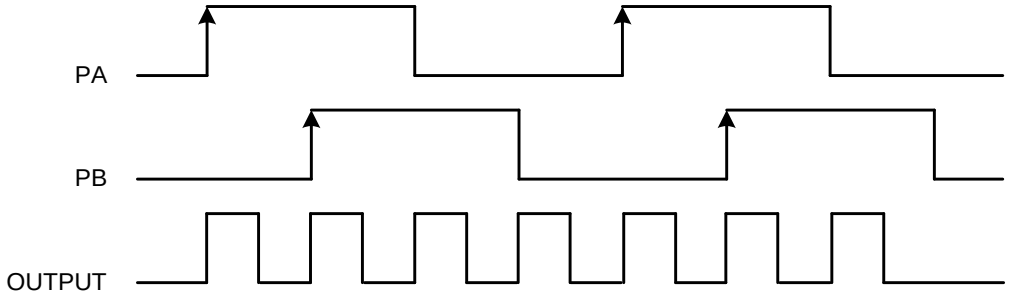
+/- 방향은 PA 신호와 PB 신호의 위상에 따라서 결정됩니다.

• InputMode = cemIMODE_AB2X 일 때



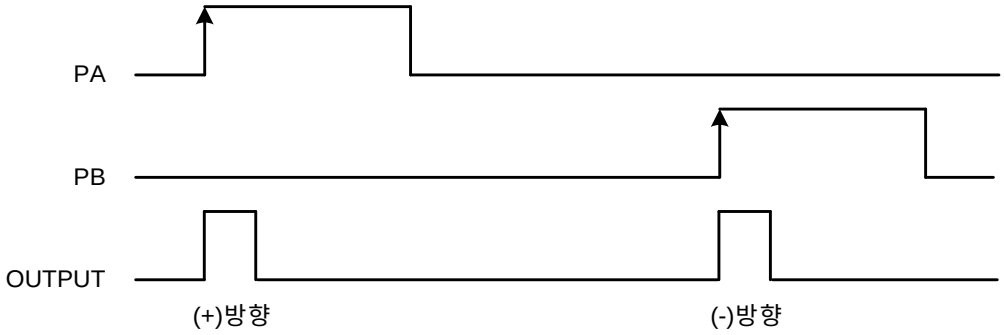
+/- 방향은 PA 신호와 PB 신호의 위상에 따라서 결정됩니다.

• InputMode = cemIMODE_AB4X 일 때



+/- 방향은 PA 신호와 PB 신호의 위상에 따라서 결정됩니다.

• InputMode = cemIMODE_CWCCW 일 때



EXAMPLE

C/C++

```
#include “ceSDK.h”
#include “ceSDKDef.h”

void OnSetPlsrInMode ()
{
    long nInputMode, nIsInverse;          // Pulsar 입력 모드 정보.

    // 설정된 Manual Pulsar 설정 값을 확인한 후, 입력 모드를 ‘CW/CCW’ 모드로 설정합니다.
    if ( cemPlsrInMode_Get ( cemX1, &nInputMode, &nIsInverse ) == ceERR_NONE )
    {
            if ( nInputMode != cemIMODE_CWCCW )
            {
                    cemPlsrInMode_Set ( cemX1, cemIMODE_CWCCW, CE_FALSE );
            }
    }
}
```

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 cemPlsrInMode_Get ( cemX1, nInputMode, nIsInverse ) = ceERR_NONE Then
            If nInputMode <> cemIMODE_CWCCW Then
                    Call cemPlsrInMode_Set ( cemX1, cemIMODE_CWCCW, CE_FALSE )
            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 cemPlsrInMode_Get ( cemX1, @nInputMode, @nIsInverse ) = ceERR_NONE then
    begin
            if nInputMode <> cemIMODE_CWCCW then
            begin
                    cemPlsrInMode_Set ( cemX1, cemIMODE_CWCCW, CE_FALSE );
            end;
    end;
end;
```

NAME

cemPlsrGain_Set / cemPlsrGain_Get

- PA/PB 입력 대비, 출력 펄스 환경 설정 및 설정 값 반환

INFORMATION	
1	Manual Pulsar
!	VC++ (6, 7, 8)/VB
	BCB/Delphi
:	Level 5
K	다소 주의

함수의 내용이 약간의 난이도를 가지고 있습니다. 주의 깊게 읽어 주시기 바랍니다.

SYNOPSIS

r VT_I4 cemPlsrGain_Set ([in] VT_I4 Channel, [in] VT_I4 GainFactor, [in] VT_I4 DivFactor)

r VT_I4 cemPlsrGain_Get ([in] VT_I4 Channel, [out] VT_PI4 GainFactor,

[out] VT_PI4 DivFactor)

DESCRIPTION

cemPlsrGain_Set 함수는 PA/PB 입력 펄스 대비 Command 출력 펄스 수의 비를 사용자가 임의로 조절할 수 있도록 하는 함수이며, cemPlsrGain_Get 함수는 현재 설정된 Pulsar 출력 펄스 수 Gain 을 반환하는 함수입니다.

Manual Pulsar 출력은 PA/PB 로 입력되는 신호가 다음과 같이 3 단계의 회로를 거쳐서 출력되게 됩니다.

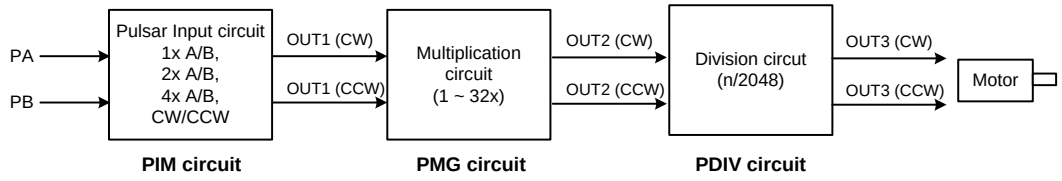


그림 9-1 PIM Circuit

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 cemPlsrInMode_Set 함수의 InputMode 매개 변수에 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 회로를 거쳐 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 cemPlsrGain_Set 함수의 DivFactor 설정 값을 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

▶ *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.

▶ *GainFactor* : PMG 회로에 설정되는 사용자 정수를 설정 혹은 반환합니다. GainFactor 는 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.

▶ *DivFactor* : PDIV 회로에 설정되는 사용자 정수를 설정 혹은 반환합니다. DivFactor 는 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.

RETURN VALUE

Value	Meaning
음수	수행 실패. 자세한 내용은 '에러처리' 편을 참고합니다
0 (ceERR_NONE)	수행 성공

REFERENCE

□ 다음의 표는 PA/PB 입력 펄스와 Command 출력 펄스 수의 관계를 규정하는 3 가지 항목 설정 값에 따라서 입력 펄스와 출력 펄스 수의 비율이 어떻게 되는지를 몇 가지를 예시한 것입니다. 아래 표에서 Pulsar Input Mode 는 cemPlsrInMode_Set 함수의 InputMode 매개변수를 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

EXAMPLE

C/C++

```
#include “ceSDK.h”
#include “ceSDKDef.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 ( cemPlsrInMode_Set ( cemX1, cemIMODE_CWCCW, CE_FALSE ) == ceERR_NONE )
    {
            cemPlsrGain_Set ( cemX1,        // 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 ( cemPlsrInMode_Set ( cemX1, cemIMODE_CWCCW, CE_FALSE ) == ceERR_NONE )
    {
            cemPlsrGain_Set ( cemX1, 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 cemPlsrInMode_Set ( cemX1, cemIMODE_AB2X, CE_FALSE ) = ceERR_NONE Then
            Call cemPlsrGain_Set ( cemX1, 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 cemPlsrInMode_Set ( cemX1, cemIMODE_AB2X, CE_FALSE ) = ceERR_NONE Then
            Call cemPlsrGain_Set ( cemX1, 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 cemPlsrInMode_Set ( cemX1, cemIMODE_AB1X, CE_FALSE ) = ceERR_NONE then
    begin
            cemPlsrGain_Set ( cemX1, 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 cemPlsrInMode_Set ( cemX1, cemIMODE_AB2X, CE_FALSE ) = ceERR_NONE then
    begin
            cemPlsrGain_Set ( cemX1, 5, 2048 );
    end;

end;
```

NAME

cemPlsrHomeMoveStart

- Pulsar Input 에 의한 원점 복귀 이송

INFORMATION
1 Manual Pulsar
! VC++ (6, 7, 8)/VB
BCB/Delphi
: Level 5
K 이송 함수
실제 이송이 진행되는 함수이므로, 사전에 반드시 안전을 확인합니다.

SYNOPSIS

r VT_I4 cemPlsrHomeMoveStart ([in] VT_I4 Channel, [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 cemSxStopEmg 함수가 호출되면 모션을 종료합니다.

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.

▶ *HomeType* : Pulsar Input 에 의해 원점 복귀를 수행하는 모드를 설정합니다. 이 값은 다음 중 하나의 값이어야 합니다.

Value	Meaning
0	Command 카운터가 0 이 될 때 원점복귀를 종료합니다.
1	ORG 신호가 ON 이 되면 원점복귀를 종료합니다.

RETURN VALUE

Value	Meaning
음수	수행 실패. 자세한 내용은 '에러처리' 편을 참고합니다.
0 (ceERR_NONE)	수행 성공.

REFERENCE

□ Manual Pulsar 모드 모션제어를 사용하고자 할 때에도 Manual Pulsar 모드 원점복귀 대신 일반적인 원점복귀를 수행할 수 있습니다.

EXAMPLE

```
//* cemPlsrMove / cemPlsrMoveStart 예제를 참고하여 주시기 바랍니다.
```

NAME

cemPlsrMove / cemPlsrMoveStart

- Pulsar Input 에 의한 상대좌표 이송

INFORMATION
1 Manual Pulsar
! VC++ (6, 7, 8)/VB
BCB/Delphi
: Level 5
K 이송 함수
실제 이송이 진행되는 함수이므로, 사전에 반드시 안전을 확인합니다.

SYNOPSIS

r VT_I4 cemPlsrMove ([in] VT_I4 Channel, [in] VT_R8 Distance, [in] VT_I4 IsBlocking)

r VT_I4 cemPlsrMoveStart ([in] VT_I4 Channel, [in] VT_R8 Distance)

DESCRIPTION

cemPlsrMove/cemPlsrMoveStart 함수는 Pulsar 신호 입력에 맞추어 지정한 거리(상대 좌표) 만큼 이송을 수행한 후에 Manual Pulsar 모드를 해제합니다.

cemPlsrMove 함수는 모션이 완료되기 전까지 반환되지 않으며, cemPlsrMoveStart 함수는 모션을 시작시킨 후에 바로 반환됩니다.

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.

▶ *Distance* : 이동할 거리를 지정합니다. 지정한 거리만큼 이동한 후 Manual Pulsar 모드는 자동으로 해제됩니다. 거리의 단위는 “Unit distance” 에 의해 정의되는 논리적 거리입니다.

▶ *IsBlocking* : cemPlsrMove 함수의 인자이며, 완료될 때까지 기다리는 동안 윈도우 메시지를 블록(Blocking)할 것인지를 결정합니다. 단, 쓰레드내에서 실행할 때는 이 값을 1(CE_TRUE)로 설정해 주어야 합니다.

Value	Meaning
0 (CE_FALSE)	윈도우 메시지를 블록(Blocking) 하지 않습니다. 따라서 해당 모션이 진행되는 동안 윈도우 이벤트를 처리합니다.
1 (CE_TRUE)	윈도우 메시지를 블록(Blocking) 합니다. 따라서 해당 모션이 완료되는 동안에는 윈도우 이벤트가 처리되지 않습니다.

RETURN VALUE

Value	Meaning
음수	수행 실패. 자세한 내용은 '에러처리' 편을 참고합니다.
0 (ceERR_NONE)	수행 성공.

SEE ALSO

cemPlsrMoveTo, cemPlsrMoveToStart

REFERENCE

□ Manual Pulsar 모드 모션을 중지하기 위해서는 cemSxStopEmg 함수를 사용하십시오. 이 함수가 호출되면 Manual Pulsar 모드는 해제됩니다.

윈도우 이벤트라는 것은 무엇입니까?

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

EXAMPLE

C/C++

```
#include “ceSDK.h”
#include “ceSDKDef.h”

void OnSetPlsrConfig ()
{
    // Manual Pulsar 입력 모드 설정 및 입력 펄스와 출력 펄스 비를 1:2 로 설정합니다.
    cemPlsrInMode_Set ( cemX1, cemIMODE_CWCCW, CE_FALSE );
    cemPlsrGain_Set ( cemX1, 2, 2048 );

    /* PA/PB 신호 입력 회로에 노이즈 필터 기능을 적용합니다.
     노이즈 필터 기능이 활성화 되면 3.25MHz 이상의 주파수를 가지는 펄스는 노이즈로 처리됩니다.*/
    cemCfgFilterAB_Set ( cemX1, cemAB_PULSAR, CE_TRUE );
}


void OnPulsarMove ()
{
    /* 속도 환경을 설정합니다. 작업 속도는 출력 펄스의 폭을 결정하며,
    일반적으로 사용하는 서보 드라이버의 최대 속도로 설정하시면 됩니다. */
    cemCfgSpeedPattern_Set ( cemX1,          // 대상 축을 설정합니다.
                             cemSMODE_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
    cemStReadMotionState 함수의 모션 상태 반환 값이 8 ( cemMST_WAIT_PLSR ) 이 됩니다. */

    // 지정한 거리만큼 상대좌표 이송을 수행합니다. 모션이 완료되기 전까지 반환되지 않습니다.
    cemPlsrMove ( cemX1, 10000, CE_FALSE );

    // cemPlsrMoveStart()함수는 모션을 시작시킨 후에 바로 반환됩니다.
}
```

Visual Basic

```
Private Sub OnSetPlsrConfig ()

    ‘ Manual Pulsar 입력 모드 설정 및 입력 펄스와 출력 펄스 비를 1:2 로 설정합니다.
    Call cemPlsrInMode_Set ( cemX1, cemIMODE_CWCCW, CE_FALSE )
    Call cemPlsrGain_Set ( cemX1, 2, 2048 )
```

```
    ‘ PA/PB 신호 입력 회로에 노이즈 필터 기능을 적용합니다.
    Call cemCfgFilterAB_Set ( cemX1, cemAB_PULSAR, CE_TRUE )

End Sub


Private Sub OnPulsarMove ()

    ‘ 속도 환경을 설정합니다. 작업 속도는 출력 펄스의 폭을 결정하며,
    ‘ 일반적으로 사용하는 서보 드라이버의 최대 속도로 설정하시면 됩니다.
    Call cemCfgSpeedPattern_Set ( cemX1, cemSMODE_S, 655350, 10000, 10000 )


    ‘ Manual Pulsar 입력에 맞춰서 지정한 거리만큼 이송을 수행한 후
    ‘ Manual Pulsar 모드를 해제 합니다. Pulsar 이송 함수가 수행되면
    ‘ cemStReadMotionState 함수의 모션 상태 반환 값이 8 ( cemMST_WAIT_PLSR ) 이 됩니다.

    ‘ 지정한 거리만큼 상대좌표 이송을 수행합니다. 모션이 완료되기 전까지 반환되지 않습니다.
    Call cemPlsrMove ( cemX1, 10000, CE_FALSE )

    ‘ cemPlsrMoveStart()함수는 모션을 시작시킨 후에 바로 반환됩니다.

End Sub
```

Delphi

```
procedure OnSetPlsrConfig ();
begin
    // Manual Pulsar 입력 모드 설정 및 입력 펄스와 출력 펄스 비를 1:2 로 설정합니다.
    cemPlsrInMode_Set ( cemX1, cemIMODE_CWCCW, CE_FALSE );
    cemPlsrGain_Set ( cemX1, 2, 2048 );

    // PA/PB 신호 입력 회로에 노이즈 필터 기능을 적용합니다.
    cemCfgFilterAB_Set ( cemX1, cemAB_PULSAR, CE_TRUE );

end;

procedure OnPulsarMove ();
begin
    { 속도 환경을 설정합니다. 작업 속도는 출력 펄스의 폭을 결정하며,
    일반적으로 사용하는 서보 드라이버의 최대 속도로 설정하시면 됩니다. }
    cemCfgSpeedPattern_Set ( cemX1, cemSMODE_S, 655350, 10000, 10000 );

    { Manual Pulsar 입력에 맞춰서 지정한 거리만큼 이송을 수행한 후
    Manual Pulsar 모드를 해제 합니다. Pulsar 이송 함수가 수행되면
    cemStReadMotionState 함수의 모션 상태 반환 값이 8 ( cemMST_WAIT_PLSR ) 이 됩니다. }

    // 지정한 거리만큼 상대좌표 이송을 수행합니다. 모션이 완료되기 전까지 반환되지 않습니다.
    cemPlsrMove ( cemX1, 10000, CE_FALSE );
    // cemPlsrMoveStart()함수는 모션을 시작시킨 후에 바로 반환됩니다.

end;
```

NAME

cemPlsrMoveTo / cemPlsrMoveToStart

- Pulsar Input 에 의한 절대좌표 이송

INFORMATION
1 Manual Pulsar
! VC++ (6, 7, 8)/VB
BCB/Delphi
: Level 5
K 이송 함수
실제 이송이 진행되는 함수이므로, 사전에 반드시 안전을 확인합니다.

SYNOPSIS

r VT_I4 cemPlsrMoveTo ([in] VT_I4 Channel, [in] VT_R8 Position, [in] VT_I4 IsBlocking)

r VT_I4 cemPlsrMoveToStart ([in] VT_I4 Channel, [in] VT_R8 Position)

DESCRIPTION

cemPlsrMoveTo/cemPlsrMoveToStart 함수는 Pulsar 신호 입력에 맞추어 지정한 위치(절대 좌표)로 이송을 수행합니다. 지정한 위치로 이송한 후에는 Manual Pulsar 모드가 자동으로 해제됩니다.

cemPlsrMoveTo 함수는 모션이 완료되기 전까지 반환되지 않으며, cemPlsrMoveToStart 함수는 모션을 시작시킨 후에 바로 반환됩니다.

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*** : 이송할 목표 위치(절대좌표)를 지정합니다. 지정한 위치로 이송한 후에는 Manual Pulsar 모드가 자동으로 해제됩니다. 좌표의 단위는 “Unit Distance” 에 의해 정의되는 논리적 거리입니다.

▶ ***IsBlocking*** : cemPlsrMoveTo 함수의 인자이며, 완료될 때까지 기다리는 동안 윈도우 메시지를 블록 (Blocking)할 것인지를 결정합니다. 단, 쓰레드내에서 실행할 때는 이 값을 1(CE_TRUE)로 설정해 주어야 합니다.

Value	Meaning
0 (CE_FALSE)	윈도우 메시지를 블록(Blocking) 하지 않습니다. 따라서 해당 모션이 진행되는 동안 윈도우 이벤트를 처리합니다.
1 (CE_TRUE)	윈도우 메시지를 블록(Blocking) 합니다. 따라서 해당 모션이 완료되는 동안에는 윈도우 이벤트가 처리되지 않습니다.

RETURN VALUE

Value	Meaning
음수	수행 실패. 자세한 내용은 '에러처리' 편을 참고합니다.
0 (ceERR_NONE)	수행 성공.

SEE ALSO

cemPlsrMove, cemPlsrMoveStart

REFERENCE

□ Manual Pulsar 모드 모션을 중지하기 위해서는 cemSxStopEmg 함수를 사용하십시오. 이 함수가 호출되면 Manual Pulsar 모드는 해제됩니다.

윈도우 이벤트라는 것은 무엇입니까?

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

EXAMPLE

```
//* cemPlsrMove / cemPlsrMoveStart 예제를 참고하여 주시기 바랍니다.
```

NAME

cemPlsrVMoveStart

- Pulsar 입력에 의한 연속 속도 이송

INFORMATION
1 Manual Pulsar
! VC++ (6, 7, 8)/VB
BCB/Delphi
: Level 5
K 이송 함수
실제 이송이 진행되는 함수이므로, 사전에 반드시 안전을 확인합니다.

SYNOPSIS

r VT_I4 cemPlsrVMoveStart ([in] VT_I4 Channel)

DESCRIPTION

cemPlsrVMoveStart 함수는 정지 함수가 호출될 때까지 Pulsar 신호 입력에 맞추어 계속적인 모션을 수행합니다. 모션의 속도는 Pulsar 신호의 주파수에 따라 결정됩니다.

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.

RETURN VALUE

Value	Meaning
음수	수행 실패. 자세한 내용은 '에러처리' 편을 참고합니다.
0 (ceERR_NONE)	수행 성공.

EXAMPLE

```
//* cemPlsrMove / cemPlsrMoveStart 예제를 참고하여 주시기 바랍니다.
```

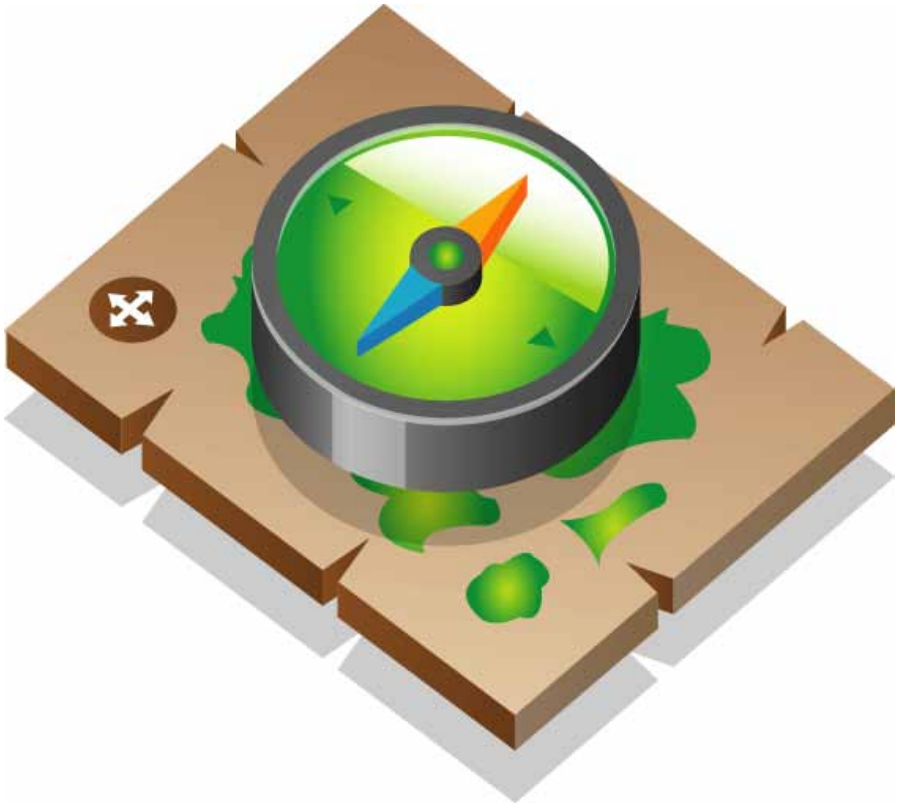
Chapter

10

Monitoring Motion Status

모션 제어의 상태를 확인하는 역할은 고객 여러분들께서 개발하시는 응용 프로그램의 필수 요건 중에 하나입니다. ceSDK 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 감시에 관련된 함수들에 대하여 설명합니다. 상태 감시 함수들은 모션의 상태를 감시하는데 필요한 함수들을 그룹화한 것입니다. 상태 감시에는 모션의 속도, 위치 등을 확인하는 것을 포함하며 이외에도 현재 모션이 진행되고 있는지를 확인하고, 현재 진행되고 있는 모션의 단계 및 각 I/O 상태들을 점검하는 기능도 포함합니다.



10 상태 감시 편

10.1 모션제어 상태(Status) 감시 및 설정

이 단원에서는 모션제어의 상태 감시에 관련된 함수들에 대하여 설명합니다.

상태 감시 함수들은 모션의 상태를 감시하는데 필요한 함수들을 그룹화한 것입니다.

상태 감시에는 모션의 속도, 위치 등을 확인하는 것을 포함하며, 이외에도 현재 모션이 진행되고 있는지를 확인하고, 현재 진행되고 있는 모션의 단계 및 각 I/O 상태들을 확인하는 기능도 포함합니다.

10.1.1 함수 요약

상태 감시 및 제어 함수에 관련된 함수들은 다음과 같습니다.

Summary of Functions
r VT_I4 cemStCount_Set ([in] VT_I4 Axis, [in] VT_I4 Target, [in] VT_I4 Count) 대상 모션 축의 지정한 카운터(Counter) 값을 전달된 매개변수를 통해 설정합니다. 단, 이때 지정하는 카운터 값의 단위는 펄스 수(PPS) 입니다.
r VT_I4 cemStCount_Get ([in] VT_I4 Axis, [in] VT_I4 Source, [out] VT_PI4 Count) 대상 모션 축의 지정한 카운터 값을 전달된 매개변수를 통해 반환합니다. 단, 이때 반환되는 카운터 값의 단위는 펄스 수 입니다.
r VT_I4 cemStPosition_Set ([in] VT_I4 Axis, [in] VT_I4 Target, [in] VT_R8 Position) 대상 모션 축의 지정한 카운터 값을 전달된 매개변수를 통해 설정합니다. 단, 이때 지정하는 카운터 값의 단위는 논리적인 단위 거리(Unit Distance) 입니다.
r VT_I4 cemStPosition_Get ([in] VT_I4 Axis, [in] VT_I4 Source, [out] VT_PR8 Position) 대상 모션 축의 지정한 카운터 값을 전달된 매개변수를 통해 반환합니다. 단, 이때 반환되는 카운터 값의 단위는 논리적인 단위 거리 입니다.
r VT_I4 cemStSpeed_Get ([in] VT_I4 Axis, [in] VT_I4 Source, [out] VT_PR8 Speed) 대상 모션 축의 Command 또는 Feedback 속도를 확인하여, 전달된 매개변수를 통해 논리적 속도 단위로 반환합니다.
r VT_I4 cemStReadMotionState ([in] VT_I4 Axis, [out] VT_PI4 MotStates) 대상 모션 축에 대해서, 현재 모션 동작 상태를 반환합니다.
r VT_I4 cemStReadMioStatuses ([in] VT_I4 Axis, [out] VT_PI4 MioStates) 대상 모션 축에 대해서, 현재 모션 관련 I/O 신호 및 주변 신호(Machine I/O) 상태를 반환합니다.
r VT_I4 cemStGetMstString ([in] VT_I4 MstCode, [out] VT_PSTR Buffer, [in] VT_I4 BufferLen) 대상 모션 축에 대해서, 현재 모션 동작 상태와 관련된 문자열을 반환합니다. 이 문자열은 라이브러리에서 생성된 문자열이므로, 사용자의 의도에 의해 결정된 문자열은 아닙니다.
r VT_I4 cemStReadIOMessageCount ([out] PDWORD IOMessageCount) 수신된 I/O 메시지의 개수를 반환합니다. I/O 메시지는 Digital I/O 와 MIO 등을 모니터링 하기 위해서 주기적으로 수신 받는 메시지입니다.

10.1.2 함수 설명

NAME

cemStCount_Set / cemStCount_Get

- 사용자 정의 하드웨어 카운트 값 설정 및 반환

INFORMATION	
1	Motion Status
!	VC++ (6, 7, 8)/VB
	BCB/Delphi
:	Level 7
J	위험 요소 없음

SYNOPSIS

r VT_I4 cemStCount_Set ([in] VT_I4 Axis, [in] VT_I4 Target, [in] VT_I4 Count)

r VT_I4 cemStCount_Get ([in] VT_I4 Axis, [in] VT_I4 Source, [out] VT_PI4 Count)

DESCRIPTION

cemStCount_Set 함수는 해당 축의 지정한 카운터 값을 새로이 설정합니다. 단, 이때 지정하는 카운터 값의 단위는 펄스 수(PPS)입니다. 이 함수는 카운터의 값을 지정하는 매개변수의 단위가 펄스 수라는 것 을 제외하고는 cemStPosition_Set 함수와 동일합니다.

cemStCount_Get 함수는 해당 축의 지정한 카운터의 값을 읽어서 반환합니다. 단, 이때 반환되는 값의 단위는 펄스 수입니다.

PARAMETER

▶ ***Axis*** : 축 번호. 통합 축으로 관리되는 축 번호를 의미하며, 상수 값으로 0 (Zero Based) 이상, (최대 통합 축 개수 - 1) 이하의 값을 축 번호로 설정할 수 있습니다.

▶ ***Target*** : 설정할 카운터 번호. cemStCount_Set 함수의 인자이며, 다음의 4 가지 값 중의 하나이어야 합니다.

Value	Meaning
0 (cemCNT_COMM)	Command Counter.
1 (cemCNT_FEED)	Feedback Counter.
2 (cemCNT_DEV)	Deviation Counter : Command 와 Feedback 카운터의 편차 카운터.
3 (cemCNT_GEN)	General Counter : 사용자의 정의에 따라 여러 가지 용도로 사용될 수 있는 카운터.

▶ ***Source*** : 대상 카운터 번호. cemStCount_Get 함수의 인자입니다.

▶ ***Count*** : 지정한 값으로 대상 카운터의 값을 설정 혹은 반환합니다. 이 값은 이 값은 논리 단위가 아닌 실제 펄스 카운트 값입니다.

RETURN VALUE

Value	Meaning
음수	수행 실패. 자세한 내용은 '에러처리' 편을 참고합니다.
0 (ceERR_NONE)	수행 성공.

SEE ALSO

cemStPosition_Set, cemStPosition_Get

EXAMPLE

C/C++

```
#include "ceSDK.h"
#include "ceSDKDef.h"


void OnSetCount ()
{
    long nCommCount;          // Command Count
    long nFeedCount;          // Feedback Count

    /* Command, Feedback 카운트 값을 확인한 후, '0' 으로 초기화 합니다. */

    if ( cemStCount_Get ( cemX1, cemCNT_COMM, &nCommCount ) == ceERR_NONE )
    {
            if ( nCommCount != 0 )
            {
                    // Command 카운트를 0 으로 설정합니다.
                    cemStCount_Set ( cemX1, cemCNT_COMM, 0 );
            }
    }

    if ( cemStCount_Get ( cemX1, cemCNT_FEED, &nFeedCount ) == ceERR_NONE )
    {
            if ( nFeedCount != 0 )
            {
                    // Feedback 카운트를 0 으로 설정합니다.
                    cemStCount_Set ( cemX1, cemCNT_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 cemStCount_Get ( cemX1, cemCNT_COMM, nCommCount ) = ceERR_NONE Then
```

```
        If nCommCount <> 0 Then
            ' Command 카운트를 0 으로 설정합니다.
            Call cemStCount_Set ( cemX1, cemCNT_COMM, 0 )
        End If
    End If

    If cemStCount_Get ( cemX1, cemCNT_FEED, nFeedCount ) = ceERR_NONE Then

        If nFeedCount <> 0 Then
            ' Feedback 카운트를 0 으로 설정합니다.
            Call cemStCount_Set ( cemX1, cemCNT_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 cemStCount_Get ( cemX1, cemCNT_COMM, @nCommCount ) = ceERR_NONE then
    begin
        if nCommCount <> 0 then
        begin
            // Command 카운트를 0 으로 설정합니다.
            cemStCount_Set ( cemX1, cemCNT_COMM, 0 );
        end;
    end;

    if cemStCount_Get ( cemX1, cemCNT_FEED, @nFeedCount ) = ceERR_NONE then
    begin
        if nFeedCount <> 0 then
        begin
            // Feedback 카운트를 0 으로 설정합니다.
            cemStCount_Set ( cemX1, cemCNT_FEED, 0 );
        end;
    end;

end;
```

NAME

cemStPosition_Set / cemStPosition_Get

사용자 정의 논리적 카운트 값 설정 및 반환

INFORMATION	
1	Motion Status
!	VC++ (6, 7, 8)/VB
	BCB/Delphi
:	Level 7
J	위험 요소 없음

SYNOPSIS

r VT_I4 cemStPosition_Set ([in] VT_I4 Axis, [in] VT_I4 Target, [in] VT_R8 Position)

r VT_I4 cemStPosition_Get ([in] VT_I4 Axis, [in] VT_I4 Source, [out] VT_PR8 Position)

DESCRIPTION

cemStPosition_Set 함수는 지정한 축의 지정한 카운터의 값을 새로이 설정합니다. 단, 이때 지정하는 카운터 값의 단위는 “Unit Distance” 에 의해 정의되는 논리적 단위 거리입니다. 이 함수는 카운터의 값을 지정하는 매개변수가 논리적 단위 거리라는 것을 제외하고는 cemStCount_Set 함수와 동일합니다.

cemStPosition_Get()함수는 지정한 축의 지정한 카운터의 값을 읽어서 반환합니다. 단, 이때 반환되는 값의 단위는 “Unit Distance” 에 의해 정의되는 논리적 거리입니다.

PARAMETER

▶ ***Axis*** : 축 번호. 통합 축으로 관리되는 축 번호를 의미하며, 상수 값으로 0 (Zero Based) 이상, (최대 통합 축 개수 - 1) 이하의 값을 축 번호로 설정할 수 있습니다.

▶ ***Target*** : 설정할 카운터 번호. cemStPosition_Set 함수의 인자이며, 다음의 4 가지 값 중의 하나이어야 합니다.

Value	Meaning
0 (cemCNT_COMM)	Command Counter.
1 (cemCNT_FEED)	Feedback Counter.
2 (cemCNT_DEV)	Deviation Counter : Command 와 Feedback 카운터의 편차 카운터.
3 (cemCNT_GEN)	General Counter : 사용자의 정의에 따라 여러 가지 용도로 사용될 수 있는 카운터.

▶ ***Source*** : 대상 카운터 번호. cemStPosition_Ge) 함수의 인자입니다.

▶ ***Position*** : 대상 카운터에 설정 혹은 반환될 값. 단, 이 값은 논리적 단위 거리 입니다.

RETURN VALUE

Value	Meaning
음수	수행 실패. 자세한 내용은 '에러처리' 편을 참고합니다.
0 (ceERR_NONE)	수행 성공.

SEE ALSO

cemStCount_Set, cemStCount_Get

REFERENCE

□ 논리적 단위 거리는 cemCfgUnitDist_Set 함수에 의해 결정됩니다

EXAMPLE

```
//* cemStCount_Set / cemStCount_Get 예제를 참고하여 주시기 바랍니다.
```

NAME

cemStSpeed_Get

- 논리적 속도 반환

INFORMATION	
1	Motion Status
!	VC++ (6, 7, 8)/VB
	BCB/Delphi
:	Level 7
J	위험 요소 없음

SYNOPSIS

r VT_I4 cemStSpeed_Get ([in] VT_I4 Axis, [in] VT_I4 Source, [out] VT_PR8 Speed)

DESCRIPTION

이 함수는 Command 또는 Feedback 속도를 읽어서 논리적 속도 단위로 반환합니다. Source 매개변수에 따라서 Command 속도 혹은 Feedback 속도 중 해당하는 속도에 대해서 반환 대상이 결정됩니 다.

PARAMETER

▶ ***Axis*** : 축 번호. 통합 축으로 관리되는 축 번호를 의미하며, 상수 값으로 0 (Zero Based) 이상, (최대 통합 축 개수 - 1) 이하의 값을 축 번호로 설정할 수 있습니다.

▶ ***Source*** : 속도 반환 대상이 되는 카운터 번호. 이 값은 다음의 2 가지 값 중의 하나이어야 합니다.

Value	Meaning
0 (cemCNT_COMM)	Command Counter.
1 (cemCNT_FEED)	Feedback Counter.

▶ ***Speed*** : 전달된 변수를 통해 지정한 카운터의 속도를 읽어서 논리적 속도 단위로 반환합니다.

RETURN VALUE

Value	Meaning
음수	수행 실패. 자세한 내용은 '에러처리' 편을 참고합니다.
0 (ceERR_NONE)	수행 성공.

REFERENCE

□ 논리적 속도 단위는 cemCfgUnitSpeed_Set 함수에 의해 결정됩니다.

NAME

cemStReadMotionState

- 모션 동작 상태 반환

INFORMATION	
1	Motion Status
!	VC++ (6, 7, 8)/VB
	BCB/Delphi
:	Level 7
J	위험 요소 없음

SYNOPSIS

r VT_I4 cemStReadMotionState ([in] VT_I4 Axis, [out] VT_PI4 MotStates)

DESCRIPTION

현재 모션의 동작 상태를 반환합니다.

PARAMETER

▶ ***Axis*** : 축 번호. 통합 축으로 관리되는 축 번호를 의미하며, 상수 값으로 0 (Zero Based) 이상, (최대 통합 축 개수 - 1) 이하의 값을 축 번호로 설정할 수 있습니다.

▶ ***MotStates*** : 모션 상태.

Value	Meaning
음수	모션작업 도중 에러 발생 => 에러코드 참조.
0 (cemMST_STOP)	Stop
1 (cemMST_WAIT_DR)	Waiting for DR input signal
2 (cemMST_WAIT_STA)	Waiting for STA input signal
3 (cemMST_WAIT_INSYNC)	Waiting for internal sync. signal
4 (cemMST_WAIT_OTHER)	Waiting other axis
5 (cemMST_WAIT_ERC)	Waiting for ERC output finished
6 (cemMST_WAIT_DIR)	Waiting for DIR change (DIR 은 외부 입력 신호가 아닌 내부적인 신호임)
7 (cemMST_RESERVED1)	Reserved
8 (cemMST_WAIT_PLSR)	Waiting for PA/PB input signal
9 (cemMST_IN_RVSSPD)	In home special speed (reverse speed)
10 (cemMST_IN_INISPD)	In start velocity motion
11 (cemMST_IN_ACC)	In acceleration
12 (cemMST_IN_WORKSPD)	In working velocity
13 (cemMST_IN_DEC)	In deceleration
14 (cemMST_WAIT_INP)	Waiting for INP input signal
15 (cemMST_SPARE0)	Reserved
16 (cemMST_HOMMING)	In Homming

RETURN VALUE

Value	Meaning
음수	수행 실패. 자세한 내용은 '에러처리' 편을 참고합니다.
0 (ceERR_NONE)	수행 성공.

SEE ALSO

cemStGetMstString

EXAMPLE

C/C++

```
#include “ceSDK.h”
#include “ceSDKDef.h”

void OnGetMotionStste ()
{
    char szMstList[18][20] = {
                                    "Stop",
                                    "Wait DR",
                                    "Wait STA",
                                    "Wait INSYNC",
                                    "Wait Other Axis",
                                    "Wait ERC",
                                    "Wait DIR",
                                    "Reserved1",
                                    "Wait PA/PB",
                                    "On Reverse Speed",
                                    "On Initial Speed",
                                    "On Acceleration",
                                    "On Work Speed",
                                    "On Deceleration",
                                    "Wait INP",
                                    "Reserved",
                                    “In Homming”,
                                    “”
                                    };

    long nMotState;             // 모션 상태 정보.

    // 모션 상태를 반환합니다.
    cemStReadMotionState ( cemX1, &nMST );

    if ( nMotState < 0 )        // 모션 상태가 음수이면 마지막 모션 작업에 에러가 발생한 상태 입니다.
    {
            OutputDebugString ( "ReadMotionState Error!" );
    }


    CString sMsg = "Current Motion State : " + szMstList[nMotState];

    //DisplayStatus() 함수는 화면에 상태를 표시하는 가상의 함수입니다.
    DisplayStatus( sMsg );
}
```

Visual Basic

```
‘ 모션 상태를 얻어와 지정한 Label Component 에 업데이트 하는 예제입니다.

Private Sub OnGetMotionStste ()

    Dim nMotState As Long                ‘ 모션 상태 정보.

    If cemStReadMotionState ( cemX1, &nMotState ) = ceERR_NONE Then

            Select Case nMotState

                    Case cemMST_STOP : lblState.Caption = "Stop"
                    Case cemMST_WAIT_DR : lblState.Caption = "Wait DR"
                    Case cemMST_WAIT_STA : lblState.Caption = "Wait STA"
                    Case cemMST_WAIT_INSYNC : lblState.Caption = "Wait INSYNC"
                    Case cemMST_WAIT_OTHER : lblState.Caption = "Wait Other Axis"
                    Case cemMST_WAIT_ERC : lblState.Caption = "Wait ERC"
                    Case cemMST_WAIT_DIR : lblState.Caption = "Wait DIR"
                    ‘cemMST_RESERVED1 은 내부적으로 예약되었기 때문에 사용하지 않습니다.
                    ‘Case cemMST_RESERVED1 : lblState.Caption = "Reserved1"
                    Case cemMST_WAIT_PLSR : lblState.Caption = “Wait PA/PB"
                    Case cemMST_IN_RVSSPD : lblState.Caption = "On Reverse Speed"
                    Case cemMST_IN_INISPD : lblState.Caption = "On Initial Speed"
                    Case cemMST_IN_ACC : lblState.Caption =  "On Acceleration"
                    Case cemMST_IN_WORKSPD : lblState.Caption = "On Work Speed"
                    Case cemMST_IN_DEC : lblState.Caption =  "On Deceleration"
                    Case cemMST_WAIT_INP : lblState.Caption = "Wait INP"
                    ‘cemMST_SPARE0 은 내부적으로 예약되었기 때문에 사용하지 않습니다.
                    ‘ Case cemMST_SPARE0 : lblState.Caption = "Reserved2"
                    Case cemMST_HOMMING : lblState.Caption = “In Homming”

            Emd Select
    End If

End Sub
```

Delphi

```
function GetMitonStateString( i : Integer ) : String;
begin
    Case i of

            cemMST_STOP : Result := 'Stop';
            cemMST_WAIT_DR : Result := 'Waiting for DR input signal';
            cemMST_WAIT_STA : Result := 'Waiting for STA input signal';
            cemMST_WAIT_INSYNC : Result := 'Waiting for internal sync, signal';
            cemMST_WAIT_OTHER : Result := 'Waiting other axis';
            cemMST_WAIT_ERC : Result := 'Waiting for ERC output finished';
            cemMST_WAIT_DIR : Result := 'Waiting for DIR change';
            // cemMST_RESERVED1 은 내부적으로 예약되었습니다. 사용하지 않습니다.
            //cemMST_RESERVED1 : Result := 'RESERVED1';
            cemMST_WAIT_PLSR : Result := 'Waiting for PA/PB input signal';
            cemMST_IN_RVSSPD : Result := 'In home special speed (reverse speed)';
            cemMST_IN_INISPD : Result := 'In start velocity motion';
            cemMST_IN_ACC : Result := 'In Acceleration';
```

```
            cemMST_IN_WORKSPD : Result := 'In Working Velocity';
            cemMST_IN_DEC : Result := 'In Deceleration';
            cemMST_WAIT_INP : Result := 'Waiting for INP input  signal';
            // cemMST_SPARE0 는 내부적으로 예약되었습니다. 사용하지 않습니다.
            //cemMST_SPARE0 : Result := 'RESERVED2';
    end;
end;

procedure OnEvent ();
var
    stateus : LongInt;
    i : Integer;

begin

    // 에러 발생하면, 에러 메시지를 화면에 나타냅니다.
    if cemStReadMotionState(cmX1, @stateus <> cmERR_NONE then
    begin
            ShowMessage ( 'cemStReadMotionState has been failed' );
            exit;
    end;

    if stateus < 0 then
    begin
            ShowMessage ( '모션 동작 상태에 에러가 발생하였습니다.' );
            // OnEvent procedure 를 종료합니다.
            exit;
    end;
end;
```

NAME

cemStReadMioStatuses

- 모션 관련 I/O (Motion I/O) 상태 반환

INFORMATION	
1	Motion Status
!	VC++ (6, 7, 8)/VB
	BCB/Delphi
:	Level 7
J	위험 요소 없음

SYNOPSIS

r VT_I4 cemStReadMioStatuses ([in] VT_I4 Axis, [out] VT_PI4 MioStates)

DESCRIPTION

이 함수는 현재 모션과 관련된 여러 가지 MIO 상태를 반환합니다.

각 비트별로 할당된 MIO 의 상태를 표시하므로 사용자는 비트 마스크를 수행하여 원하는 I/O 의 상태를 확인하여야 합니다.

PARAMETER

▶ ***Axis*** : 축 번호. 통합 축으로 관리되는 축 번호를 의미하며, 상수 값으로 0 (Zero Based) 이상, (최대 통합 축 개수 - 1) 이하의 값을 축 번호로 설정할 수 있습니다.

▶ ***MioStates*** : 이 매개변수를 통하여 Machine I/O 상태를 반환합니다.

Bit No.	Name	Meaning
0 (cemIOST_RDY)	RDY	Servo ready signal input status(1=ON)
1 (cemIOST_ALM)	ALM	Alarm signal status(1=ON)
2 (cemIOST_ELN)	-EL	Negative limit switch status(1=ON)
3 (cemIOST_ELP)	+EL	Positive limit switch status(1=ON)
4 (cemIOST_ORG)	ORG	Origin switch status(1=ON)
5 (cemIOST_DIR)	DIR	Operating direction status(1=ON)
6 (cemIOST_EZ)	EZ	Index signal status(1=ON)
7 (cemIOST_LTC)	LTC	Latch signal input status(1=ON)
8 (cemIOST_SD)	SD	Slow Down signal input status(1=ON)
9 (cemIOST_INP)	INP	In-Position signal input status(1=ON)
10 (cemIOST_DRN)	DRN	-DR input signal status(1=ON)
11 (cemIOST_DRP)	DRP	+DR input signal status(1=ON)
12 (cemIOST_STA)	STA	STA input signal status(1=ON)
13 (cemIOST_STP)	STP	STP input signal status(1=ON)
14 (cemIOST_ALMR)	ALMR	Alarm Reset output signal status(1=ON)
15 (cemIOST_EMG)	EMG	Emergency output signal status(1=ON)
16 (cemIOST_SVON)	SVON	Servo-ON output signal status(1=ON)
17 (cemIOST_HOMS)	HOMS	(1=ON)
18 (cemIOST_PLSA)	PLSA	(1=ON)
19 ~31		Reserved

RETURN VALUE

Value	Meaning
음수	수행 실패. 자세한 내용은 '에러처리' 편을 참고합니다
0 (ceERR_NONE)	수행 성공

REFERENCE

□ 범용적인 모션 응용 프로그램에서는 MIO(Machine I/O) 상태를 표현하기 위한 용도로 본 함수의 사용 빈도가 매우 높습니다.

EXAMPLE

C/C++

```
#include "ceSDK.h"
#include "ceSDKDef.h"

void OnGetMioStates ()
{
    long nMioStates;            // MIO 상태 정보.

    // 해당 축의 MIO State Bit 가 설정된 32Bit 값을 반환합니다.
    if ( cemStReadMioStatuses ( cemX1, &nMioStates ) == ceERR_NONE )
    {
            // nMioState 의 값을 오른쪽으로 쉬프트 연산하여, 해당 상태 값을 얻습니다.
            bool RDY_State = ( nMioStates >> cemIOST_RDY ) & 0x01;
            bool ALM_State = ( nMioStates >> cemIOST_ALM ) & 0x01;
            bool ELN_State = ( nMioStates >> cemIOST_ELN ) & 0x01;
            …
            …
    }
}
```

Visual Basic

```
Private Sub OnGetMioStates ()

    Dim nMioStates          ' MIO 상태 정보
    Dim nRdyState As Long, nAlmState As Long, nElnState As Long
    Dim bRdyState As Boolean, bAlmState As Boolean, bElnState As Boolean

    ' 해당 축의 MIO State Bit 가 설정된 32Bit 값을 반환합니다.
    If cemStReadMioStatuses ( cemX1, nMioStates ) = ceERR_NONE Then

            ' nMioState 의 값을 오른쪽으로 쉬프트 연산하여, 해당 상태 값을 얻습니다.
            ' ceSDK 는 VB 사용자를 위해 편리한 비트 연산 함수를 제공합니다.
            Call cemGnBitShift ( nMioStates, cemIOST_RDY, nRdyState )
            Call cemGnBitShift ( nMioStates, cemIOST_ALM, nAlmState )
            Call cemGnBitShift ( nMioStates, cemIOST_ELN, nElnState )
            …
            …
```

```
        bRdyState = nRdyState And &H1
        bAlmState = nAlmState And &H1
        bElnState = nElnState And &H1
        …
        …
    End If

End Sub
```

Delphi

```
var
    dwMioState : LongInt;
    RDY_State : Boolean;
    ALM_State : Boolean;
    ELN_State : Boolean;

begin
    cemStReadMioStatuses ( cmX1,@dwMioState );

    // dwMioState 의 값을 오른쪽으로 쉬프트 연산(Shift Operation) 하여, 해당 상태 값을 얻습니다.
    RDY_State := Boolean ( ( dwMioState shr cemIOST_RDY ) and $1);
    ALM_State := Boolean ( ( dwMioState shr cemIOST_ALM ) and $1);
    ELN_State := Boolean ( ( dwMioState shr cemIOST_ELN ) and $1);
    …
    …

end;
```

NAME

cemStGetMstString

- 모션 동작 상태와 관련된 문자열 반환

INFORMATION	
1	Motion Status
!	VC++ (6, 7, 8)/VB
	BCB/Delphi
:	Level 7
J	위험 요소 없음

SYNOPSIS

r VT_I4 cemStGetMstString

([in] VT_I4 MstCode, [out] VT_PSTR Buffer, [in] VT_I4 BufferLen)

DESCRIPTION

이 함수는 현재 모션 동작 상태와 관련된 문자열을 반환합니다.

ceSDK 의 모션 동작 상태를 확인 하는 cemStReadMotionState 함수를 통해 반환된 모션 동작 상태 값을 매개변수로 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. cemStReadMotionState 함수를 통해 지정된 상태 값을 알아볼 수 있으며, 해당 상태 값을 이 함수의 매개변수로 활용할 수 있습니다.

▶ ***Buffer*** : 모션 동작 상태와 관련된 문자열을 복사해 올 문자열 버퍼의 주소입니다.

▶ ***BufferLen*** : 모션 동작 상태를 확인하기 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.
0 (ceERR_NONE)	수행 성공.

SEE ALSO

cemStReadMotionState

EXAMPLE

C/C++

```
#include “ceSDK.h”
#include “ceSDKDef.h”

void OnGetMstString ()
{
    char szBuffer[32] = {0, };

    // 현재 Motion State 를 Buffer 에 문자열로 저장합니다.
    if ( cemStGetMstString ( cemX1, szBuffer, 32 ) == ceERR_NONE )
    {
            //DisplayMstString() 함수는 화면에 문자를 표시하는 가상의 함수입니다.
            DisplayMstString( szBuffer );
    }
}
```

NAME

cemStReadIOMessageCount

- 입·출력(I/O) 메시지 카운트

INFORMATION	
1	Motion Status
!	VC++ (6, 7, 8)/VB
	BCB/Delphi
:	Level 7
J	위험 요소 없음

SYNOPSIS

r VT_I4 cemStReadIOMessageCount ([out] PDWORD IOMessageCount)

DESCRIPTION

수신된 I/O 메시지의 개수를 알 수 있습니다. I/O 메시지는 Digital I/O 와 MIO 등을 모니터링 하기 위해서 주기적으로 수신 받는 메시지입니다.

PARAMETER

▶ *IOMessageCount* : 이 매개변수를 통하여 수신된 I/O 메시지 개수를 반환합니다.

RETURN VALUE

Value	Meaning
음수	수행 실패. 자세한 내용은 '에러처리' 편을 참고합니다.
0 (ceERR_NONE)	수행 성공.

EXAMPLE

C/C++

```
#include “ceSDK.h”
#include “ceSDKDef.h”

DWORD dwIOMsgCount = 0;

//수신된 I/O 메시지의 개수를 반환합니다.
cemStReadIOMessageCount( &dwIOMsgCount );
```

10.2 위치 값 래치(Position Latch)

Position Latch 는 특정 순간에 Motion 의 위치 관련 카운터 값을 래치(Latch)하여 읽을 수 있도록 하는 기능입니다.

Position Latch 는 LTC 입력 핀에 LATCH 신호가 입력되면 그 순간의 각 카운트 값을 래치(Latch) 합니다.

이 기능은 LTC 핀에 입력되는 하드웨어 신호에 동기되어서 각 위치값이 래치 되므로 시간 지연이 발생하지 않습니다.

10.2.1 함수 요약

Position Latch 함수에 관련된 함수들은 다음과 같습니다.

Summary of Functions
r VT_I4 cemLtcIsLatched ([in] VT_I4 Axis, [out] VT_PI4 IsLatched) 대상 모션 축의 래치 카운터(Latch Counter) 가 활성화 되었음을 확인하고, 결과를 반환합니다.
r VT_I4 cemLtcReadLatch ([in] VT_I4 Axis, [in] VT_I4 Counter, [out] VT_PR8 LatchedPos) 대상 모션 축의 래치 카운터에 저장된 카운터(Counter) 값을 반환합니다.
r VT_I4 cemLtcQue_Alloc ([in] VT_I4 Axis, [in] VT_I4 QueSize, [in] VT_I4 SrcCntr) 대상 모션 축의 래치 카운터의 Queue 버퍼를 지정한 크기로 설정합니다.
r VT_I4 cemLtcQue_Free ([in] VT_I4 Axis) 대상 모션 축의 래치 카운터의 Queue 버퍼 할당을 해제 합니다.
r VT_I4 cemLtcQue_GetSize ([in] VT_I4 Axis, [out] VT_PI4 QueSize) 대상 모션 축의 래치 카운터의 Queue 버퍼 크기를 얻습니다.
r VT_I4 cemLtcQue_Reset ([in] VT_I4 Axis) 대상 모션 축의 래치 카운터의 Queue 를 모두 비우고 Push 및 Pop Count 를 모두 리셋합니다.
r VT_I4 cemLtcQue_Check ([in] VT_I4 Axis, [out] VT_PI4 NonReadCount) 대상 모션 축의 래치 카운터의 Queue 버퍼에 저장된 데이터 수를 반환합니다.
r VT_I4 cemLtcQue_Pop ([in] VT_I4 Axis, [out] VT_PR8 fLtcVal) 대상 모션 축의 래치 카운터의 Queue 버퍼에서 읽혀지지 않은 데이터를 반환합니다. 이때 해당 Queue 의 Pop 카운트를 자동으로 1 증가 시킵니다.
r VT_I4 cemLtcQue_GetAt ([in] VT_I4 Axis, [in] VT_I4 Idx, [out] VT_PR8 fLtcVal) 대상 모션 축의 래치 카운터의 Queue 버퍼에서 지정된 인덱스에 해당하는 데이터를 반환합니다.

NAME

cemLtcIsLatched

- 래치 카운터(Latch Counter) 활성화 여부 확인

INFORMATION	
1	Latch
!	VC++ (6, 7, 8)/VB
	BCB/Delphi
:	Level 7
J	위험 요소 없음

SYNOPSIS

r VT_I4 cemLtcIsLatched ([in] VT_I4 Axis, [out] VT_PI4 IsLatched)

DESCRIPTION

해당 축의 래치 카운터가 활성화 되었음을 확인하고, 결과를 반환합니다.

PARAMETER

▶ ***Axis*** : 축 번호. 통합 축으로 관리되는 축 번호를 의미하며, 상수 값으로 0 (Zero Based) 이상, (최대 통합 축 개수 - 1) 이하의 값을 축 번호로 설정할 수 있습니다.

▶ ***IsLatched*** : 해당 축의 래치 카운터의 활성화 여부를 반환합니다.

Value	Meaning
0 (CE_FALSE)	래치 카운터 비활성.
1 (CE_TRUE)	래치 카운터 활성.

RETURN VALUE

Value	Meaning
음수	수행 실패. 자세한 내용은 '에러처리' 편을 참고합니다.
0 (ceERR_NONE)	수행 성공.

SEE ALSO

cemLtcReadLatch

EXAMPLE

C/C++

```
#include “ceSDK.h”
#include “ceSDKDef.h”

// 0 번 축의 래치 카운터 활성화 여부를 반환합니다.
long nIsLatched = CE_FALSE;
cemLtcIsLatched( cemX1, &nIsLatched );
```

NAME

cemLtcReadLatch

- 래치 카운터(Latch Counter) 값 확인

INFORMATION
1 Latch
! VC++ (6, 7, 8)/VB
BCB/Delphi
: Level 7
J 위험 요소 없음

SYNOPSIS

r VT_I4 cemLtcReadLatch ([in] VT_I4 Axis, [in] VT_I4 Counter, [out] VT_PR8 LatchedPos)

DESCRIPTION

지정한 축의 현재 래치 된 카운트 값을 반환합니다. 이때 반환되는 위치 값의 단위는 논리적 거리 단위(Unit Distance)가 적용됩니다.

PARAMETER

▶ ***Axis*** : 축 번호. 통합 축으로 관리되는 축 번호를 의미하며, 상수 값으로 0 (Zero Based) 이상, (최대 통합 축 개수 - 1) 이하의 값을 축 번호로 설정할 수 있습니다.

▶ ***Counter***: 읽을 래치 카운터를 지정합니다. 이 값은 다음과 같습니다.

Value	Meaning
0 (cemCNT_COMM)	명령 위치 카운터. (Command position counter)
1 (cemCNT_FEED)	실제 위치 카운터. (Feedback position counter)
2 (cemCNT_DEV)	Deviation 또는 펄스 출력 속도.

▶ ***LatchedPos*** : 지정한 축의 래치 된 카운트 값을 반환합니다. 이때 반환되는 위치 값의 단위는 논리적 거리 단위가 적용됩니다.

RETURN VALUE

Value	Meaning
음수	수행 실패. 자세한 내용은 '에러처리' 편을 참고합니다.
0 (ceERR_NONE)	수행 성공.

SEE ALSO

cemLtcIsLatched

EXAMPLE

C/C++

```
#include “ceSDK.h”
#include “ceSDKDef.h”

// 0 번 축의 래치 카운트를 Feedback position counter 로 설정하고 래치된 카운트 값을 반환합니다.
long nLtcCounter = cemCNT_FEED;
double fLatchedPos = 0.0f;
cemLtcReadLatche( cemX1, nLtcCounter, &nLatchedPos );
```

NAME

cemLtcQue_Alloc

- 래치 카운터(Latch Counter) 큐(Queue) 버퍼 설정

INFORMATION	
1	Latch
!	VC++ (6, 7, 8)/VB
	BCB/Delphi
:	Level 7
J	위험 요소 없음

SYNOPSIS

r VT_I4 cemLtcQue_Alloc ([in] VT_I4 Axis, [in] VT_I4 QueSize, [in] VT_I4 SrcCntr)

DESCRIPTION

대상 모션 축의 래치 카운터(Latch Counter) 의 큐(Queue) 버퍼를 지정한 크기로 설정합니다.

초기 버퍼 크기는 '0' 으로 설정되어 있습니다.

PARAMETER

▶ ***Axis*** : 축 번호. 통합 축으로 관리되는 축 번호를 의미하며, 상수 값으로 0 (Zero Based) 이상, (최대 통합 축 개수 - 1) 이하의 값을 축 번호로 설정할 수 있습니다.

▶ ***QueSize*** : Queue 버퍼 크기. 최대 버퍼 크기는 1024 입니다.

▶ ***SrcCntr*** : 저장할 대상 래치 카운터를 설정합니다.

Value	Meaning
0 (cemCNT_COMM)	명령 위치 카운터. (Command position counter)
1 (cemCNT_FEED)	실제 위치 카운터. (Feedback position counter)
2 (cemCNT_DEV)	Deviation 또는 펄스 출력 속도.

RETURN VALUE

Value	Meaning
음수	수행 실패. 자세한 내용은 '에러처리' 편을 참고합니다.
0 (ceERR_NONE)	수행 성공.

SEE ALSO

cemLtcQue_Free

EXAMPLE

C/C++

```
#include “ceSDK.h”
#include “ceSDKDef.h”

// 0 번 축의 래치 카운터의 큐 버퍼를 512 로 설정합니다.
long nQueSize = 512;
if(cemLtcQue_Alloc ( cemX1, nQueSize, cemCNT_FEED ) != ceErr_None)
{
    OutputDebugString ( “cemLtcQue_Alloc function failed” );
}
```

NAME

cemLtcQue_Free

- 래치 카운터(Latch Counter) 큐(Queue) 버퍼 할당 해제

INFORMATION	
1	Latch
!	VC++ (6, 7, 8)/VB
	BCB/Delphi
:	Level 7
J	위험 요소 없음

SYNOPSIS

r VT_I4 cemLtcQue_Free ([in] VT_I4 Axis)

DESCRIPTION

대상 모션 축의 래치 카운터(Latch Counter) 의 큐(Queue) 버퍼 할당을 해제 합니다.

PARAMETER

▶ ***Axis*** : 축 번호. 통합 축으로 관리되는 축 번호를 의미하며, 상수 값으로 0 (Zero Based) 이상, (최대 통합 축 개수 - 1) 이하의 값을 축 번호로 설정할 수 있습니다.

RETURN VALUE

Value	Meaning
음수	수행 실패. 자세한 내용은 '에러처리' 편을 참고합니다.
0 (ceERR_NONE)	수행 성공.

SEE ALSO

cemLtcQue_Alloc

EXAMPLE

C/C++

```
#include “ceSDK.h”
#include “ceSDKDef.h”

// 0 번 축의 래치 카운터의 큐 버퍼 할당을 해제 합니다.
if(cemLtcQue_Free ( cemX1 ) != ceErr_None)
{
    OutputDebugString ( “cemLtcQue_Free function failed” );
}
```

NAME

cemLtcQue_GetSize

- 래치 카운터(Latch Counter) 큐(Queue) 버퍼 크기 반환

INFORMATION	
1	Latch
!	VC++ (6, 7, 8)/VB
	BCB/Delphi
:	Level 7
J	위험 요소 없음

SYNOPSIS

r VT_I4 cemLtcQue_GetSize ([in] VT_I4 Axis, [out] VT_PI4 QueSize)

DESCRIPTION

대상 모션 축의 래치 카운터(Latch Counter) 의 큐(Queue) 버퍼 크기를 반환합니다.

PARAMETER

▶ ***Axis*** : 축 번호. 통합 축으로 관리되는 축 번호를 의미하며, 상수 값으로 0 (Zero Based) 이상, (최대 통합 축 개수 – 1) 이하의 값을 축 번호로 설정할 수 있습니다.

▶ ***QueSize*** : 이 매개변수를 통하여 Queue 버퍼 크기를 반환합니다.

RETURN VALUE

Value	Meaning
음수	수행 실패. 자세한 내용은 '에러처리' 편을 참고합니다.
0 (ceERR_NONE)	수행 성공.

SEE ALSO

cemLtcQue_Alloc

EXAMPLE

C/C++

```
#include “ceSDK.h”
#include “ceSDKDef.h”

// 0 번 축의 래치 카운터의 큐 버퍼 의 할당된 크기를 얻습니다.
long nQueSize;
if(cemLtcQue_GetSize ( cemX1, &nQueSize ) != ceErr_None)
{
    OutputDebugString ( “cemLtcQue_GetSize function failed” );
}
```

NAME

cemLtcQue_Reset

- 래치 카운터(Latch Counter) 큐(Queue) 버퍼 초기화

INFORMATION	
1	Latch
!	VC++ (6, 7, 8)/VB
	BCB/Delphi
:	Level 7
J	위험 요소 없음

SYNOPSIS

r VT_I4 cemLtcQue_Reset ([in] VT_I4 Axis)

DESCRIPTION

대상 모션 축의 래치 카운터(Latch Counter) 의 큐(Queue) 버퍼를 모두 비우고 Push 및 Pop Count 를 모두 '0'으로 리셋합니다.

PARAMETER

▶ ***Axis*** : 축 번호. 통합 축으로 관리되는 축 번호를 의미하며, 상수 값으로 0 (Zero Based) 이상, (최대 통합 축 개수 - 1) 이하의 값을 축 번호로 설정할 수 있습니다.

RETURN VALUE

Value	Meaning
음수	수행 실패. 자세한 내용은 '에러처리' 편을 참고합니다.
0 (ceERR_NONE)	수행 성공.

SEE ALSO

cemLtcQue_Alloc

EXAMPLE

C/C++

```
#include “ceSDK.h”
#include “ceSDKDef.h”

// 0 번 축의 래치 카운터의 큐 버퍼를 초기화 시킵니다.
if(cemLtcQue_Reset ( cemX1 ) != ceErr_None)
{
    OutputDebugString ( “cemLtcQue_Reset function failed” );
}
```

NAME

cemLtcQue_Check

- 래치 카운터(Latch Counter) 큐(Queue) 버퍼 에 저장된 데이터 수 반환

INFORMATION	
1	Latch
!	VC++ (6, 7, 8)/VB
	BCB/Delphi
:	Level 7
J	위험 요소 없음

SYNOPSIS

r VT_I4 cemLtcQue_Check ([in] VT_I4 Axis, [out] VT_PI4 NonReadCount)

DESCRIPTION

대상 모션 축의 래치 카운터(Latch Counter) 의 큐(Queue) 버퍼에 저장된 데이터 수를 반환합니다. 즉, 버퍼에 읽혀지지 않은 데이터가 몇 개 남아 있는지를 알아보는 함수입니다.

PARAMETER

▶ ***Axis*** : 축 번호. 통합 축으로 관리되는 축 번호를 의미하며, 상수 값으로 0 (Zero Based) 이상, (최대 통합 축 개수 - 1) 이하의 값을 축 번호로 설정할 수 있습니다.

▶ ***NonReadCount*** : 이 매개변수를 통하여 현재 큐에 있는 읽혀지지 않은 데이터 수를 반환합니다.

RETURN VALUE

Value	Meaning
음수	수행 실패. 자세한 내용은 '에러처리' 편을 참고합니다.
0 (ceERR_NONE)	수행 성공.

SEE ALSO

cemLtcQue_Pop

EXAMPLE

C/C++

```
#include “ceSDK.h”
#include “ceSDKDef.h”

// 0 번 축의 래치 카운터의 큐 버퍼에 래치된 데이터 개수를 얻습니다.
long nNonReadCount;
if(cemLtcQue_Check ( cemX1, &nNonReadCount) != ceErr_None)
{
    OutputDebugString ( “cemLtcQue_Check function failed” );
}
```

NAME

cemLtcQue_Pop

- 래치 카운터(Latch Counter) 큐(Queue) 버퍼의 읽혀지지 않은 데이터를 반환

INFORMATION	
1	Latch
!	VC++ (6, 7, 8)/VB
	BCB/Delphi
:	Level 7
J	위험 요소 없음

SYNOPSIS

r VT_I4 cemLtcQue_Pop ([in] VT_I4 Axis, [out] VT_PR8 fLtcVal)

DESCRIPTION

대상 모션 축의 래치 카운터(Latch Counter) 의 큐(Queue) 버퍼에서 읽혀지지 않은 데이터를 반환합니다. 즉, FIFO 형태의 버퍼에서 선두 위치의 래치된 데이터를 얻는 함수입니다.

데이터를 읽은 후 자동으로 Pop 카운트가 1 만큼 증가됩니다.

PARAMETER

▶ *Axis* : 축 번호. 통합 축으로 관리되는 축 번호를 의미하며, 상수 값으로 0 (Zero Based) 이상, (최대 통합 축 개수 - 1) 이하의 값을 축 번호로 설정할 수 있습니다.

▶ *fLtcVal* : 이 매개변수를 통하여 래치된 데이터 중 Queue 버퍼에서 읽혀지지 않은 데이터를 반환합니다.

RETURN VALUE

Value	Meaning
음수	수행 실패. 자세한 내용은 '에러처리' 편을 참고합니다.
0 (ceERR_NONE)	수행 성공.

SEE ALSO

cemLtcQue_Pop

EXAMPLE

C/C++

```
#include “ceSDK.h”
#include “ceSDKDef.h”

// 0 번 축의 래치 카운터의 큐 버퍼에 래치된 선두 위치의 데이터를 얻습니다.
double fLtcVal;
if(cemLtcQue_Pop ( cemX1, &fLtcVal) != ceErr_None)
{
    OutputDebugString ( “cemLtcQue_Pop function failed” );
}
```

NAME

cemLtcQue_GetAt

- 래치 카운터(Latch Counter) 큐(Queue) 버퍼에서 지정된 인덱스에 해당하는 데이터 반환

INFORMATION	
1	Latch
!	VC++ (6, 7, 8)/VB
	BCB/Delphi
:	Level 7
J	위험 요소 없음

SYNOPSIS

r VT_I4 cemLtcQue_GetAt ([in] VT_I4 Axis, [in] VT_I4 Idx, [out] VT_PR8 fLtcVal)

DESCRIPTION

대상 모션 축의 래치 카운터(Latch Counter) 의 큐(Queue) 버퍼에서 지정된 인덱스에 해당하는 버퍼 데이터를 반환합니다.

PARAMETER

▶ ***Axis*** : 축 번호. 통합 축으로 관리되는 축 번호를 의미하며, 상수 값으로 0 (Zero Based) 이상, (최대 통합 축 개수 - 1) 이하의 값을 축 번호로 설정할 수 있습니다.

▶ ***Idx*** : Queue 버퍼의 인덱스를 지정합니다.

▶ ***fLtcVal*** : 이 매개변수를 통하여 래치된 데이터 중 Queue 버퍼의 지정된 인덱스에 해당하는 버퍼 데이터를 반환합니다.

RETURN VALUE

Value	Meaning
음수	수행 실패. 자세한 내용은 '에러처리' 편을 참고합니다.
0 (ceERR_NONE)	수행 성공.

SEE ALSO

cemLtcQue_Pop

EXAMPLE

C/C++

```
#include “ceSDK.h”
#include “ceSDKDef.h”

// 0 번 축의 래치 카운터의 큐 버퍼에 래치된 지정된 인덱스 위치의 데이터를 얻습니다.
double fLtcVal;
long idx = 7;
if(cemLtcQue_GetAt ( cemX1, idx, &fLtcVal) != ceErr_None) // 래치 큐 버퍼에서 7 번째 데이터를 얻습니다.
{
    OutputDebugString ( “cemLtcQue_GetAt function failed” );
}
```

Chapter

11

Motion Digital I/O Control

외부 디지털 입력과 출력을 위해 마련된 모션 디지털 입·출력 기능에 대해서 ceSDK 는 매우 유연한 입·출력 기능을 제공하고 있습니다. 특히 SW 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 디지털 입·출력은 주요 외부 입·출력 인터페이스(Interface)의 하나로서, 소위 디지털 접점(Digital Contact) 이라는 말로 일컫습니다. ceSDK 는 모션 제품의 자체 디지털 입·출력 인터페이스를 제공합니다. 이 장에서는 편리한 ceSDK 의 디지털 입·출력 기능들을 자세히 살펴보고, 활용할 수 있도록 구성하였습니다.

11 모션 디지털 입·출력 편

㈜커미조아 모션 모듈은 모션제어 전용 I/O 뿐만 아니라 범용적으로 사용할 수 있는 디지털 입·출력 채널을 제공합니다.

모션 디지털 입·출력 채널에 대한 명시적인 채널 번호의 관계이므로, 이 점을 잘 숙지해 주시기를 부탁드립니다.

디지털 입력 채널과 디지털 출력 채널의 번호는 서로 상관관계를 가지지 않습니다. 독립적인 채널 번호로 순서화 됩니다.

원격 노드의 번호, 노드 내의 모션 모듈 번호를 기준으로 디지털 입력 채널과 디지털 출력 채널 번호가 증가됩니다.

11.1 함수 요약

“모션 디지털 입·출력” 기능과 관련된 함수는 다음의 표와 같습니다.

Summary of Functions
r VT_I4 cemDiOne_Get ([in] VT_I4 Channel, [out] VT_PI4 State) 모션 디지털 입력이 지원되는 제품에서 단일 채널에 대해 입력 상태를 확인합니다.
r VT_I4 cemDiMulti_Get ([in] VT_I4 IniChannel, [in] VT_I4 NumChannels, [out] VT_PI4 InputState) 모션 디지털 입력이 지원되는 제품에서 다중 채널에 대해 입력 상태를 확인합니다.
r VT_I4 cemDoOne_Put ([in] VT_I4 Channel, [in] VT_I4 OutState) 모션 디지털 출력이 지원되는 제품에서 단일 채널에 대해 출력을 발생합니다.
r VT_I4 cemDoOne_Get ([in] VT_I4 Channel, [out] VT_PI4 OutState) 모션 디지털 출력이 지원되는 제품에서 단일 채널에 대해 출력 상태를 확인합니다.
r VT_I4 cemDoMulti_Put ([in] VT_I4 IniChannel, [in] VT_I4 NumChannels, [in] VT_I4 OutStates) 모션 디지털 출력이 지원되는 제품에서 다중 채널에 대해 출력을 발생합니다.
r VT_I4 cemDoMulti_Get ([in] VT_I4 IniChannel, [in] VT_I4 NumChannels, [out] VT_PI4 OutStates) 모션 디지털 출력이 지원되는 제품에서 다중 채널에 대해 출력 상태를 확인합니다.

11.2 함수 설명

NAME

cemDiOne_Get

- 모션 디지털 입력이 지원되는 제품에서 단일 채널에 대해 입력 상태 반환

INFORMATION	
1	Motion DIO Control
!	VC++ (6, 7, 8)/VB BCB/Delphi
:	Level 1
J	위험 요소 없음

SYNOPSIS

r VT_I4 cemDiOne_Get ([in] VT_I4 Channel, [out] VT_PI4 State)

DESCRIPTION

모션 디지털 입력이 지원되는 제품에서 단일 채널에 대한 디지털 입력 상태를 반환합니다.

PARAMETER

▶ ***Channel*** **:** 채널 번호. 통합 채널로 관리되는 채널 번호를 의미하며, 상수 값으로 0 (Zero Based) 이상, (최대 통합 채널수 - 1) 이하의 값을 채널 번호로 설정할 수 있습니다.

▶ ***InputState*** **:** 해당 채널의 디지털 입력(Digital Input) 상태를 확인합니다.

Value	Meaning
0 (CE_FALSE)	OFF
1 (CE_TRUE)	ON

RETURN VALUE

Value	Meaning
음수	수행 실패. 자세한 내용은 '에러처리' 편을 참고합니다.
0 (ceERR_NONE)	수행 성공.

EXAMPLE

C/C++

```
#include “ceSDK.h”
#include “ceSDKDef.h”

/* MDO 1 번 채널의 출력을 제어하고, MDO 1 번 채널의 출력을 MDI 1 번 채널을 통해 입력 받아 확인합니다. */

long nMdiChNo = 1;              // MDI 채널 번호 설정
long nMdoChNo = 1;              // MDO 채널 번호 설정
long nMdiState, nMdoState;      // MDIO 상태 정보

// MDO 1 번 채널의 출력 상태를 ON 으로 설정하고 출력 상태를 확인 합니다.
if ( cemDoOne_Put ( nMdoChNo, CE_TRUE ) == ceERR_NONE )
{
    cemDoOne_Get ( nMdoChNo, &nMdoState );
    if ( nMdoState != CE_TRUE )
    {
            OutputDebugString ( “cemDoOne_Put has been failed” )
    }
}

// MDI 1 번 채널의 입력 상태를 확인합니다.
if ( cemDiOne_Get ( nMdiChNo, &nMdiState ) == ceERR_NONE )
{
    if ( nMdiState != nMdoState )
    {
            OutputDebugString ( "cemDiOne_Get has been failed" );
    }
}
```

Visual Basic

```
‘ MDO 1 번 채널의 출력을 제어하고, MDO 1 번 채널의 출력을 MDI 1 번 채널을 통해 입력 받아 확인합니다.

Dim nMdiChNo As Long                        ‘ MDI 채널 번호 설정
Dim nMdoChNo As Long                        ‘ MDO 채널 번호 설정
Dim nMdiState As Long, nMdoState As Long    ‘ MDIO 상태 정보

nMdiChNo = 1
nMdoChNo = 1

‘ MDO 1 번 채널의 출력 상태를 ON 으로 설정하고 출력 상태를 확인 합니다.
If cemDoOne_Put ( nMdoChNo, CE_TRUE ) = ceERR_NONE Then

    Call cemDoOne_Get ( nMdoChNo, nMdoState )
    If nMdoState <> CE_TRUE Then
            MsgBox ( “cemDoOne_Put has been failed” )
    End If
End If

‘ MDI 1 번 채널의 입력 상태를 확인합니다.
```

```
If cemDiOne_Get ( nMdiChNo, nMdiState ) = ceERR_NONE Then

    If nMdiState <> nMdoState Then
            MsgBox ( "cemDiOne_Get has been failed" )
    End If
End If
```

Delphi

```
// MDO 1 번 채널의 출력을 제어하고, MDO 1 번 채널의 출력을 MDI 1 번 채널을 통해 입력 받아 확인합니다.

var
    nMdiChNo : LongInt;                     // MDI 채널 번호 설정
    nMdoChNo : LongInt;                     // MDO 채널 번호 설정
    nMdiState, nMdoState : LongInt;         // MDIO 상태 정보

begin
    nMdiChNo := 1;
    nMdoChNo := 1;

    // MDO 1 번 채널의 출력 상태를 ON 으로 설정하고 출력 상태를 확인 합니다.
    if cemDoOne_Put ( nMdoChNo, CE_TRUE ) = ceERR_NONE then
    begin
            cemDoOne_Get ( nMdoChNo, @nMdoState );

            if nMdoState <> CE_TRUE then
            begin
                    ShowMessage ( 'cemDoOne_Put has been failed' );
            end;
    end;

    // MDI 1 번 채널의 입력 상태를 확인합니다.
    if cemDiOne_Get ( nMdiChNo, @nMdiState ) = ceERR_NONE then
    begin
            if nMdiState <> nMdoState then
                    ShowMessage ( 'cemDiOne_Get has been failed' );
    end;
end;
```

NAME

cemDiMulti_Get

- 모션 디지털 입력이 지원되는 제품에서 다중 채널에 대해 입력 상태 반환

INFORMATION	
1	Motion DIO Control
!	VC++ (6, 7, 8)/VB
	BCB/Delphi
:	Level 1
J	위험 요소 없음

SYNOPSIS

r VT_I4 cemDiMulti_Get

([in] VT_I4 IniChannel, [in] VT_I4 NumChannels, [out] VT_PI4 InputState)

DESCRIPTION

모션 디지털 입력이 지원되는 제품에서 다중 채널에 대해 디지털 입력 상태를 반환합니다.

PARAMETER

▶ *IniChannel* : 시작 채널 번호. 통합 채널로 관리되는 채널 번호를 의미하며, 상수 값으로 0 (Zero Based) 이상, (최대 통합 채널수 - 1) 이하의 값을 채널 번호로 설정할 수 있습니다.

▶ *NumChannels* : 시작 채널로부터 몇 개의 채널의 상태를 확인할 것인지에 대한 값을 전달합니다. (최대 32 개 채널까지 설정 가능합니다.)

▶ *InputState* : 이 매개변수를 통하여 다중 디지털 입력 채널의 입력 상태를 반환합니다. (32 비트, BIT0 ~ BIT31).

Value	Meaning
0 (CE_FALSE)	OFF
1 (CE_TRUE)	ON

RETURN VALUE

Value	Meaning
음수	수행 실패. 자세한 내용은 '에러처리' 편을 참고합니다.
0 (ceERR_NONE)	수행 성공.

EXAMPLE

C/C++

```
#include “ceSDK.h”
#include “ceSDKDef.h”

/* MDO 0~3 번 채널의 출력을 제어하고,
MDO 0~3 번 채널의 출력 상태를 MDI 0~3 번 채널을 통해 입력 받아 확인합니다. */

long nMdiIniChNo = 0;                   // MDI 시작 채널 번호 설정
long nMdiChNum = 4;                     // MDI 채널 개수 설정
long nMdoIniChNo = 0;                   // MDO 시작 채널 번호 설정
long nMdoChNum = 4;                     // MDO 채널 개수 설정
long dwMdiStates, dwMdoStates;          // MDIO 상태 정보

// MDO 0~3 번 채널의 출력 상태를 0, 1, 3 ch : ON, 2 ch : OFF 로 설정하고 출력 상태를 확인 합니다.
if ( cemDoMulti_Put ( nMdoIniChNo, nMdoChNum, 0xB ) == ceERR_NONE )
{
    cemDoMulti_Get ( nMdoIniChNo, nMdoChNum, &nMdoStates );
    if ( nMdoStates != 0xB )
    {
            OutputDebugString ( “cemDoMulti_Put has been failed” )
    }
}

// MDI 0~3 번 채널의 입력 상태를 확인합니다.
if ( cemDiMulti_Get ( nMdiIniChNo, nMdiChNum, &nMdiStates ) == ceERR_NONE )
{
    if ( nMdiStates != nMdoStates )
    {
            OutputDebugString ( "cemDiMulti_Get has been failed" );
    }
}
```

Visual Basic

```
‘ MDO 0~3 번 채널의 출력을 제어하고,
‘ MDO 0~3 번 채널의 출력 상태를 MDI 0~3 번 채널을 통해 입력 받아 확인합니다.

Dim nMdiIniChNo As Long                         ‘ MDI 시작 채널 번호 설정
Dim nMdiChNum As Long                           ‘ MDI 채널 개수 설정
Dim nMdoIniChNo As Long                         ‘ MDO 시작 채널 번호 설정
Dim nMdoChNum As Long                           ‘ MDO 채널 개수 설정
Dim dwMdiStates As Long, dwMdoStates As Long    ‘ MDIO 상태 정보

nMdiIniChNo = 0
nMdiChNum = 4
nMdoIniChNo = 0
nMdoChNum = 4
```

```
‘MDO 0~3 번 채널의 출력 상태를 0, 1, 3 ch : ON, 2 ch : OFF 로 설정하고 출력 상태를 확인 합니다.
If cemDoMulti_Put ( nMdoIniChNo, nMdoChNum, &HB ) = ceERR_NONE Then

    Call cemDoMulti_Get ( nMdoIniChNo, nMdoChNum, nMdoStates )
    If nMdoStates <> &HB Then
            MsgBox ( “cemDoMulti_Put has been failed” )
    End If
End If

‘ MDI 0~3 번 채널의 입력 상태를 확인합니다.
If cemDiMulti_Get ( nMdiIniChNo, nMdiChNum, nMdiStates ) = ceERR_NONE Then

    If nMdiStates <> nMdoStates Then
            MsgBox ( "cemDiMulti_Get has been failed" )
    End If
End If
```

Delphi

```
// MDO 0~3 번 채널의 출력을 제어하고,
// MDO 0~3 번 채널의 출력 상태를 MDI 0~3 번 채널을 통해 입력 받아 확인합니다.

var
    nMdiIniChNo, nMdiChNum, nMdoIniChNo, nMdoChNum : LongInt;       // MDIO 채널 설정
    dwMdiStates, dwMdoStates : LongInt                              // MDIO 상태 정보

begin
    nMdiIniChNo := 0;               // MDI 시작 채널 번호 설정
    nMdiChNum := 4;                 // MDI 채널 개수 설정
    nMdoIniChNo := 0;               // MDO 시작 채널 번호 설정
    nMdoChNum := 4;                 // MDO 채널 개수 설정

    // MDO 0~3 번 채널의 출력 상태를 0, 1, 3 ch : ON, 2 ch : OFF 로 설정하고 출력 상태를 확인 합니다.
    if cemDoMulti_Put ( nMdoIniChNo, nMdoChNum, $B ) = ceERR_NONE then
    begin
            cemDoMulti_Get ( nMdoIniChNo, nMdoChNum, @nMdoStates );

            If nMdoStates <> $B then
                    ShowMessage ( ‘cemDoMulti_Put has been failed’ );
            end;
    end;

    // MDI 0~3 번 채널의 입력 상태를 확인합니다.
    if cemDiMulti_Get ( nMdiIniChNo, nMdiChNum, @nMdiStates ) = ceERR_NONE then
    begin
            if nMdiStates <> nMdoStates then
            begin
                    ShowMessage ( ‘cemDiMulti_Get has been failed’ );
            end;
    end;
end;
```

NAME

cemDoOne_Put / cemDoOne_Get

- 모션 디지털 출력이 지원되는 제품에서 단일 채널에 대해 출력 발생 및 출력 상태 확인

INFORMATION	
1	Motion DIO Control
!	VC++ (6, 7, 8)/VB
	BCB/Delphi
:	Level 1
J	위험 요소 없음

SYNOPSIS

r VT_I4 cemDoOne_Put ([in] VT_I4 Channel, [in] VT_I4 OutState)

r VT_I4 cemDoOne_Get ([in] VT_I4 Channel, [out] VT_PI4 OutState)

DESCRIPTION

모션 디지털 출력이 지원되는 제품에서 cemDoOne_Put 함수는 단일 채널에 대한 디지털 출력 상태를 발생시키며, cemDoOne_Ge 함수는 단일 채널에 대한 디지털 출력 상태를 반환합니다.

PARAMETER

▶ ***Channel*** **:** 채널 번호. 통합 채널로 관리되는 채널 번호를 의미하며, 상수 값으로 0 (Zero Based) 이상, (최대 통합 채널수 - 1) 이하의 값을 채널 번호로 설정할 수 있습니다.

▶ ***OutState*** **:** 단일 채널에 대한 디지털 출력 상태를 발생 혹은 출력 상태를 반환합니다

Value	Meaning
0 (CE_FALSE)	OFF
1 (CE_TRUE)	ON

RETURN VALUE

Value	Meaning
음수	수행 실패. 자세한 내용은 '에러처리' 편을 참고합니다.
0 (ceERR_NONE)	수행 성공.

EXAMPLE

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

NAME

cemDoMulti_Put / cemDoMulti_Get

- 모션 디지털 출력이 지원되는 제품에서 다중 디지털 출력 채널의 출력 발생 및 출력 상태 확인

INFORMATION	
1	Motion DIO Control
!	VC++ (6, 7, 8)/VB
	BCB/Delphi
:	Level 1
J	위험 요소 없음

SYNOPSIS

r VT_I4 cemDoMulti_Put

([in] VT_I4 IniChannel, [in] VT_I4 NumChannels, [in] VT_I4 OutStates)

r VT_I4 cemDoMulti_Get

([in] VT_I4 IniChannel, [in] VT_I4 NumChannels, [out]VT_PI4 OutStates)

DESCRIPTION

모션 디지털 출력이 지원되는 제품에서 cemDoMulti_Put 함수는 다중 채널 대한 디지털 출력 상태를 발생시키며, cemDoMulti_Get 함수는 다중 채널에 대한 디지털 출력 상태를 반환합니다.

PARAMETER

▶ ***IniChannel*** : 시작 채널 번호. 통합 채널로 관리되는 채널 번호를 의미하며, 상수 값으로 0 (Zero Based) 이상, (최대 통합 채널수 - 1) 이하의 값을 채널 번호로 설정할 수 있습니다.

▶ ***NumChannels*** : 시작 채널로부터 몇 개의 채널의 상태를 확인할 것인지에 대한 값을 전달합니다. (최대 32 개 채널까지 설정 가능합니다.)

▶ ***OutStates*** : 다중 채널에 대한 출력 상태를 발생 혹은 출력 상태를 반환합니다. 이 출력 상태는 BitMask(비트 마스크) 로 설정되며, 설정된 비트가 1 일 경우 디지털 출력이 발생합니다.

Value	Meaning
0 (CE_FALSE)	OFF
1 (CE_TRUE)	ON

RETURN VALUE

Value	Meaning
음수	수행 실패. 자세한 내용은 '에러처리' 편을 참고합니다.
0 (ceERR_NONE)	수행 성공.

EXAMPLE

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

Chapter

12

Universal Digital I/O Control

범용 디지털 입·출력 기능에 대해서 ceSDK 는 매우 유연한 입·출력 기능을 제공하고 있습니다. 특히 S/W 노이즈 필터 기능은 보다 안정적인 디지털 입력 신호 처리를 이유로 사용될 수 있으며, 결과적으로 시스템의 전체적인 신뢰성을 동시에 향상시키는 효과를 경험하실 수 있습니다.

시스템에서 범용 디지털 입·출력은 주요 외부 입·출력 인터페이스(Interface)의 하나로서, 소위 디지털 접점(Digital Contact) 이라는 말로 일컫습니다. 이 장에서는 편리한 ceSDK 의 디지털 입·출력 기능들을 자세히 살펴보고, 활용할 수 있도록 구성하였습니다.

12 범용 디지털 입·출력 편

이 단원에서는 범용 디지털 입력/출력 모듈에 관련된 함수를 소개합니다. cEIP 디지털 입·출력 모듈은 3 가지로 구분되며 각각의 특징 및 사용 가능 함수는 다음과 같습니다.

가. cEIP 디지털 입·출력 모듈 특징

모듈 명	디지털 입력/출력 구성	채널 수
ceD16CM	디지털 입력 / 출력 채널 설정	16 Channel Ditital Input/Output
ceDI32N	디지털 입력 채널	32 Channel Digital Input
ceDO32N	디지털 출력 채널	32 Channel Digital Output

나. 범용 디지털 입·출력 함수 지원 모듈

함수 명	사용 가능 모듈
cedioMode_Set	ceD16CM
cedioMode_Get	
cedioModeMulti_Set	
cedioModeMulti_Get	
cedioLogicOne_Set	ceD16CM ceDI32N ceDO32N
cedioLogicOne_Get	
cedioLogicMulti_Set	
cedioLogicMulti_Get	
cedioOne_Get	
cedioOne_Put	
cedioMulti_Get	
cedioMulti_Put	
cedioOneF_Get	
cedioMultiF_Get	
cedioPulseOne	ceD16CM (디지털 출력 채널 대상) ceDO32N
cedioPulseMulti	

디지털 입력 채널과 출력 채널 번호의 관계는 다음과 같습니다.

디지털 입력 채널과 디지털 출력 채널은 서로 구분되지 않으며 통합하여 채널 번호가 부여됩니다.
원격 노드의 번호, 노드 내의 모션 모듈 번호를 기준으로 디지털 입력 채널과 디지털 출력 채널 번호가 증가됩니다.

12.1 함수 요약

범용 디지털 입·출력 기능과 관련된 함수 는 다음의 표와 같습니다.

Summary of Functions
r VT_I4 cedioMode_Set ([in] VT_I4 Channel, [in] VT_I4 InOutMode) 대상 디지털 I/O 채널의 입·출력 모드(Mode)를 설정합니다.
r VT_I4 cedioMode_Get ([in] VT_I4 Channel, [out] VT_PI4 InOutMode) 대상 디지털 I/O 채널의 입·출력 모드 설정 상태를 반환합니다.
r VT_I4 cedioModeMulti_Set ([in] VT_I4 IniChan, [in] VT_I4 NumChan, [in] VT_I4 InOutModeMask) 다중 디지털 I/O 채널의 입·출력 모드를 설정합니다.
r VT_I4 cedioModeMulti_Get ([in] VT_I4 IniChan, [in] VT_I4 NumChan, [out] VT_PI4 InOutModeMask) 다중 디지털 I/O 채널의 입·출력 모드 설정 상태를 반환합니다.
r VT_I4 cedioLogicOne_Set ([in] VT_I4 Channel, [in] VT_I4 Logic) 대상 디지털 I/O 채널의 로직(Logic)을 설정합니다.
r VT_I4 cedioLogicOne_Get ([in] VT_I4 Channel, [out] VT_PI4 Logic) 대상 디지털 I/O 채널의 로직 설정 상태를 반환합니다.
r VT_I4 cedioLogicMulti_Set ([in] VT_I4 IniChan, [in] VT_I4 NumChan, [in] VT_I4 LogicMask) 다중 디지털 I/O 채널의 로직을 설정합니다.
r VT_I4 cedioLogicMulti_Get ([in] VT_I4 IniChan, [in] VT_I4 NumChan, [out] VT_PI4 LogicMask) 다중 디지털 I/O 채널의 로직 설정 상태를 반환합니다.
r VT_I4 cedioOne_Get ([in] VT_I4 Channel, [out] VT_PI4 State) 대상 디지털 I/O 채널의 디지털 입력 또는 출력 상태를 반환합니다.
r VT_I4 cedioOne_Put ([in] VT_I4 Channel, [in] VT_I4 State) 대상 디지털 I/O 채널을 통해 디지털 입력 또는 출력 상태를 설정합니다.
r VT_I4 cedioMulti_Get ([in] VT_I4 IniChan, [in] VT_I4 NumChan, [out] VT_PI4 States) 다중 디지털 I/O 채널의 디지털 입력 또는 출력 상태를 반환합니다.
r VT_I4 cedioMulti_Put ([in] VT_I4 IniChan, [in] VT_I4 NumChan, [in] VT_I4 States) 다중 디지털 출력 채널을 통해 디지털 입력 또는 출력 상태를 설정합니다.
r VT_I4 cedioOneF_Get ([in] VT_I4 Channel, [in] VT_I4 CutoffTime_us, [out] VT_PI4 State) 대상 디지털 I/O 채널의 노이즈 필터 기능을 활성화 하여 디지털 입력 또는 출력 상태를 반환합니다.
r VT_I4 cedioMultiF_Get ([in] VT_I4 IniChan, [in] VT_I4 NumChan, [in] VT_I4 CutoffTime_us, [out] VT_PI4 States) 다중 디지털 I/O 채널의 노이즈 필터 기능을 활성화 하여 디지털 입력 또는 출력 상태를 반환합니다.
r VT_I4 cedioPulseOne ([in] VT_I4 Channel, [in] VT_I4 IsOnPulse, [in] VT_I4 Duration, [in] VT_I4 IsWaitPulseEnd) 대상 디지털 출력 채널을 통해 지정한 시간 동안 단일 펄스 출력을 발생시킵니다.
r VT_I4 cedioPulseMulti ([in] VT_I4 IniChan, [in] VT_I4 NumChan, [in] VT_I4 OutStates, [in] VT_I4 Duration, [in] VT_I4 IsWaitPulseEnd) 다중 디지털 출력 채널을 통해 지정한 시간 동안 단일 펄스 출력을 발생시킵니다.

12.2 함수 설명

NAME

cedioMode_Set / cedioMode_Get

- 대상 디지털 채널의 입·출력 모드(Mode) 설정 및 설정 상태 반환

INFORMATION	
1	Universal DIO Control
!	VC++ (6, 7, 8)/VB
	BCB/Delphi
:	Level 1
J	위험 요소 없음

SYNOPSIS

r VT_I4 cedioMode_Set ([in] VT_I4 Channel, [in] VT_I4 InOutMode)

r VT_I4 cedioMode_Get ([in] VT_I4 Channel, [out] VT_PI4 InOutMode)

DESCRIPTION

cedioMode_Set/cedioMode_Get 함수는 지정 채널의 디지털 입·출력 모드(Mode)를 설정 혹은 설정 상태를 반환 합니다.

PARAMETER

▶ ***Channel*** **:** 채널 번호. 통합 채널로 관리되는 채널 번호를 의미하며, 상수 값으로 0 (Zero Based) 이상, (최대 통합 채널수 - 1) 이하의 값을 채널 번호로 설정할 수 있습니다.

▶ ***InOutMode*** **:** 디지털 입·출력 채널의 모드를 설정 혹은 설정상태를 반환합니다.

Value	Meaning
0 (CE_FALSE)	Input Mode.
1 (CE_TRUE)	Output Mode.

RETURN VALUE

Value	Meaning
음수	수행 실패. 자세한 내용은 '에러처리' 편을 참고합니다.
0 (ceERR_NONE)	수행 성공.

REFERENCE

본 함수는 ceD16CM 모듈 전용 함수입니다.

EXAMPLE

C/C++

```
#include “ceSDK.h”
#include “ceSDKDef.h”

long nDioChNo = 1;              // 채널 번호 설정
long nDioMode;                  // 입/출력 채널 모드 설정 정보

// ceD16CM 모듈의 채널에 대해 입/출력 모드를 확인 후, 출력 모드로 설정합니다.
if ( cedioMode_Get ( nDioChNo, &nDioMode ) == ceERR_NONE )
{
    if ( nDioMode != CE_TRUE )
    {
            cedioMode_Set ( nDioChNo, CE_TRUE );
    }
}
```

Visual Basic

```
Dim nDioChNo As Long            ‘ 채널 번호 설정
Dim nDioMode As Long            ‘ 입/출력 채널 모드 설정 정보

nDioChNo = 1

‘ceD16CM 모듈의 채널에 대해 입/출력 모드를 확인 후, 출력 모드로 설정합니다.
If cedioMode_Get ( nDioChNo, nDioMode ) = ceERR_NONE Then

    If nDioMode <> CE_TRUE Then
            Call cedioMode_Set ( nDioChNo, CE_TRUE )
    End If
End If
```

Delphi

```
var
    nDioChNo : LongInt;         // 채널 번호 설정
    nDioMode : LongInt;         // 입/출력 채널 모드 설정 정보

begin
    nDioChNo := 1;

    // ceD16CM 모듈의 채널에 대해 입/출력 모드를 확인 후, 출력 모드로 설정합니다.
    if cedioMode_Get ( nDioChNo, @nDioMode ) = ceERR_NONE then
    begin
            if nDioMode <> CE_TRUE then
            begin
                    cedioMode_Set ( nDioChNo, CE_TRUE );
            end;
    end;
end;
```

NAME

cedioModeMulti_Set / cedioModeMulti_Get

- 다중(Multi) 디지털 채널의 입·출력 모드(Mode) 설정 및 설정 상태 반환

INFORMATION	
1	Universal DIO Control
!	VC++ (6, 7, 8)/VB
	BCB/Delphi
:	Level 1
J	위험 요소 없음

SYNOPSIS

r VT_I4 cedioModeMulti_Set ([in] VT_I4 IniChan, [in] VT_I4 NumChan, [in] VT_I4 InOutModeMask)

r VT_I4 cedioModeMulti_Get ([in] VT_I4 IniChan, [in] VT_I4 NumChan, [out] VT_PI4 InOutModeMask)

DESCRIPTION

cedioModeMulti_Set/cedioModeMulti_Get 함수는 다중(Multi) 디지털 채널의 입·출력 모드(Mode)를 설정 혹은 설정 상태를 반환합니다.

PARAMETER

▶ ***IniChan*** **:** 시작 채널 번호. 통합 채널로 관리되는 채널 번호를 의미하며, 상수 값으로 0 (Zero Based) 이상, (최대 통합 채널수 - 1) 이하의 값을 채널 번호로 설정할 수 있습니다.

▶ ***NumChan*** **:** 시작 채널로부터 몇 개의 채널의 상태를 확인할 것인지에 대한 값을 전달합니다. (최대 32 개 채널까지 설정 가능합니다.)

▶ ***InOutModeMask*** **:** 이 매개변수를 통하여 다중(Multi) 디지털 I/O 채널의 용도(Mode) 설정 및 설정상태를 반환합니다. (32 비트, BIT0 ~ BIT31)

Value	Meaning
0 (CE_FALSE)	Input Mode.
1 (CE_TRUE)	Output Mode.

RETURN VALUE

Value	Meaning
음수	수행 실패. 자세한 내용은 '에러처리' 편을 참고합니다.
0 (ceERR_NONE)	수행 성공.

REFERENCE

본 함수는 ceD16CM 모듈 전용 함수입니다.

EXAMPLE

C/C++

```
#include “ceSDK.h”
#include “ceSDKDef.h”

long nDioIniChNo = 0;                    // 시작 채널 번호 설정
long nDioChNum = 4;                      // 채널 개수 설정
long nDioModeMulti;                      // 입/출력 채널 모드 설정 정보

/* ceD16CM 모듈의 0~3 번 채널의 입/출력 모드를 확인 후,
0, 1, 3 ch 출력 모드, 2 ch 입력 모드로 설정합니다. */

if ( cedioModeMulti_Get ( nDioIniChNo, nDioChNum, &nDioModeMulti ) == ceERR_NONE )
{
    if ( nDioModeMulti != 0xB )
    {
            cedioModeMulti_Set ( nDioIniChNo, nDioChNum, 0xB );
    }
}
```

Visual Basic

```
Dim nDioIniChNo As Long          ‘ 시작 채널 번호 설정
Dim nDioChNum As Long            ‘ 채널 개수 설정
Dim nDioModeMulti As Long        ‘ 입/출력 채널 모드 설정 정보

nDioIniChNo = 0
nDioChNum = 4

‘ ceD16CM 모듈의 0~3 번 채널의 입/출력 모드를 확인 후,
‘ 0, 1, 3 ch 출력 모드, 2 ch 입력 모드로 설정합니다.

If cedioModeMulti_Get ( nDioIniChNo, nDioChNum, nDioModeMulti ) = ceERR_NONE Then

    If nDioModeMulti <> &HB Then
            Call cedioModeMulti_Set ( nDioIniChNo, nDioChNum, &HB )
    End If
End If
```

Delphi

```
var
```

```
    nDioIniChNo : LongInt;              // 시작 채널 번호 설정
    nDioChNum : LongInt;                // 채널 개수 설정
    nDioModeMulti : LongInt;            // 입/출력 채널 모드 설정 정보

begin
    nDioIniChNo := 0;
    nDioChNum := 4;

    { ceD16CM 모듈의 0~3 번 채널의 입/출력 모드를 확인 후,
     0, 1, 3 ch 출력 모드, 2 ch 입력 모드로 설정합니다. }

    if cedioModeMulti_Get ( nDioIniChNo, nDioChNum, @nDioModeMulti ) = ceERR_NONE then
    begin
            if nDioModeMulti <> $B then
            begin
                    cedioModeMulti_Set ( nDioIniChNo, nDioChNum, $B );
            end;
    end;

end;
```

NAME

cedioLogicOne_Set / cedioLogicOne_Get

- 대상 디지털 입·출력 채널의 로직(Logic) 설정 및 설정 상태 반환

INFORMATION	
1	Universal DIO Control
!	VC++ (6, 7, 8)/VB
	BCB/Delphi
:	Level 1
J	위험 요소 없음

SYNOPSIS

r VT_I4 cedioLogicOne_Set ([in] VT_I4 Channel, [in] VT_I4 Logic)

r VT_I4 cedioLogicOne_Get ([in] VT_I4 Channel, [out] VT_PI4 Logic)

DESCRIPTION

cedioLogicOne_Set/cedioLogicOne_Get 함수는 대상 디지털 입·출력 채널의 로직(Logic) 설정 혹은 설정상태를 반환합니다.

PARAMETER

▶ ***Channel*** **:** 채널 번호. 통합 채널로 관리되는 채널 번호를 의미하며, 상수 값으로 0 (Zero Based) 이상, (최대 통합 채널수 - 1) 이하의 값을 채널 번호로 설정할 수 있습니다.

▶ ***Logic*** **:** 대상 디지털 I/O 채널의 로직 설정 혹은 설정상태를 반환합니다.

Value	Meaning
0 (cemLOGIC_A)	A 접점 방식 => 평상시 Open, 감지되면 Close 되는 스위치 방식.
1 (cemLOGIC_B)	B 접점 방식 => 평상시 Close, 감지되면 Open 되는 스위치 방식.

RETURN VALUE

Value	Meaning
음수	수행 실패. 자세한 내용은 '에러처리' 편을 참고합니다.
0 (ceERR_NONE)	수행 성공.

EXAMPLE

C/C++

```
#include “ceSDK.h”
#include “ceSDKDef.h”

long nDioChNo = 1;              // 채널 번호 설정
long nDioLogic;                 // 입/출력 채널 로직 설정 정보

// ceD16CM 모듈의 채널에 대해 입/출력 로직을 확인 후, B 접점으로 설정합니다.
if ( cedioLogicOne_Get ( nDioChNo, &nDioLogic ) == ceERR_NONE )
{
    if ( nDioLogic != cemLogic_B )
    {
            cedioLogicOne_Set ( nDioChNo, cemLogic_B );
    }
}
```

Visual Basic

```
Dim nDioChNo As Long            ‘ 채널 번호 설정
Dim nDioLogic As Long           ‘ 입/출력 채널 모드 설정 정보

nDioChNo = 1

‘ceD16CM 모듈의 채널에 대해 입/출력 로직을 확인 후, B 접점으로 설정합니다.
If cedioLogicOne_Get ( nDioChNo, nDioLogic ) = ceERR_NONE Then

    If nDioLogic <> cemLogic_B Then
            Call cedioMode_Set ( nDioChNo, cemLogic_B )
    End If
End If
```

Delphi

```
var
    nDioChNo : LongInt;         // 채널 번호 설정
    nDioLogic : LongInt;        // 입/출력 채널 모드 설정 정보

begin
    nDioChNo := 1;

    // ceD16CM 모듈의 채널에 대해 입/출력 로직을 확인 후, B 접점으로 설정합니다.
    if cedioLogicOne_Get ( nDioChNo, @nDioLogic ) = ceERR_NONE then
    begin
            if nDioLogic <> cemLogic_B then
            begin
                    cedioMode_Set ( nDioChNo, cemLogic_B );
            end;
    end;
end;
```

NAME

cedioLogicMulti_Set / cedioLogicMulti_Get

- 다중(Multi) 디지털 입·출력 채널의 로직(Logic) 설정 및 설정 상태 반환

INFORMATION	
1	Universal DIO Control
!	VC++ (6, 7, 8)/VB
	BCB/Delphi
:	Level 1
J	위험 요소 없음

SYNOPSIS

r VT_I4 cedioLogicMulti_Set ([in] VT_I4 IniChan, [in] VT_I4 NumChan, [in] VT_I4 LogicMask)

r VT_I4 cedioLogicMulti_Get ([in] VT_I4 IniChan, [in] VT_I4 NumChan, [out] VT_PI4 LogicMask)

DESCRIPTION

cedioLogicMulti_Set/cedioLogicMulti_Get 함수는 다중(Multi) 디지털 입·출력 채널의 로직(Logic) 설정 및 설정상태를 반환합니다.

PARAMETER

▶ ***IniChan*** : 시작 채널 번호. 통합 채널로 관리되는 채널 번호를 의미하며, 상수 값으로 0 (Zero Based) 이상, (최대 통합 채널수 – 1) 이하의 값을 채널 번호로 설정할 수 있습니다.

▶ ***NumChan*** : 시작 채널로부터 몇 개의 채널의 상태를 확인할 것인지에 대한 값을 전달합니다. (최대 32 개 채널까지 설정 가능합니다.)

▶ ***LogicMask*** : 이 매개변수를 통하여 다중(Multi) 디지털 I/O 채널의 로직(Logic) 설정 및 설정 상태를 반환합니다. (32 비트, BIT0 ~ BIT31)

Value	Meaning
0 (cemLOGIC_A)	A 접점 방식 => 평상시 Open, 감지되면 Close 되는 스위치 방식.
1 (cemLOGIC_B)	B 접점 방식 => 평상시 Close, 감지되면 Open 되는 스위치 방식.

RETURN VALUE

Value	Meaning
음수	수행 실패. 자세한 내용은 '에러처리' 편을 참고합니다.
0 (ceERR_NONE)	수행 성공.

EXAMPLE

C/C++

```
#include “ceSDK.h”
#include “ceSDKDef.h”

long nDioIniChNo = 0;                    // 시작 채널 번호 설정
long nDioChNum = 4;                      // 채널 개수 설정
long nDioLogicMulti;                     // 입/출력 채널 로직 설정 정보

/* 0~3 번 채널의 입ㆍ출력 로직을 확인 후,
0, 1, 3 ch 은 A 접점, 2 ch 은 B 접점 로직으로 설정합니다. */

if ( cedioLogicMulti_Get ( nDioIniChNo, nDioChNum, &nDioLogicMulti ) == ceERR_NONE )
{
    if ( nDioLogicMulti != 0x2 )
    {
            cedioModeMulti_Set ( nDioIniChNo, nDioChNum, 0x2 );
    }
}
```

Visual Basic

```
Dim nDioIniChNo As Long              ‘ 시작 채널 번호 설정
Dim nDioChNum As Long                ‘ 채널 개수 설정
Dim nDioLogicMulti As Long           ‘ 입/출력 채널 로직 설정 정보

nDioIniChNo = 0
nDioChNum = 4

‘ 0~3 번 채널의 입ㆍ출력 로직을 확인 후,
‘ 0, 1, 3 ch 은 A 접점, 2 ch 은 B 접점 로직으로 설정합니다.

If cedioModeMulti_Get ( nDioIniChNo, nDioChNum, nDioLogicMulti ) = ceERR_NONE Then

    If nDioLogicMulti <> &H2 Then
            Call cedioModeMulti_Set ( nDioIniChNo, nDioChNum, &H2 )
    End If
End If
```

Delphi

```
var
    nDioIniChNo : LongInt;           // 시작 채널 번호 설정
    nDioChNum : LongInt;             // 채널 개수 설정
    nDioLogicMulti : LongInt ;       // 입/출력 채널 로직 설정 정보

begin
    nDioIniChNo := 0;
    nDioChNum := 4;
```

```
    { 0~3 번 채널의 입ㆍ출력 로직을 확인 후,
    0, 1, 3 ch 은 A 접점, 2 ch 은 B 접점 로직으로 설정합니다. }

    if cedioModeMulti_Get ( nDioIniChNo, nDioChNum, @nDioLogicMulti ) = ceERR_NONE then
    begin
            if nDioLogicMulti <> $2 then
            begin
                    cedioModeMulti_Set ( nDioIniChNo, nDioChNum, $2 );
            end;
    end;
end;
```

NAME

cedioOne_Get / cedioOne_Put

대상 디지털 입·출력 채널의 상태 반환 및 설정

INFORMATION	
1	Universal DIO Control
!	VC++ (6, 7, 8)/VB
	BCB/Delphi
:	Level 1
J	위험 요소 없음

SYNOPSIS

r VT_I4 cedioOne_Get ([in] VT_I4 Channel, [out] VT_PI4 State)

r VT_I4 cedioOne_Put ([in] VT_I4 Channel, [in] VT_I4 State)

DESCRIPTION

cedioOne_Get 함수는 대상 디지털 채널의 입·출력 모드에 따라 입력 또는 출력 상태를 반환합니다.

cedioOne_Put 함수는 대상 디지털 채널의 입·출력 모드에 따라 입력 또는 출력 상태를 설정합니다.

PARAMETER

▶ ***Channel*** **:** 채널 번호. 통합 채널로 관리되는 채널 번호를 의미하며, 상수 값으로 0 (Zero Based) 이상, (최대 통합 채널수 - 1) 이하의 값을 채널 번호로 설정할 수 있습니다.

▶ ***State*** **:** 대상 채널의 입·출력 모드에 따라 cedioOne_Get 함수에서는 입력 또는 출력 상태를 반환하며, cedioOne_Put 함수에서는 입력 또는 출력 상태를 설정합니다.

Value	Meaning
0 (CE_FALSE)	OFF
1 (CE_TRUE)	ON

RETURN VALUE

Value	Meaning
음수	수행 실패. 자세한 내용은 '에러처리' 편을 참고합니다.
0 (ceERR_NONE)	수행 성공.

EXAMPLE

C/C++

```
#include “ceSDK.h”
#include “ceSDKDef.h”

long nDoChNo = 1;              // 채널 번호 설정, 출력 채널이라 가정합니다.
long nDoState;                 // 채널 상태 정보

// 해당 채널의 출력 상태를 ON 으로 설정하고, 출력 상태를 확인합니다.
if ( cedioOne_Put ( nDoChNo, CE_TRUE ) == ceERR_NONE )
{
    cedioOne_Get ( nDoChNo, &nDoState );

    if ( nDoState != CE_TRUE )
    {
            OutputDebugString ( "cedioOne_Get has been failed" );
    }
}
```

Visual Basic

```
Dim nDoChNo As Long        ‘ 채널 번호 설정, 출력 채널이라 가정합니다.
Dim nDoState As Long       ‘ 채널 상태 정보

nDoChNo = 1

‘ 해당 채널의 출력 상태를 ON 으로 설정하고, 출력 상태를 확인합니다.
If cedioOne_Put ( nDoChNo, CE_TRUE ) = ceERR_NONE Then

    Call cedioOne_Get ( nDoChNo, nDoState )

    If nDoState <> CE_TRUE Then
            MsgBox ( "cedioOne_Get has been failed" )
    End If
End If
```

Delphi

```
var
    nDoChNo : LongInt          // 채널 번호 설정, 출력 채널이라 가정합니다.
    nDoState : LongInt         // 채널 상태 정보

begin
    nDoChNo := 1;

    // 해당 채널의 출력 상태를 ON 으로 설정하고, 출력 상태를 확인합니다.
    if cedioOne_Put ( nDoChNo, CE_TRUE ) = ceERR_NONE then
    begin
            cedioOne_Get ( nDoChNo, @nDoState );
```

```
            if nDoState <> CE_TRUE then
            begin
                ShowMessage ( 'cedioOne_Get has been failed' );
            end;
    end;
end;
```

NAME

cedioMulti_Get / cedioMulti_Put

- 다중 디지털 입·출력 채널의 입력/출력 상태 반환 및 설정

INFORMATION	
1	Universal DIO Control
!	VC++ (6, 7, 8)/VB
	BCB/Delphi
:	Level 1
J	위험 요소 없음

SYNOPSIS

r VT_I4 cedioMulti_Get ([in] VT_I4 IniChan, [in] VT_I4 NumChan, [out] VT_PI4 States)

r VT_I4 cedioMulti_Put ([in] VT_I4 IniChan, [in] VT_I4 NumChan, [in] VT_I4 States)

DESCRIPTION

cedioMulti_Get 함수는 다중 디지털 채널의 입·출력 모드에 따라 입력 또는 출력 상태를 반환합니다.

cedioMulti_Put() 함수는 다중 디지털 채널의 입·출력 모드에 따라 입력 또는 출력 상태를 설정합니다.

PARAMETER

▶ ***IniChan*** **:** 시작 채널 번호. 통합 채널로 관리되는 채널 번호를 의미하며, 상수 값으로 0 (Zero Based) 이상, (최대 통합 채널수 - 1) 이하의 값을 채널 번호로 설정할 수 있습니다.

▶ ***NumChan*** **:** 시작 채널로부터 몇 개의 채널의 상태를 확인할 것인지에 대한 값을 전달합니다. (최대 32 개 채널까지 설정 가능합니다.)

▶ ***States*** **:** 대상 채널의 입·출력 모드에 따라 cedioMulti_Get 함수에서는 입력 또는 출력 상태를 반환하며, cedioMulti_Put 함수에서는 입력 또는 출력 상태를 설정합니다.

Value	Meaning
0 (CE_FALSE)	OFF
1 (CE_TRUE)	ON

RETURN VALUE

Value	Meaning
음수	수행 실패. 자세한 내용은 '에러처리' 편을 참고합니다.
0 (ceERR_NONE)	수행 성공.

EXAMPLE

C/C++

```
#include “ceSDK.h”
#include “ceSDKDef.h”

/* 0 ~ 3 번 채널을 출력 채널이라 가정합니다.
1, 3 번 채널의 출력 상태를 ON, 0, 2 번 채널의 출력 상태를 OFF 로 설정 후 확인합니다. */

long nDoIniChNo = 0;          // 시작 채널 번호 설정
long nDoChNum = 4;            // 채널 개수
long nDoStates;               // 채널 상태 정보

// 대상 채널들의 출력 상태를 설정하고, 출력 상태를 확인합니다.
if ( cedioMulti_Put ( nDoIniChNo, nDoChNum, 0xA ) == ceERR_NONE )
{
    cedioMulti_Get ( nDoChNo, &nDoStates );

    if ( nDoStates != 0xA )
    {
            OutputDebugString ( "cedioMulti_Put has been failed" );
    }
}
```

Visual Basic

```
‘ 0 ~ 3 번 채널을 출력 채널이라 가정합니다.
‘ 1, 3 번 채널의 출력 상태를 ON, 0, 2 번 채널의 출력 상태를 OFF 로 설정 후 확인합니다.

Dim nDoIniChNo As Long      ‘ 시작 채널 번호 설정
Dim nDoChNum As Long        ‘ 시작 채널 번호 설정
Dim nDoStates As Long       ‘ 채널 상태 정보

nDoIniChNo = 0
nDoChNum = 4

‘ 대상 채널들의 출력 상태를 설정하고, 출력 상태를 확인합니다.
If cedioMulti_Put ( nDoIniChNo, nDoChNum, &HA ) = ceERR_NONE Then

    Call cedioMulti_Get ( nDoIniChNo, nDoChNum, nDoStates )

    If nDoStates <> &H A Then
            MsgBox ( "cedioMulti_Put has been failed" )
    End If
End If
```

Delphi

```
{ 0 ~ 3 번 채널을 출력 채널이라 가정합니다.
```

```
 1, 3 번 채널의 출력 상태를 ON, 0, 2 번 채널의 출력 상태를 OFF 로 설정 후 확인합니다. }

var
    nDoIniChNo : LongInt;       // 시작 채널 번호 설정
    nDoChNum : LongInt;         // 시작 채널 번호 설정
    nDoStates : LongInt;        // 채널 상태 정보

begin
    nDoIniChNo := 0;
    nDoChNum := 4;

    // 대상 채널들의 출력 상태를 설정하고, 출력 상태를 확인합니다.
    if cedioMulti_Put ( nDoIniChNo, nDoChNum, $A ) = ceERR_NONE then
    begin
            cedioMulti_Get ( nDoIniChNo, nDoChNum, @nDoStates );

            if nDoStates <> $A then
            begin
                    ShowMessage ( 'cedioMulti_Put has been failed' );
            end;
    end;
end;
```

NAME

cedioOneF_Get

- 대상 디지털 입력 채널의 노이즈 필터 기능을 활성화 하여 입력 상태 반환

INFORMATION	
1	Universal DIO Control
!	VC++ (6, 7, 8)/VB
	BCB/Delphi
:	Level 1
J	위험 요소 없음

SYNOPSIS

r VT_I4 cedioOneF_Get ([in] VT_I4 Channel, [in] VT_I4 CutoffTime_us, [out] VT_PI4 State)

DESCRIPTION

단일 채널을 대상으로 노이즈 필터 기능에 대응 되는 디지털 입력 채널 상태를 확인합니다. 이 상태는 cedioLogicOne_Set 함수를 통해 설정된 디지털 입력 로직(Digital Input Logic)이 적용됩니다.

이 함수에서 전달되는 매개변수 중 'CutoffTime_us'는 입력 신호 유지 시간(Signal Width)을 의미합니다.

이 함수를 통해 확인되는 디지털 입력 채널은 입력 신호 유지 시간 이상의 신호가 확인되어야만 합니다. 원하지 않는 신호(Noise)나 일정 시간 이상의 펄스 입력(Pulse Input)을 확인할 때 매우 유용한 상태 확인 함수입니다.

PARAMETER

▶ ***Channel*** **:** 채널 번호. 통합 채널로 관리되는 채널 번호를 의미하며, 상수 값으로 0 (Zero Based) 이상, (최대 통합 채널수 - 1) 이하의 값을 채널 번호로 설정할 수 있습니다.

▶ ***CutoffTime_us*** **:** 디지털 입력 신호 유지 시간을 마이크로 초(us) 단위로 설정합니다.

▶ ***State*** **:** 단일 채널에 대한 디지털 입력 상태를 반환합니다.

Value	Meaning
0 (CE_FALSE)	OFF
1 (CE_TRUE)	ON

RETURN VALUE

Value	Meaning
음수	수행 실패. 자세한 내용은 '에러처리' 편을 참고합니다.
0 (ceERR_NONE)	수행 성공.

EXAMPLE

C/C++

```
#include “ceSDK.h”
#include “ceSDKDef.h”

/* 0 번 채널은 디지털 출력 채널, 1 번 채널은 디지털 입력 채널이라 가정합니다.
0 번 채널을 통해 일정 시간의 Pulse 출력을 내고,
1 번 입력 채널의 노이즈 필터 기능을 통해 입력 상태를 확인합니다. */

long nDoChNo = 0;           // 디지털 출력 채널 번호 설정
long nDiChNo = 1;           // 디지털 입력 채널 번호 설정
long nDiState;              // 디지털 입력 상태 정보
long nRetVal;

// 출력 채널의 로직을 B 접점으로 설정합니다
cedioLogicOne_Set ( nDoChNo, cemLOGIC_B );

// 10usec 동안 단일 펄스 출력을 발생합니다.
nRetVal = cedioPulseOne (   nDoChNo,        // 채널 번호
                            CE_TRUE,        // B 접점일 경우 Active High 상태가 됩니다.
                            10,             // 펄스 출력 시간. 10usec
                            CE_FALSE        // 펄스 출력 후 바로 반환
                            );

if ( nRetVal == ceERR_NONE )
{
    // 입력 채널의 노이즈 필터 기능을 활성화 하고,
    // 20 usec 입력 신호 유지 시간을 설정하여 입력 상태를 확인합니다.
    cedioOneF_Get ( nDiChNo, 20, &nDiState );

    /* 10usec 펄스가 입력되어 노이즈 필터 기능에 의해 20usec 이하의 신호는 ON 신호로 인식하지 않습니다. */

}
```

Visual Basic

```
‘ 0 번 채널은 디지털 출력 채널, 1 번 채널은 디지털 입력 채널이라 가정합니다.
‘ 0 번 채널을 통해 일정 시간의 Pulse 출력을 내고,
‘ 1 번 입력 채널의 노이즈 필터 기능을 통해 입력 상태를 확인합니다.

Dim nDoChNo As Long             ‘ 디지털 출력 채널 번호 설정
Dim nDiChNo As Long             ‘ 디지털 입력 채널 번호 설정
Dim nDiState As Long            ‘ 디지털 입력 상태 정보

nDoChNo = 0
nDiChNo = 1

‘출력 채널의 로직을 B 접점으로 설정합니다.
```

```
Call cedioLogicOne_Set ( nDoChNo, cemLOGIC_B )

‘ 30usec 동안 단일 펄스 출력을 발생합니다.
If cedioPulseOne ( nDoChNo, CE_TRUE, 30, CE_FALSE ) = ceERR_NONE Then

    ‘ 입력 채널의 노이즈 필터 기능을 활성화 하고,
    ‘ 20 usec 입력 신호 유지 시간을 설정하여 입력 상태를 확인합니다.
    Call cedioOneF_Get ( nDiChNo, 20, nDiState )

    ‘ 30usec 펄스가 입력되었고 노이즈 필터는 20usec 로 설정되어 있으므로 ON 신호로 인식합니다.

End If
```

Delphi

```
{ 0 번 채널은 디지털 출력 채널, 1 번 채널은 디지털 입력 채널이라 가정합니다.
 0 번 채널을 통해 일정 시간의 Pulse 출력을 내고,
 1 번 입력 채널의 노이즈 필터 기능을 통해 입력 상태를 확인합니다. }

var
    nDoChNo : LongInt;              // 디지털 출력 채널 번호 설정
    nDiChNo : LongInt;              // 디지털 입력 채널 번호 설정
    nDiState : LongInt;             // 디지털 입력 상태 정보

begin
    nDoChNo := 0;
    nDiChNo := 1;

    // 출력 채널의 로직을 B 접점으로 설정합니다.
    cedioLogicOne_Set ( nDoChNo, cemLOGIC_B );

    // 30usec 동안 단일 펄스 출력을 발생합니다.
    if cedioPulseOne ( nDoChNo, CE_TRUE, 30, CE_FALSE ) = ceERR_NONE then
    begin
            { 입력 채널의 노이즈 필터 기능을 활성화 하고,
             20 usec 입력 신호 유지 시간을 설정하여 입력 상태를 확인합니다. }
            cedioOneF_Get ( nDiChNo, 20, @nDiState ) ;

            // 30usec 펄스가 입력되었고 노이즈 필터는 20usec 로 설정되어 있으므로 ON 신호로 인식합니다.
    end;
end;
```

NAME

cedioMultiF_Get

- 다중 디지털 입력 채널의 노이즈 필터 기능을 활성화 하여 디지털 입력 상태를 반환

INFORMATION	
1	Universal DIO Control
!	VC++ (6, 7, 8)/VB
	BCB/Delphi
:	Level 1
J	위험 요소 없음

SYNOPSIS

r VT_I4 cedioMultiF_Get ([in] VT_I4 IniChan, [in] VT_I4 NumChan, [in] VT_I4 CutoffTime_us, [out] VT_PI4 States)

DESCRIPTION

다중 채널을 대상으로 노이즈 필터 기능에 대응 되는 디지털 입력 상태를 확인합니다. 이 상태는 cedioLogicOne_Set, cedioLogicMulti_Set 함수를 통해 설정된 디지털 입력 로직(Digital Input)이 적용됩니다.

이 함수에서 전달되는 매개변수 중 'CutoffTime_us'는 입력 신호 유지 시간(Signal Width)을 의미합니다.

이 함수를 통해 확인되는 디지털 입력 채널들은 제한된 입력 신호 유지 시간 이상의 신호가 확인되어야만 합니다. 원하지 않는 신호(Noise) 나 일정 시간 이상의 펄스 입력(Pulse Input)을 확인할 때 매우 유용한 상태 확인 함수입니다.

PARAMETER

▶ ***IniChan*** **:** 시작 채널 번호. 통합 채널로 관리되는 채널 번호를 의미하며, 상수 값으로 0 (Zero Based) 이상, (최대 통합 채널수 - 1) 이하의 값을 채널 번호로 설정할 수 있습니다.

▶ ***NumChan*** **:** 시작 채널로부터 몇 개의 채널의 상태를 확인할 것인지에 대한 값을 전달합니다. (최대 32 개 채널까지 설정 가능합니다.)

▶ ***CutoffTime_us*** **:** 디지털 입력 신호 유지 시간을 마이크로 초(us) 단위로 설정합니다.

▶ ***States*** **:** 다중 채널에 대한 디지털 입력 상태를 반환합니다.

Value	Meaning
0 (CE_FALSE)	OFF
1 (CE_TRUE)	ON

RETURN VALUE

Value	Meaning
음수	수행 실패. 자세한 내용은 '에러처리' 편을 참고합니다
0 (ceERR_NONE)	수행 성공

EXAMPLE

C/C++

```
#include “ceSDK.h”
#include “ceSDKDef.h”

/* 0~3 번 채널은 디지털 출력 채널, 4~7 번 채널은 디지털 입력 채널이라 가정합니다.
출력 채널을 통해 일정 시간의 Pulse 출력을 내고,
입력 채널의 노이즈 필터 기능을 통해 입력 상태를 확인합니다. */

long nDoIniChNo = 0;                    // 디지털 출력 시작 채널 번호 설정
long nDoChNum = 4;                      // 디지털 출력 채널 개수
long nDiIniChNo = 4;                    // 디지털 입력 시작 채널 번호 설정
long nDiChNum = 4;                      // 디지털 입력 채널 개수
long nDiStates;                         // 디지털 입력 상태 정보
long nRetVal;

// 0 ~ 3 번 채널의 로직을 B 접점으로 설정합니다.
cedioLogicMulti_Set ( nDoIniChNo, nDoChNum, 0xF );

// 0 ~ 3 번 채널에 대해 10usec 동안 단일 펄스 출력을 발생합니다.
nRetVal = cedioPulseMulti (  nDoIniChNo,     // 디지털 출력 시작 채널 번호
                             nDoChNum,       // 디지털 출력 채널 개수
                             0xF,            // B 접점일 경우 Active High 상태가 됩니다.
                             10,             // 펄스 출력 시간. 10usec
                             CE_FALSE        // 펄스 출력 후 바로 반환
                             );

if ( nRetVal == ceERR_NONE )
{
    // 4 ~ 7 번 입력 채널의 노이즈 필터 기능을 활성화 하고,
    // 20 usec 입력 신호 유지 시간을 설정하여 입력 상태를 확인합니다.
    cedioMultiF_Get ( nDiIniChNo, nDiChNum, 20, &nDiStates );

    /* 10usec 펄스가 입력되어 노이즈 필터 기능에 의해 20usec 이하의 신호는 ON 신호로 인식하지 않습니다. */

}
```

Visual Basic

```
' 0 ~ 3 번 채널은 디지털 출력 채널, 4 ~ 7 번 채널은 디지털 입력 채널이라 가정합니다.
```

```
‘ 출력 채널을 통해 일정 시간의 Pulse 출력을 내고,
‘ 입력 채널의 노이즈 필터 기능을 통해 입력 상태를 확인합니다.

Dim nDoIniChNo As Long          ‘ 디지털 출력 시작 채널 번호
Dim nDoChNum As Long            ‘ 디지털 출력 채널 개수
Dim nDiIniChNo As Long          ‘ 디지털 입력 시작 채널 번호
Dim nDiChNum As Long            ‘ 디지털 입력 채널 개수
Dim nDiStates As Long           ‘ 디지털 입력 상태 정보

nDoIniChNo = 0
nDoChNum = 4
nDiIniChNo = 4
nDiChNum = 4

‘ 0 ~ 3 번 채널의 로직을 B 접점으로 설정합니다
Call cedioLogicOne_Set ( nDoIniChNo, nDoChNum, &HF )

‘ 0 ~ 3 번 채널에 대해 30usec 동안 단일 펄스 출력을 발생합니다.
If cedioPulseMulti ( nDoChNo, CE_TRUE, 30, CE_FALSE ) = ceERR_NONE Then

    ‘ 4 ~ 7 번 입력 채널의 노이즈 필터 기능을 활성화 하고,
    ‘ 20 usec 입력 신호 유지 시간을 설정하여 입력 상태를 확인합니다.
    Call cedioMultiF_Get ( nDiIniChNo, nDiChNum, 20, nDiStates )

    ‘ 30usec 펄스가 입력되었고 노이즈 필터는 20usec 로 설정되어 있으므로 ON 신호로 인식합니다.

End If
```

Delphi

```
{ 0 ~ 3 번 채널은 디지털 출력 채널, 4 ~ 7 번 채널은 디지털 입력 채널이라 가정합니다.
 출력 채널을 통해 일정 시간의 Pulse 출력을 내고,
 입력 채널의 노이즈 필터 기능을 통해 입력 상태를 확인합니다. }

var
    nDoIniChNo, nDoChNum, nDiIniChNo, nDiChNum : LongInt;        // 채널 정보
    nDiStates : LongInt;                                         // 디지털 입력 상태 정보

begin
    nDoIniChNo = 0              // 디지털 출력 시작 채널 번호
    nDoChNum = 4                // 디지털 출력 채널 개수
    nDiIniChNo = 4              // 디지털 입력 시작 채널 번호
    nDiChNum = 4                // 디지털 입력 채널 개수

    // 0 ~ 3 번 채널의 로직을 B 접점으로 설정합니다
    cedioLogicOne_Set ( nDoIniChNo, nDoChNum, $F );

    // 0 ~ 3 번 채널에 대해 30usec 동안 단일 펄스 출력을 발생합니다.
    if cedioPulseMulti ( nDoChNo, CE_TRUE, 30, CE_FALSE ) = ceERR_NONE then
    begin
```

```
            { 4 ~ 7 번 입력 채널의 노이즈 필터 기능을 활성화 하고,
             20 usec 입력 신호 유지 시간을 설정하여 입력 상태를 확인합니다. }
            cedioMultiF_Get ( nDiIniChNo, nDiChNum, 20, @nDiStates ) ;

            // 30usec 펄스가 입력되었고 노이즈 필터는 20usec 로 설정되어 있으므로 ON 신호로 인식합니다.
    end;
end;
```

NAME

cedioPulseOne

- 대상 디지털 출력(Digital Output) 채널을 통해 단일 펄스 출력을 발생

INFORMATION	
1	Universal DIO Control
!	VC++ (6, 7, 8)/VB BCB/Delphi
:	Level 1
J	위험 요소 없음

SYNOPSIS

r VT_I4 cedioPulseOne ([in] VT_I4 Channel, [in] VT_I4 IsOnPulse, [in] VT_I4 Duration, [in] VT_I4 IsWaitPulseEnd)

DESCRIPTION

이 함수는 대상 디지털 출력 채널 (Digital Output Channel) 을 통해 단일 펄스 출력을 발생 합니다.

PARAMETER

▶ ***Channel*** **:** 채널 번호. 통합 채널로 관리되는 채널 번호를 의미하며, 상수 값으로 0 (Zero Based) 이상, (최대 통합 채널수 - 1) 이하의 값을 채널 번호로 설정할 수 있습니다.

▶ ***IsOnPulse*** **:** 설정된 디지털 출력 논리(Logic)에 따라 초기 펄스 출력의 형태를 결정합니다. 만약 A 접점의 디지털 출력 논리 상황에서 이 매개변수의 값이 CE_TRUE 일 경우에는 Active Low 상태가 되며, B 접점의 디지털 출력 논리 상황에서는 Active High 상태가 됩니다.

Value	Meaning
0 (CE_FALSE)	A 접점 : Active High, B 접점 : Active Low
1 (CE_TRUE)	A 접점 : Active Low, B 접점 : Active High

▶ ***Duration*** **:** 펄스 출력 시간을 설정합니다. 이 출력 시간은 Active 된 펄스 출력 시간을 의미합니다.

▶ ***IsWaitPulseEnd*** **:** 펄스 출력 시작 후 함수를 바로 반환할 것인지, 아니면 펄스 출력 시간 동안 함수 반환을 대기할 것인지를 결정합니다.

Value	Meaning
0 (CE_FALSE)	펄스 출력 시작 후 바로 함수를 반환합니다.
1 (CE_TRUE)	함수의 반환 시점을 펄스 출력 종료 시점까지 대기합니다.

RETURN VALUE

Value	Meaning
음수	수행 실패. 자세한 내용은 '에러처리' 편을 참고합니다.
0 (ceERR_NONE)	수행 성공.

EXAMPLE

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

NAME

cedioPulseMulti

- 다중 디지털 출력(Digital Output) 채널을 통해 단일 펄스 출력 발생

INFORMATION	
1	Universal DIO Control
!	VC++ (6, 7, 8)/VB
	BCB/Delphi
:	Level 1
J	위험 요소 없음

SYNOPSIS

r VT_I4 cedioPulseMulti ([in] VT_I4 IniChan, [in] VT_I4 NumChan, [in] VT_I4 OutStates, [in] VT_I4 Duration, [in] VT_I4 IsWaitPulseEnd)

DESCRIPTION

다중 디지털 출력 채널(Digital Output Channel) 을 통해 단일 펄스 출력을 발생합니다.

PARAMETER

▶ ***IniChan*** **:** 시작 채널 번호. 통합 채널로 관리되는 채널 번호를 의미하며, 상수 값으로 0 (Zero Based) 이상, (최대 통합 채널수 - 1) 이하의 값을 채널 번호로 설정할 수 있습니다.

▶ ***NumChan*** **:** 시작 채널로부터 몇 개의 채널의 상태를 확인할 것인지에 대한 값을 전달합니다. (최대 32 개 채널까지 설정 가능합니다.)

▶ ***OutStates*** **:** 설정된 디지털 출력 논리에 따라 초기 펄스 출력의 형태를 결정합니다. 만약 A 접점의 디지털 출력 논리 상황에서 이 매개변수의 값이 CE_TRUE 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.

Value	Meaning
0 (CE_FALSE)	A 접점 : Active High, B 접점 : Active Low
1 (CE_TRUE)	A 접점 : Active Low, B 접점 : Active High

▶ ***Duration*** **:** 펄스 출력 시간을 설정합니다. 이 출력 시간은 Active 된 펄스 출력 시간을 의미합니다.

▶ ***IsWaitPulseEnd*** **:** 펄스 출력 시작 후 함수를 바로 반환할 것인지, 아니면 펄스 출력 시간 동안 함수 반환을 대기할 것인지를 결정합니다.

Value	Meaning
0 (CE_FALSE)	펄스 출력 시작 후 바로 함수를 반환합니다.
1 (CE_TRUE)	함수의 반환 시점을 펄스 출력 종료 시점까지 대기합니다.

RETURN VALUE

Value	Meaning
음수	수행 실패. 자세한 내용은 '에러처리' 편을 참고합니다.
0 (ceERR_NONE)	수행 성공.

EXAMPLE

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

Chapter

13

Counter Control

사용자의 신호 입력 속도에 따라 4 단계로 설정할 수 있는 디지털 필터 내장 및 과전류로 인한 제품의 손상을 방지할 수 있는 Over-current shutdown 기능은 시스템의 전체적인 안정성을 향상시킵니다.

이 단원에서는 카운터 함수들을 소개합니다. ㈜ 커미조아의 카운터 모듈은 카운터는 펄스 신호를 카운트 할 경우에 사용하며, 카운트 Overflow 체크, Cutoff Frequency 설정 기능을 제공하여 시스템의 신뢰성을 향상시킵니다.

13 카운터 (Counter) 편

이 단원에서는 카운터와 관련된 함수를 소개합니다. ㈜ 커미조아의 카운터 모듈은 카운터는 펄스 신호를 카운트 할 경우에 사용하며, 사용자의 신호 입력 속도에 따라 4 단계로 설정할 수 있는 디지털 필터를 내장하여 안정성을 더욱 높였습니다.

또한 외부로 제공되는 24V 전원에 대해 Over-current shutdown 기능을 제공합니다. 이 기능은 외부 배선 시 전원이 단락 되는 등의 과전류 사용 시 5A 이상의 전류를 사용할 경우 자동으로 전원을 suutdown 하고, 과전류로 인한 제품의 손상을 방지합니다.

13.1 함수 요약

카운터에 관련된 함수는 다음과 같습니다.

Summary of Functions
r VT_I4 cecEdgeOne_Set ([in] VT_I4 Channel, [out] DWORD EdgeMode) 대상 카운터 채널의 Edge Mode 를 설정합니다.
r VT_I4 cecEdgeOne_Get ([in] VT_I4 Channel, [out] PDWORD EdgeMode) 대상 카운터 채널의 Edge Mode 설정 상태를 반환합니다.
r VT_I4 cecEdgeMulti_Set ([in] VT_I4 IniChannel, [in] VT_I4 NumChannels, [in] DWORD EdgeModeMask) 대상 카운터 채널 범위에 포함된 채널들의 Edge Mode 를 설정합니다.
r VT_I4 cecEdgeMulti_Get ([in] VT_I4 IniChannel, [in] VT_I4 NumChannels, [out] PDWORD EdgeModeMask) 대상 카운터 채널 범위에 포함된 채널들의 Edge Mode 설정 상태를 반환합니다.
r VT_I4 cecClearOne ([in] VT_I4 Channel) 대상 카운터 채널의 카운트 값을 Clear (0 으로 리셋) 시킵니다.
r VT_I4 cecClearMulti ([in] VT_I4 IniChannel, [in] VT_I4 NumChannels) 대상 카운터 채널 범위에 포함된 채널들의 카운트 값을 모두 Clear (0 으로 리셋) 시킵니다.
r VT_I4 cecClearAll ([in] VT_I4 NodeID) 대상 원격 노드의 모든 카운터 채널의 카운트 값을 Clear (0 으로 리셋) 시킵니다.
r VT_I4 cec_Get ([in] VT_I4 Channel, [out] PDWORD Count) 대상 카운터 채널의 카운트 값을 얻어 옵니다.
r VT_I4 cecEnableOne_Set ([in] VT_I4 Channel, [in] DWORD Enable) 대상 카운터 채널의 카운트 기능의 활성/비활성 상태를 설정합니다.
r VT_I4 cecEnableOne_Get ([in] VT_I4 Channel, [out] PDWORD IsEnabled) 대상 카운터 채널의 카운트 기능의 활성/비활성 상태를 반환합니다.
r VT_I4 cecEnableMulti_Set ([in] VT_I4 IniChannel, [in] VT_I4 NumChannels, [in] DWORD EnableMask) 대상 카운터 채널 범위에 포함된 채널들의 카운트 기능의 활성/비활성 상태를 설정합니다.

r VT_I4 cecEnableMulti_Get ([in] VT_I4 IniChannel, [in] VT_I4 NumChannels, [out] PDWORD EnabledMask) 대상 카운터 채널 범위에 포함된 채널들의 카운트 기능의 활성/비활성 상태를 반환합니다.
r VT_I4 cecOverflowFlagGetOne ([in] VT_I4 Channel, [out] PDWORD OverflowStatus) 대상 카운터 채널의 Overflow 상태를 얻어 옵니다.
r VT_I4 cecOverflowFlagGetMulti ([in] VT_I4 IniChannel, [in] VT_I4 NumChannels, [out] PDWORD OverflowStates) 대상 카운터 채널 범위의 Overflow 상태를 얻어 옵니다.
r VT_I4 cecOverflowFlagClearOne ([in] VT_I4 Channel) 대상 카운터 채널에 해당하는 Overflow Flag 를 0 으로 Clear 시킵니다.
r VT_I4 cecOverflowFlagClearMulti ([in] VT_I4 IniChannel, [in] VT_I4 NumChannels) 대상 카운터 채널 범위에 해당하는 채널 별 Overflow Flag 를 0 으로 Clear 시킵니다.
r VT_I4 cecOverflowFlagClearAll ([in] VT_I4 NodeID) 대상 원격 노드의 모든 카운터 채널의 Overflow Flag 를 0 으로 Clear 시킵니다.
r VT_I4 cecFilterFreq_Set ([in] VT_I4 Channel, [in] VT_I4 FilterFreq) 대상 카운터 채널 별 필터 Cutoff 주파수를 설정합니다.
r VT_I4 cecFilterFreq_Get ([in] VT_I4 Channel, [out] VT_PI4 FilterFreq) 대상 카운터 채널 별 필터 Cutoff 주파수 설정값을 반환합니다.

13.2 함수 설명

NAME

cecEdgeOne_Set / cecEdgeOne_Get

- 대상 카운터 채널의 Edge Mode 설정 및 반환

INFORMATION	
1	Counter
!	VC++ (6, 7, 8)/VB
	BCB/Delphi
:	Level 7
J	위험 요소 없음

SYNOPSIS

r VT_I4 cecEdgeOne_Set ([in] VT_I4 Channel, [in] DWORD EdgeMode)

r VT_I4 cecEdgeOne_Get ([in] VT_I4 Channel, [out] PDWORD EdgeMode)

DESCRIPTION

cecEdgeOne_Set 함수는 대상 카운터 채널의 Edge Mode 를 설정합니다.

cecEdgeOne_Get 함수는 대상 카운터 채널의 Edge Mode 설정값을 반환합니다.

PARAMETER

▶ ***Channel*** : 채널 번호. 통합 채널로 관리되는 채널 번호를 의미하며, 상수 값으로 0 (Zero Based) 이상, (최대 통합 채널수 - 1) 이하의 값을 채널 번호로 설정할 수 있습니다.

▶ ***EdgeMode*** : 대상 카운터 채널의 Edge Mode 를 설정합니다.

Value	Meaning
0 [Default]	Falling Edge.
1	Rising Edge.

RETURN VALUE

Value	Meaning
음수	수행 실패. 자세한 내용은 '에러처리' 편을 참고합니다.
0 (ceERR_NONE)	수행 성공.

EXAMPLE

C/C++

```
#include “ceSDK.h”
#include “ceSDKDef.h”

long nCntChNo = 0;                    // 카운터 채널 번호 설정
long nEdgeMode;                       // Edge Mode 설정 정보

/* 카운터 채널의 Edge Mode 를 확인 후, Edge Mode 를 Rising Edge 로 설정합니다.

if ( cecEdgeOne_Get ( nCntChNo, &nEdgeMode ) == ceERR_NONE )
{
    if ( nEdgeMode != CE_TRUE )
    {
            // Edge Mode 를 Rising Edge 로 설정합니다.
            cecEdgeOne_Set ( nCntChNo, CE_TRUE );
    }
}
```

Visual Basic

```
Dim nCntChNo As Long                  ‘ 카운터 채널 번호 설정
Dim nEdgeMode As Long                 ‘ Edge Mode 설정 정보

nCntChNo = 0

‘ 카운터 채널의 Edge Mode 를 확인 후, Edge Mode 를 Rising Edge 로 설정합니다.

If cecEdgeOne_Get ( nCntChNo, nEdgeMode ) = ceERR_NONE Then

    If nEdgeMode <> CE_TRUE  Then

            ‘ Edge Mode 를 Rising Edge 로 설정합니다.
            Call cecEdgeOne_Set ( nCntChNo, CE_TRUE )
    End If

End If
```

Delphi

```
var
    nCntChNo : LongInt;               // 카운터 채널 번호 설정
    nEdgeMode : LongInt;              // Edge Mode 설정 정보

begin
    nCntChNo := 0;

    // 카운터 채널의 Edge Mode 를 확인 후, Edge Mode 를 Rising Edge 로 설정합니다.
```

```
    if cecEdgeOne_Get ( nCntChNo, @nEdgeMode ) = ceERR_NONE then
    begin
            if nEdgeMode <> CE_TRUE  then
            begin
                    // Edge Mode 를 Rising Edge 로 설정합니다.
                    cecEdgeOne_Set ( nCntChNo, CE_TRUE );
            end;
    end;

end;
```

NAME

cecEdgeMulti_Set / cecEdgeMulti_Get

- 다중 카운터 채널의 Edge Mode 설정 및 반환

INFORMATION
1 Counter
! VC++ (6, 7, 8)/VB
BCB/Delphi
: Level 7
J 위험 요소 없음

SYNOPSIS

r VT_I4 cecEdgeMulti_Set ([in] VT_I4 IniChannel, [in] VT_I4 NumChannels, [in] DWORD EdgeModeMask)

r VT_I4 cecEdgeMulti_Get ([in] VT_I4 IniChannel, [in] VT_I4 NumChannels, [out] PDWORD EdgeModeMask)

DESCRIPTION

cecEdgeMulti_Set 함수는 대상 카운터 채널 범위에 포함된 채널들의 Edge Mode 를 설정합니다.

cecEdgeMulti_Get 함수는 대상 카운터 채널 범위에 포함된 채널들의 Edge Mode 설정 상태를 반환합니다.

PARAMETER

▶ *IniChannel* : 채널 번호. 통합 채널로 관리되는 채널 번호를 의미하며, 상수 값으로 0 (Zero Based) 이상, (최대 통합 채널수 - 1) 이하의 값을 채널 번호로 설정할 수 있습니다.

▶ *NumChannels* : 시작 채널로부터 몇 개의 채널을 대상 채널로 할 것인지에 대한 값을 전달합니다. 최대 32 개 채널까지 설정 가능합니다.

▶ *EdgeModeMask* : 시작 채널 번호 부터 끝 채널 번호까지 각 채널 별 Edge Mode 를 해당 비트 별 마스크 로 설정합니다.

Value	Meaning
0 [Default]	Falling Edge.
1	Rising Edge.

RETURN VALUE

Value	Meaning
음수	수행 실패. 자세한 내용은 '에러처리' 편을 참고합니다.
0 (ceERR_NONE)	수행 성공.

EXAMPLE

C/C++

```
#include “ceSDK.h”
#include “ceSDKDef.h”

long nCntIniChNo = 0;          // 카운터 시작 채널 번호 설정
long nCntChNum = 4;            // 카운터 채널 개수
long nEdgeModeMask;            // Edge Mode 설정 정보

/* 0 ~ 3 번 카운터 채널의 Edge Mode 를 확인 후, Edge Mode 를
0, 2 번 채널은 Rising Edge, 1, 3 번 채널은 Falling Edge 로 설정합니다. */

if ( cecEdgeMulti_Get ( nCntIniChNo, nCntChNum, &nEdgeModeMask ) == ceERR_NONE )
{
    if ( nEdgeModeMask != 0x5 )
    {
            // Edge Mode 를 해당 비트 별 마스크로 설정합니다.
            cecEdgeOne_Set ( nCntIniChNo, nCntChNum, 0x5 );
    }
}
```

Visual Basic

```
Dim nCntIniChNo As Long          ‘ 카운터 시작 채널 번호 설정
Dim nCntChNum As Long            ‘ 카운터 채널 개수
Dim nEdgeModeMask As Long        ‘ Edge Mode 설정 정보

nCntIniChNo = 0
nCntChNum = 4

‘ 0 ~ 3 번 카운터 채널의 Edge Mode 를 확인 후, Edge Mode 를
‘ 0, 2 번 채널은 Falling Edge, 1, 3 번 채널은 Rising Edge 로 설정합니다.

If cecEdgeMulti_Get ( nCntIniChNo, nCntChNum, nEdgeModeMask ) = ceERR_NONE Then

    If nEdgeModeMask <> &HA  Then

            ‘ Edge Mode 를 해당 비트 별 마스크로 설정합니다.
            Call cecEdgeOne_Set ( nCntIniChNo, nCntChNum, &HA )
    End If

End If
```

Delphi

```
var
    nCntIniChNo : LongInt;          // 카운터 시작 채널 번호 설정
```

```
    nCntChNum : LongInt;                // 카운터 채널 개수
    nEdgeModeMask : LongInt;            // Edge Mode 설정 정보

begin
    nCntIniChNo := 0;
    nCntChNum := 4;

    { 0 ~ 3 번 카운터 채널의 Edge Mode 를 확인 후, Edge Mode 를
     0, 2 번 채널은 Falling Edge, 1, 3 번 채널은 Rising Edge 로 설정합니다. }

    if cecEdgeMulti_Get ( nCntIniChNo, nCntChNum, @nEdgeModeMask ) = ceERR_NONE then
    begin
            if nEdgeModeMask <> $A then
            begin
                    // Edge Mode 를 해당 비트 별 마스크로 설정합니다.
                    cecEdgeOne_Set ( nCntIniChNo, nCntChNum, $A );
            end;
    end;
end;
```

NAME

cecClearOne / cecClearMulti / cecClearAll
- 대상 카운터 채널의 카운트 값 Clear

INFORMATION
1 Counter
! VC++ (6, 7, 8)/VB
BCB/Delphi
: Level 7
J 위험 요소 없음

SYNOPSIS

r VT_I4 cecClearOne ([in] VT_I4 Channel)

r VT_I4 cecClearMulti ([in] VT_I4 IniChannel, [in] VT_I4 NumChannels)

r VT_I4 cecClearAll ([in] VT_I4 NodeID)

DESCRIPTION

cecClearOne 함수는 대상 카운터 채널의 카운트 값을 Clear (0 으로 리셋) 시킵니다.

cecClearMulti 함수는 대상 카운터 채널 범위에 포함된 채널들의 카운트 값을 모두 Clear 시킵니다.

cecClearAll 함수는 대상 원격 노드의 모든 카운터 채널의 카운트 값을 Clear 시킵니다.

PARAMETER

▶ ***Channel*** : cecClearOne 함수의 인자로서 채널 번호. 통합 채널로 관리되는 채널 번호를 의미하며, 상수 값으로 0 (Zero Based) 이상, (최대 통합 채널수 - 1) 이하의 값을 채널 번호로 설정할 수 있습니다.

▶ ***IniChannel*** : cecClearMulti 함수의 인자로서 시작 채널 번호. 채널 번호는 상수 값으로 0 번째 채널을 기준 채널로 임의의 채널을 설정 할 수 있습니다.

▶ ***NumChannels*** : cecClearMulti 함수의 인자로서 시작 채널로부터 몇 개의 채널을 대상 채널로 할 것인지에 대한 값을 전달합니다.

▶ ***NodeID*** : cecClearAll 함수의 인자로서 대상 카운터 모듈이 속해 있는 원격 노드 ID 를 설정합니다.

RETURN VALUE

Value	Meaning
음수	수행 실패. 자세한 내용은 '에러처리' 편을 참고합니다
0 (ceERR_NONE)	수행 성공

EXAMPLE

C/C++

```
#include “ceSDK.h”
#include “ceSDKDef.h”

long nCntChNo = 1;          // 단일 카운터 채널 번호
long nCntIniChNo = 0;       // 카운터 시작 채널 번호
long nCntChNum = 4;         // 카운터 채널 개수
long nNodeID = 1;           // 원격 노드 ID. 카운터 모듈이 속한 원격 노드의 ID

// 단일 카운트 채널의 카운트 값을 0 으로 Clear 합니다.
if ( cecClearOne ( nCntChNo ) != ceERR_NONE )
{
    OutputDebugString ( "cecClearOne has been failed" );
}

// 0 ~ 3 번 카운트 채널의 카운트 값을 0 으로 Clear 합니다.
if ( cecClearMulti ( nCntChNo, nCntChNum ) != ceERR_NONE )
{
    OutputDebugString ( "cecClearMulti has been failed" );
}

// 해당 원격 노드 내의 모든 카운트 채널에 대해 카운트 값을 0 으로 Clear 합니다.
if ( cecClearAll ( nNodeID ) != ceERR_NONE )
{
    OutputDebugString ( "cecClearAll has been failed" );
}
```

Visual Basic

```
Dim nCntChNo As Long            ‘ 단일 카운터 채널 번호
Dim nCntIniChNo As Long         ‘ 카운터 시작 채널 번호
Dim nCntChNum As Long           ‘ 카운터 채널 개수
Dim nNodeID As Long             ‘ 원격 노드 ID. 카운터 모듈이 속한 원격 노드의 ID

nCntChNo = 1
nCntIniChNo = 0
nCntChNum = 4
nNodeID = 1

‘ 단일 카운트 채널의 카운트 값을 0 으로 Clear 합니다.
If cecClearOne ( nCntChNo ) <> ceERR_NONE Then
    MsgBox ( "cecClearOne has been failed" )
End If

‘ 0 ~ 3 번 카운트 채널의 카운트 값을 0 으로 Clear 합니다.
If cecClearMulti ( nCntChNo, nCntChNum ) <> ceERR_NONE Then
    MsgBox ( "cecClearMulti has been failed" )
End If

‘ 해당 원격 노드 내의 모든 카운트 채널에 대해 카운트 값을 0 으로 Clear 합니다.
If cecClearAll ( nNodeID ) <> ceERR_NONE Then
```

```
    MsgBox ( "cecClearAll has been failed" )
End If
```

Delphi

```
var
    nCntChNo : LongInt;                 // 단일 카운터 채널 번호
    nCntIniChNo : LongInt;              // 카운터 시작 채널 번호
    nCntChNum : LongInt;                // 카운터 채널 개수
    nNodeID : LongInt;                  // 원격 노드 ID. 카운터 모듈이 속한 원격 노드의 ID

begin
    nCntChNo := 1;
    nCntIniChNo := 0;
    nCntChNum := 4;
    nNodeID := 1;

    // 단일 카운트 채널의 카운트 값을 0 으로 Clear 합니다.
    if cecClearOne ( nCntChNo ) <> ceERR_NONE then
    begin
            ShowMessage ( 'cecClearOne has been failed' );
    end;

    // 0 ~ 3 번 카운트 채널의 카운트 값을 0 으로 Clear 합니다.
    if cecClearMulti ( nCntChNo, @nCntChNum ) <> ceERR_NONE then
    begin
            ShowMessage ( 'cecClearMulti has been failed' );
    end;

    // 해당 원격 노드 내의 모든 카운트 채널에 대해 카운트 값을 0 으로 Clear 합니다.
    if cecClearAll ( nNodeID ) <> ceERR_NONE then
    begin
            ShowMessage ( 'cecClearAll has been failed' );
    end;
end;
```

NAME

cec_Get

- 대상 카운터 채널의 카운트 값 반환

INFORMATION	
1	Counter
!	VC++ (6, 7, 8)/VB
	BCB/Delphi
:	Level 7
J	위험 요소 없음

SYNOPSIS

r VT_I4 cec_Get ([in] VT_I4 Channel, [out] PDWORD Count)

DESCRIPTION

대상 카운터 채널의 카운트 값을 반환합니다.

PARAMETER

▶ *Channel* : 채널 채널 번호. 통합 채널로 관리되는 채널 번호를 의미하며, 상수 값으로 0 (Zero Based) 이상, (최대 통합 채널수 - 1) 이하의 값을 채널 번호로 설정할 수 있습니다.

▶ *Count* : 이 매개변수를 통하여 대상 채널의 카운트 값을 반환합니다.

RETURN VALUE

Value	Meaning
음수	수행 실패. 자세한 내용은 '에러처리' 편을 참고합니다
0 (ceERR_NONE)	수행 성공

EXAMPLE

C/C++

```
#include “ceSDK.h”
#include “ceSDKDef.h”

long nCntChNo = 1;                  // 카운터 채널 번호
long nCount;                        // 카운트 값
long nOverFlowState = CE_FALSE;     // 대상 채널의 Overflow 상태 정보

// 채널의 카운트 기능을 활성 상태로 설정합니다.
cecEnableOne_Set ( nCntChNo, CE_TRUE );

/* 대상 카운터 채널을 통해 카운트를 시작합니다.
Overflow 가 발생하면 카운트를 멈추고 Overflow 상태를 해제합니다. */
while ( nOverFlowState == CE_FALSE )
{
    // 채널의 카운트 값을 반환합니다.
    if ( cec_Get ( nCntChNo, &nCount ) == ceERR_NONE )
    {
            cecOverflowFlagGetOne ( nCntChNo, &nOverFlowState );
    }
}

// 카운트 채널에 Overflow 가 발생하면 while 반복문을 멈추고 Overflow 상태를 해제합니다.
if ( cecOverflowFlagClearOne ( nCntChNo ) != ceERR_NONE )
{
    OutputDebugString ( “cecOverflowFlagClearOne has been failed” );
}
```

Visual Basic

```
Dim nCntChNo As Long            ‘ 카운터 채널 번호
Dim nCount As Long              ‘ 카운트 값
Dim nOverFlowState As Long      ‘ 대상 채널의 Overflow 상태 정보

nCntChNo = 1
nOverFlowState = CE_FALSE

‘ 채널의 카운트 기능을 활성 상태로 설정합니다.
Call cecEnableOne_Set ( nCntChNo, CE_TRUE )

‘ 대상 카운터 채널을 통해 카운트를 시작합니다.
‘ Overflow 가 발생하면 카운트를 멈추고 Overflow 상태를 해제합니다.
While nOverFlowState = CE_FALSE Then

    ‘ 채널의 카운트 값을 반환합니다.
    If cec_Get ( nCntChNo, nCount ) = ceERR_NONE Then
            Call cecOverflowFlagGetOne ( nCntChNo, &nOverFlowState );
    End If
```

```
Wend

‘ 카운트 채널에 Overflow 가 발생하면 while 반복문을 멈추고 Overflow 상태를 해제합니다.
If cecOverflowFlagClearOne ( nCntChNo ) <> ceERR_NONE Then
    MsgBox ( “cecOverflowFlagClearOne has been failed” )
End If
```

Delphi

```
var
    nCntChNo : LongInt;                 // 카운터 채널 번호
    nCount : LongInt;                   // 카운트 값
    nOverFlowState : LongInt;           // 대상 채널의 Overflow 상태 정보

begin
    nCntChNo := 1;
    nOverFlowState := CE_FALSE;

    // 채널의 카운트 기능을 활성 상태로 설정합니다.
    cecEnableOne_Set ( nCntChNo, CE_TRUE );

    // 대상 카운터 채널을 통해 카운트를 시작합니다.
    // Overflow 가 발생하면 카운트를 멈추고 Overflow 상태를 해제합니다.
    while nOverFlowState = CE_FALSE do
    begin
            // 채널의 카운트 값을 반환합니다.
            if cec_Get ( nCntChNo, @nCount ) = ceERR_NONE then
            begin
                    cecOverflowFlagGetOne ( nCntChNo, @nOverFlowState )
            end;
    end;

    // 카운트 채널에 Overflow 가 발생하면 while 반복문을 멈추고 Overflow 상태를 해제합니다.
    if cecOverflowFlagClearOne ( nCntChNo ) <> ceERR_NONE then
    begin
            ShowMessage ( ‘cecOverflowFlagClearOne has been failed’ );
    end;

end;
```

NAME

cecEnableOne_Set / cecEnableOne_Get

- 대상 카운터 채널의 카운트 기능 활성/비활성 상태 설정 및 반환

INFORMATION	
1	Counter
!	VC++ (6, 7, 8)/VB
	BCB/Delphi
:	Level 7
J	위험 요소 없음

SYNOPSIS

r VT_I4 cecEnableOne_Set ([in] VT_I4 Channel, [in] DWORD Enable)

r VT_I4 cecEnableOne_Get ([in] VT_I4 Channel, [out] PDWORD IsEnabled)

DESCRIPTION

cecEnableOne_Set 함수는 대상 카운터 채널의 카운트 기능 활성/비활성 상태를 설정합니다.

cecEnableOne_Get 함수는 대상 카운터 채널의 카운트 기능 활성/비활성 설정 상태를 반환합니다.

PARAMETER

▶ ***Channel*** : 채널 번호. 통합 채널로 관리되는 채널 번호를 의미하며, 상수 값으로 0 (Zero Based) 이상, (최대 통합 채널수 – 1) 이하의 값을 채널 번호로 설정할 수 있습니다.

▶ ***Enable*** : 대상 카운터 채널의 카운트 기능 활성/비활성 상태 설정 및 설정 상태를 반환합니다.

Value	Meaning
0 [Default]	Count Disable.
1	Count Enable.

RETURN VALUE

Value	Meaning
음수	수행 실패. 자세한 내용은 '에러처리' 편을 참고합니다.
0 (ceERR_NONE)	수행 성공.

EXAMPLE

C/C++

```
#include “ceSDK.h”
#include “ceSDKDef.h”

long nCntChNo = 1;              // 단일 카운터 채널 번호
long nCntEnable;                // 카운트 기능 활성/비활성 여부

// 채널의 카운트 기능을 활성 상태로 설정합니다.
if ( cecEnableOne_Get ( nCntChNo, nCntEnable ) == ceERR_NONE )
{
    if ( nCntEnable != CE_TRUE )
    {
            cecEnableOne_Set ( nCntChNo, CE_TRUE );
    }
}
```

Visual Basic

```
Dim nCntChNo As Long        ‘ 카운터 채널 번호
Dim nCntEnable As Long      ‘ 카운트 기능 활성/비활성 여부

nCntChNo = 1

‘ 채널의 카운트 기능을 활성 상태로 설정합니다.
If cecEnableOne_Get ( nCntChNo, nCntEnable ) = ceERR_NONE Then

    if nCntEnable <> CE_TRUE Then
            Call cecEnableOne_Set ( nCntChNo, CE_TRUE )
    End If

End If
```

Delphi

```
var
    nCntChNo : LongInt;         // 카운터 채널 번호
    nCntEnable : LongInt;       // 카운트 기능 활성/비활성 여부

begin
    nCntChNo := 1;

    // 채널의 카운트 기능을 활성 상태로 설정합니다.
    if cecEnableOne_Get ( nCntChNo, @nCntEnable ) = ceERR_NONE then
    begin
            if nCntEnable <> CE_TRUE then
            begin
                    cecEnableOne_Set ( nCntChNo, CE_TRUE );
            end;
    end;

end;
```

NAME

cecEnableMulti_Set / cecEnableMulti_Get
- 다중 카운터 채널의 카운트 기능 활성/비활성 상태 설정 및 반환

INFORMATION	
1	Counter
!	VC++ (6, 7, 8)/VB
	BCB/Delphi
:	Level 7
J	위험 요소 없음

SYNOPSIS

r VT_I4 cecEnableOne_Set ([in] VT_I4 IniChannel, [in] VT_I4 NumChannels, [in] DWORD EnableMask)

r VT_I4 cecEnableOne_Get ([in] VT_I4 IniChannel, [in] VT_I4 NumChannels, [out] PDWORD EnabledMask)

DESCRIPTION

cecEnableMulti_Set 함수는 다중 카운터 채널의 카운트 기능 활성/비활성 상태를 설정합니다.

cecEnableMulti_Get 함수는 다중 카운터 채널의 카운트 기능 활성/비활성 설정 상태를 반환합니다.

PARAMETER

▶ *IniChannel* : 시작 채널 번호. 통합 채널로 관리되는 채널 번호를 의미하며, 상수 값으로 0 (Zero Based) 이상, (최대 통합 채널수 - 1) 이하의 값을 채널 번호로 설정할 수 있습니다.

▶ *NumChannels* : 시작 채널로부터 몇 개의 채널을 대상 채널로 할 것인지에 대한 값을 전달합니다.

▶ *EnableMask* : 이 매개변수를 통하여 다중 카운터 채널의 카운트 기능 활성/비활성 상태를 해당 비트 별 마스크로 설정 혹은 설정 상태를 반환합니다.

Value	Meaning
0 [Default]	Count Disable.
1	Count Enable.

RETURN VALUE

Value	Meaning
음수	수행 실패. 자세한 내용은 '에러처리' 편을 참고합니다.
0 (ceERR_NONE)	수행 성공.

EXAMPLE

C/C++

```
#include “ceSDK.h”
#include “ceSDKDef.h”

long nCntIniChNo = 0;          // 카운터 시작 채널 번호
long nCntChNum = 4;            // 카운터 채널 개수
long nCntEnableMask;           // 카운트 기능 활성/비활성 여부

// 0 ~ 3 번 채널의 카운트 기능을 활성 상태로 설정합니다.
if ( cecEnableMulti_Get ( nCntIniChNo, nCntChNum, &nCntEnableMask ) == ceERR_NONE )
{
    if ( nCntEnableMask != 0xF )
    {
            cecEnableMulti_Set ( nCntIniChNo, nCntChNum, 0xF );
    }
}
```

Visual Basic

```
Dim nCntIniChNo As Long          ‘ 카운터 시작 채널 번호
Dim nCntChNum As Long            ‘ 카운터 채널 개수
Dim nCntEnableMask As Long       ‘ 카운트 기능 활성/비활성 여부

nCntIniChNo = 0
nCntChNum = 4

‘ 0 ~ 3 번 채널의 카운트 기능을 활성 상태로 설정합니다.
If cecEnableMulti_Get ( nCntIniChNo, nCntChNum, nCntEnableMask ) = ceERR_NONE Then

    If nCntEnableMask <> &HF Then

            Call cecEnableMulti_Set ( nCntIniChNo, nCntChNum, &HF )
    End If

End If
```

Delphi

```
var
    nCntIniChNo : LongInt;          // 카운터 시작 채널 번호
    nCntChNum : LongInt;            // 카운터 채널 개수
    nCntEnableMask : LongInt;       // 카운트 기능 활성/비활성 여부

begin
    nCntIniChNo := 0;
    nCntChNum := 4;

    // 0 ~ 3 번 채널의 카운트 기능을 활성 상태로 설정합니다.
```

```
    if cecEnableMulti_Get ( nCntIniChNo, nCntChNum, @nCntEnableMask ) = ceERR_NONE then
    begin
            if nCntEnableMask <> $F then
            begin
                    cecEnableMulti_Set ( nCntIniChNo, nCntChNum, $F );
            end;
    end;

end;
```

NAME

cecOverflowFlagGetOne /

cecOverflowFlagGetMulti

- 대상 카운터 채널의 Overflow 상태 반환

INFORMATION	
1	Counter
!	VC++ (6, 7, 8)/VB
	BCB/Delphi
:	Level 7
J	위험 요소 없음

SYNOPSIS

r VT_I4 cecOverflowFlagGetOne ([in] VT_I4 Channel, [out] PDWORD OverrflowStatus)

r VT_I4 cecOverflowFlagGetMulti ([in] VT_I4 IniChannel, [in] VT_I4 NumChannels, [out] PDWORD OverflowStates)

DESCRIPTION

cecOverflowFlagGetOne 함수는 대상 카운터 채널의 카운트 Overflow 상태를 반환합니다.
cecOverflowFlagGetMulti 함수는 대상 카운터 채널 범위에 포함된 채널들의 카운트 Overflow 상태를 반환합니다.

PARAMETER

▶ *Channel* : cecOverflowFlagGetOne 함수의 인자로서 채널 번호. 통합 채널로 관리되는 채널 번호를 의미하며, 상수 값으로 0 (Zero Based) 이상, (최대 통합 채널수 - 1) 이하의 값을 채널 번호로 설정할 수 있습니다.

▶ *IniChannel* : cecOverflowFlagGetMulti 함수의 인자로서 시작 채널 번호. 채널 번호는 상수 값으로 0 번째 채널을 기준 채널로 임의의 채널을 설정 할 수 있습니다.

▶ *NumChannels* : cecOverflowFlagGetMulti 함수의 인자로서 시작 채널로부터 몇 개의 채널을 대상 채널로 할 것인지에 대한 값을 전달합니다. 최대 32 개 채널까지 설정 가능합니다.

▶ *OverflowStatus* : 대상 카운터 채널의 Overflow 상태(24 비트 범위 초과)를 반환합니다.

Value	Meaning
0	Overflow 발생 안함.
1	Overflow 발생 함.

RETURN VALUE

Value	Meaning
음수	수행 실패. 자세한 내용은 '에러처리' 편을 참고합니다.
0 (ceERR_NONE)	수행 성공.

EXAMPLE

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

NAME

cecOverflowFlagClearOne /

cecOverflowFlagClearMulti /

cecOverflowFlagClearAll

- 대상 카운터 채널의 Overflow 상태 Clear

INFORMATION	
1	Counter
!	VC++ (6, 7, 8)/VB
	BCB/Delphi
:	Level 7
J	위험 요소 없음

SYNOPSIS

r VT_I4 cecOverflowFlagClearOne ([in] VT_I4 Channel)

r VT_I4 cecOverflowFlagClearMulti ([in] VT_I4 IniChannel, [in] VT_I4 NumChannels)

r VT_I4 cecOverflowFlagClearAll ([in] VT_I4 NodeID)

DESCRIPTION

cecOverflowFlagClearOne 함수는 대상 카운터 채널의 카운트 Overflow 상태를 0 으로 Clear 합니다.

cecOverflowFlagClearMulti 함수는 대상 카운터 채널 범위에 포함된 채널들의 카운트 Overflow 상태를 0 으로 Clear 합니다.

cecOverflowFlagClearAll 함수는 대상 원격 노드 내의 모든 카운터 채널 카운트 Overflow 상태를 0 으로 Clear 합니다.

PARAMETER

▶ *Channel* : cecOverflowFlagClearOne 함수의 인자로서 채널 번호. 통합 채널로 관리되는 채널 번호를 의미하며, 상수 값으로 0 (Zero Based) 이상, 최대 통합 채널수 - 1 이하의 값을 채널 번호로 설정할 수 있습니다.

▶ *IniChannel* : cecOverflowFlagClearMulti 함수의 인자로서 시작 채널 번호. 채널 번호는 상수 값으로 0 번째 채널을 기준 채널로 임의의 채널을 설정 할 수 있습니다.

▶ *NumChannels* : cecOverflowFlagClearMulti 함수의 인자로서 시작 채널로부터 몇 개의 채널을 대상 채널로 할 것인지에 대한 값을 전달합니다. 최대 32 개 채널까지 설정 가능합니다.

▶ *NodeID* : cecOverflowFlagClearAll 함수의 인자로서 대상 카운터 모듈이 속해 있는 원격 노드 ID 를 설정합니다.

RETURN VALUE

Value	Meaning
음수	수행 실패. 자세한 내용은 '에러처리' 편을 참고합니다.
0 (ceERR_NONE)	수행 성공.

EXAMPLE

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

NAME

cecFilterFreq_Set / cecFilterFreq_Get

- 대상 카운터 채널 별 필터 Cutoff 주파수 모드 설정 및 설정 상태 반환

INFORMATION	
1	Counter
!	VC++ (6, 7, 8)/VB
	BCB/Delphi
:	Level 7
J	위험 요소 없음

SYNOPSIS

r VT_I4 cecFilterFreq_Set ([in] VT_I4 Channel, [in] VT_I4 FilterFreq)

r VT_I4 cecFilterFreq_Get ([in] VT_I4 Channel, [out] VT_PI4 FilterFreq)

DESCRIPTION

cecFilterFreq_Set 함수는 대상 카운터 채널 별 필터 Cutoff 주파수 모드를 설정합니다.

cecFilterFreq_Get 함수는 대상 카운터 채널 별 필터 Cutoff 주파수 모드 설정 상태를 반환합니다.

PARAMETER

▶ *Channel* : 채널 번호. 통합 채널로 관리되는 채널 번호를 의미하며, 상수 값으로 0 (Zero Based) 이상, 최대 통합 채널수 - 1 이하의 값을 채널 번호로 설정할 수 있습니다.

▶ *FilterFreq* : 대상 카운터 채널 별 필터 Cutoff 주파수 모드 설정 및 설정 상태를 반환합니다.

Value	필터 주파수	Cutoff 주파수 (50 Duty)	Filter 를 Pass 하기 위한 최소 시간
0 [Default]	10 MHz	500 KHz	1 usec
1	312 KHz	20 KHz	25 usec
2	39 KHz	4 KHz	125 usec
3	4.88 KHz	500 Hz	1 msec

RETURN VALUE

Value	Meaning
음수	수행 실패. 자세한 내용은 '에러처리' 편을 참고합니다
0 (ceERR_NONE)	수행 성공

EXAMPLE

C/C++

```
#include “ceSDK.h”
#include “ceSDKDef.h”

long nCntChNo = 1;              // 카운터 채널 번호 설정
long nCutoffMode;               // 필터 Cutoff 주파수 모드 정보

// 필터 Cutoff 주파수 모드를 확인 후, Cutoff 주파수를 500 KHz(모드 0) 로 설정합니다.

if ( cecFilterFreq_Get ( nCntChNo, &nCutoffMode ) == ceERR_NONE )
{
    if ( nCutoffMode != 0 )
    {
            cecFilterFreq_Set ( nCntChNo, 0 )
    }
}
```

Visual Basic

```
Dim nCntChNo As Long        ‘ 카운터 채널 번호 설정
Dim nCutoffMode As Long     ‘ 필터 Cutoff 주파수 모드 정보

nCntChNo = 1

‘ 필터 Cutoff 주파수 모드를 확인 후, Cutoff 주파수를 500 KHz(모드 0) 로 설정합니다.

If cecFilterFreq_Get ( nCntChNo, nCutoffMode ) = ceERR_NONE Then

    If nCutoffMode <> 0 Then

            Call cecFilterFreq_Set ( nCntChNo, 0 )
    End If

End If
```

Delphi

```
var
    nCntChNo : LongInt;         // 카운터 채널 번호 설정
    nCutoffMode : LongInt;      // 필터 Cutoff 주파수 모드 정보

begin
    nCntChNo := 1;

    // 필터 Cutoff 주파수 모드를 확인 후, Cutoff 주파수를 500 KHz(모드 0) 로 설정합니다.

    if cecFilterFreq_Get ( nCntChNo, @nCutoffMode ) = ceERR_NONE then
    begin
```

```
            if nCutoffMode <> 0 then
            begin
                cecFilterFreq_Set ( nCntChNo, 0 );
            end;
    end;
end;
```

Chapter

14

Analog Input/Output Control

다양한 기능의 아날로그 입력과 출력기능은 고객 여러분들의 장비 구성에 편의를 더하고 있습니다. 또한 과전류로 인한 제품의 손상을 방지할 수 있는 Over-current shutdown 기능은 시스템의 전체적인 안정성을 향상시킵니다.

커미조아의 A/D 모듈은 ±10V 의 전압 입력 및 0~20mA 전류 입력 기능을 제공하여, 사용자가 선택적으로 사용할 수 있으며, D/A 모듈은 하나의 모듈에서 아날로그 출력 채널을 통해 ±10V 전압 출력과 4~20mA 전류 출력 기능을 제공하여 사용자가 선택적으로 사용할 수 있도록 함으로써 장비의 구성에 편의를 더하고 있습니다

14 아날로그 입·출력 편

이 단원에서는 아날로그 입력(Analog Input), 아날로그 출력(Analog Output) 에 관련된 함수들을 소개합니다.

14.1 아날로그 입력 (Analog Input)

㈜ 커미조아의 A/D 모듈은 ±10V 의 전압 입력 및 0~20mA 전류 입력 기능을 제공하여, 사용자가 선택적으로 사용할 수 있도록 함으로써 장비의 구성에 편의를 더하고 있습니다.

또한 외부로 제공되는 24V 전원에 대해 Over-current shutdown 기능을 제공합니다. 이 기능은 외부 배선 시 전원이 단락 되는 등의 과전류 사용 시 5A 이상의 전류를 사용할 경우 자동으로 전원을 suutdown 하고, 과전류로 인한 제품의 손상을 방지합니다.

14.1.1 함수 요약

아날로그 입력(A/D) 기능에 관련된 함수는 다음과 같습니다.

Summary of Functions
r VT_I4 ceaiVoltRangeMode_Set ([in] VT_I4 Channel, [in] VT_I4 RangeMode) 아날로그 입력에 대한 전압 범위를 지정된 모드를 통해 설정합니다.
r VT_I4 ceaiVoltRangeMode_Get ([in] VT_I4 Channel, [out] VT_PI4 RangeMode) 아날로그 입력에 대해 설정된 전압 범위에 해당하는 모드를 반환합니다.
r VT_I4 ceaiRangeDigit_Get ([in] VT_I4 Channel, [out] VT_PI4 DigitMin, [out] VT_PI4 DigitMax) 아날로그 입력에 대해 설정된 입력 범위를 Digit 값으로 반환합니다.
r VT_I4 ceaiDigit_Get ([in] VT_I4 Channel, [out] VT_PI4 Digit) 대상 아날로그 입력 채널에 대하여 A/D 결과를 Digit 값으로 반환합니다.
r VT_I4 ceaiVolt_Get ([in] VT_I4 Channel, [out] VT_PR8 fVolt) 대상 아날로그 입력 채널에 대하여 A/D 결과를 전압(Volt) 값으로 반환합니다.
r VT_I4 ceaiCurrent_Get ([in] VT_I4 Channel, [out] VT_PR8 fCurrent) 대상 아날로그 입력 채널에 대하여 A/D 결과를 전류(Current)값으로 반환합니다.

14.1.2 함수 설명

NAME

ceaiVoltRangeMode_Set /

ceaiVoltRangeMode_Get

- 대상 아날로그 입력 채널의 전압(Volt) 범위 설정 및 설정 상태 반환

INFORMATION	
1	Analog Input Control
!	VC++ (6, 7, 8)/VB
	BCB/Delphi
:	Level 1
J	위험 요소 없음

SYNOPSIS

r VT_I4 ceaiVoltRangeMode_Set ([in] VT_I4 Channel, [in] VT_I4 RangeMode)

r VT_I4 ceaiVoltRangeMode_Get ([in] VT_I4 Channel, [out] VT_PI4 RangeMode)

DESCRIPTION

ceaiVoltRangeMode_Set 함수는 아날로그 입력 채널의 전압 범위를 지정된 모드를 통해 설정합니다.

ceaiVoltRangeMode_Get 함수는 설정된 전압 범위에 해당하는 모드를 반환 합니다.

PARAMETER

▶ ***Channel*** **:** 채널 번호. 통합 채널로 관리되는 채널 번호를 의미하며, 상수 값으로 0 (Zero Based) 이상, (최대 통합 채널수 - 1) 이하의 값을 채널 번호로 설정할 수 있습니다.

▶ ***RangeMode*** **:** 전압 범위 설정 모드를 설정 혹은 설정 상태를 반환합니다. 전압 범위 설정 모드는 다음과 같습니다.

Value	Meaning
0 [Default]	-10V ~ 10V
1	-5V ~ 5V
2	-2.5V ~ 2.5V
3	0V ~ 10V (0 ~ 20mA : 전류 모드 사용 시)
4	0V ~ 5V
5	1V ~ 5V
6	1V ~ 5V (4 ~ 20 mA : 전류 모드 사용 시)

0 ~ 6 번 모드 모두 전압 모드로 사용할 수 있으나, 전류 모드 사용 시 3 번 또는 6 번 모드로 설정 하셔야 합니다.

RETURN VALUE

Value	Meaning
음수	수행 실패. 자세한 내용은 '에러처리' 편을 참고합니다
0 (ceERR_NONE)	수행 성공

EXAMPLE

C/C++

```
#include “ceSDK.h”
#include “ceSDKDef.h”

long nAiChNo = 1;              // 아날로그 채널 번호 설정
long nRangeMode;               // 전압 범위 설정 모드 정보
long nDigitMin, nDigitMax;     // 입력 전압 범위에 대한 Digit 정보

// 전압 범위 설정 모드를 확인 후, 전압 범위 설정 모드를 3 번 모드( 0~10V, 0~20mA) 로 설정합니다.
if ( ceaiVoltRangeMode_Get ( nAiChNo, &nRangeMode ) == ceERR_NONE )
{
    if ( nRangeMode != 3 )
    {
            ceaiVoltRangeMode_Set ( nAiChNo, 3 );
    }
}

/* 설정된 전압 범위를 Digit 값으로 확인 합니다.
0 ~ 4 번 모드의 전압 범위는 nDigitMin : 0, nDigitMax : 8192 이며
5 ~ 6 번 모드의 전압 범위는 nDigitMin : -2048, nDigitMax : 8192 입니다. */
if ( ceaiRangeDigit_Get ( nAiChNo, &nDigitMin, &nDigitMax ) == ceERR_NONE )
{
    if ( nDigitMin != 0 || nDigitMax != 8192 )
    {
            OutputDebugString ( “ceaiRangeDigit_Get has been failed” );
    }
}
```

Visual Basic

```
Dim nAiChNo As Long                     ‘ 아날로그 채널 번호 설정
Dim nRangeMode As Long                  ‘ 전압 범위 설정 모드 정보
Dim nDigitMin As Long, nDigitMax As Long    ‘ 전압 범위에 해당하는 Digit 값 정보

nAiChNo = 1                             ‘ 아날로그 채널 번호 설정

‘ 전압 범위 설정 모드를 확인 후, 전압 범위 설정 모드를 3 번 모드( 0~10V, 0~20mA) 로 설정합니다.
If ceaiVoltRangeMode_Get ( nAiChNo, nRangeMode ) = ceERR_NONE Then

    If nRangeMode <> 3 Then
            Call ceaiVoltRangeMode_Set ( nAiChNo, 3 )
    End If

End If

‘ 설정된 전압 범위를 Digit 값으로 확인 합니다.
‘ 0 ~ 4 번 모드의 전압 범위는 nDigitMin : 0, nDigitMax : 8192 이며
‘ 5 ~ 6 번 모드의 전압 범위는 nDigitMin : -2048, nDigitMax : 8192 입니다.
If ceaiRangeDigit_Get ( nAiChNo, nDigitMin, nDigitMax ) = ceERR_NONE Then
```

```
    If ( nDigitMin <> 0 ) Or ( nDigitMax <> 8192 ) Then
            MsgBox ( “ceaiRangeDigit_Get has been failed” )
    End If
End If
```

Delphi

```
var
    nAiChNo : LongInt;                  // 아날로그 채널 번호 설정
    nRangeMode : LongInt;               // 전압 범위 설정 모드 정보
    nDigitMin, nDigitMax : LongInt;     // 전압 범위에 해당하는 Digit 값 정보

begin
    nAiChNo := 1;               // 아날로그 채널 번호 설정

    // 전압 범위 설정 모드를 확인 후, 전압 범위 설정 모드를 3 번 모드( 0~10V, 0~20mA) 로 설정합니다.
    if ceaiVoltRangeMode_Get ( nAiChNo, @nRangeMode ) = ceERR_NONE then
    begin
            if nRangeMode <> 3 then
            begin
                    ceaiVoltRangeMode_Set ( nAiChNo, 3 );
            end;
    end;


    { 설정된 전압 범위를 Digit 값으로 확인 합니다.
     0 ~ 4 번 모드의 전압 범위는 nDigitMin : 0, nDigitMax : 8192 이며
     5 ~ 6 번 모드의 전압 범위는 nDigitMin : -2048, nDigitMax : 8192 입니다. }
    if ceaiRangeDigit_Get ( nAiChNo, @nDigitMin, @nDigitMax ) = ceERR_NONE then
    begin
            if ( nDigitMin <> 0 )  or ( nDigitMax <> 8192 ) then
            begin
                    ShowMessage ( ‘ceaiRangeDigit_Get has been failed’ );
            end;
    end;
end;
```

NAME

ceaiRangeDigit_Get

- 대상 아날로그 입력 채널의 설정된 입력 범위를 Digit 값으로 반환

INFORMATION	
1	Analog Input Control
!	VC++ (6, 7, 8)/VB
	BCB/Delphi
:	Level 1
J	위험 요소 없음

SYNOPSIS

r VT_I4 ceaiRangeDigit_Get ([in] VT_I4 Channel, [out] VT_PI4 DigitMin, [out] VT_PI4 DigitMax)

DESCRIPTION

대상 아날로그 입력 채널에 설정된 입력 Range (전압 범위)를 Digit 값으로 반환합니다.
입력 Range 설정은 ceaiVoltRangeMode_Set 함수를 통해 설정합니다.

PARAMETER

▶ ***Channel*** : 채널 번호. 통합 채널로 관리되는 채널 번호를 의미하며, 상수 값으로 0 (Zero Based) 이상, (최대 통합 채널수 - 1) 이하의 값을 채널 번호로 설정할 수 있습니다.

▶ ***DigitMin*** : 이 매개변수를 통하여 입력 Range 의 최소 입력 Digit 값을 반환합니다.

▶ ***DigitMax*** : 이 매개변수를 통하여 입력 Range 의 최대 입력 Digit 값을 반환합니다.

입력되는 Digit 값의 범위는 어떻게 되나요?

ceAI08A 모듈의 Digit 범위는 0 ~ 8192 (13 Bit) 입니다.

RETURN VALUE

Value	Meaning
음수	수행 실패. 자세한 내용은 '에러처리' 편을 참고합니다
0 (ceERR_NONE)	수행 성공

EXAMPLE

```
//* ceaiVoltRangeMode_Set / ceaiVoltRangeMode_Get 예제를 참고하여 주시기 바랍니다.
```

NAME

ceaiDigit_Get

- 대상 아날로그 입력 채널의 입력 Digit 값을 반환

INFORMATION	
1	Analog Input Control
!	VC++ (6, 7, 8)/VB
	BCB/Delphi
:	Level 1
J	위험 요소 없음

SYNOPSIS

r VT_I4 ceaiDigit_Get ([in] VT_I4 Channel, [out] VT_PI4 Digit)

DESCRIPTION

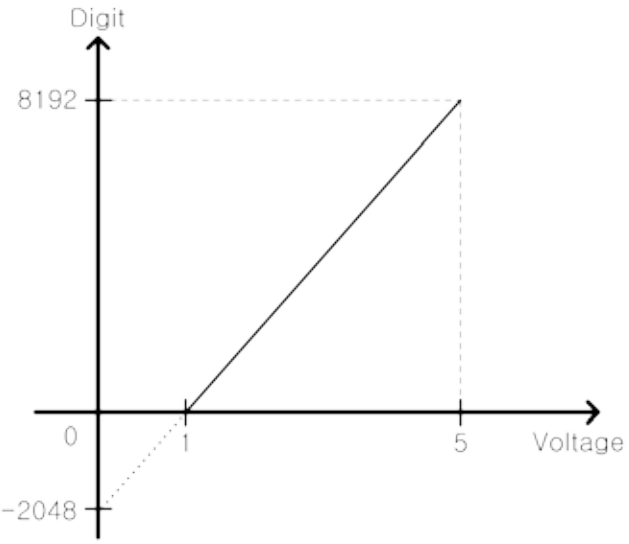
대상 아날로그 입력 채널에 대하여 A/D 변환을 수행하고, 그 값을 Digit 값으로 반환합니다.

PARAMETER

▶ ***Channel*** : 채널 번호. 통합 채널로 관리되는 채널 번호를 의미하며, 상수 값으로 0 (Zero Based) 이상, (최대 통합 채널수 - 1) 이하의 값을 채널 번호로 설정할 수 있습니다.

▶ ***Digit*** : 이 매개변수를 통하여 A/D 결과값을 Digit 값으로 반환합니다. 13Bit Straight Binery (0~8192)로 구성되어 있습니다.

단, 전압 입력 범위(Range) 모드가 5 번 모드(1~5V), 6 번 모드(1~5V, 4~20mA) 모드인 경우는 입력 전압이 1V 보다 낮은 경우, Digit 값으로 구분해서 표시하기 위해 Digit 값과 Voltage 값의 관계 직선에서 0~1V 에 해당되는 음의 Digit 값을 선형적으로 계산해서 반환합니다.



RETURN VALUE

Value	Meaning
음수	수행 실패. 자세한 내용은 '에러처리' 편을 참고합니다.
0 (ceERR_NONE)	수행 성공.

EXAMPLE

C/C++

```
#include “ceSDK.h”
#include “ceSDKDef.h”

long nAiChNo = 1;              // 아날로그 입력 채널
long nDigitVal;                // A/D 결과를 Digit 값으로 반환

// 전압 입력 모드를 3 번 (0~10V, 0 ~ 20mA) 로 설정
ceaiVoltRangeMode_Set ( nAiChNo, 3 );

// A/D 결과값을 Digit 값으로 반환합니다.
if ( ceaiDigit_Get ( nAiChNo, &nDigitVal ) != ceERR_NONE )
{
    OutputDebugString ( "ceaiDigit_Get has been failed" );
}
```

Visual Basic

```
Dim nAiChNo As Long        ' 아날로그 입력 채널
Dim nDigitVal As Long      ' A/D 결과를 Digit 값으로 반환

nAiChNo = 1

' 전압 입력 모드를 3 번 (0~10V, 0 ~ 20mA) 로 설정
Call ceaiVoltRangeMode_Set ( nAiChNo, 3 )

' A/D 결과값을 Digit 값으로 반환합니다.
If ceaiDigit_Get ( nAiChNo, nDigitVal ) <> ceERR_NONE Then

    MsgBox ( "ceaiDigit_Get has been failed" )
End If
```

Delphi

```
var
    nAiChNo : LongInt          // 아날로그 입력 채널
    nDigitVal : LongInt        // A/D 결과를 Digit 값으로 반환

begin
    nAiChNo := 1;
```

```
    // 전압 입력 모드를 3 번 (0~10V, 0 ~ 20mA) 로 설정
    ceaiVoltRangeMode_Set ( nAiChNo, 3 );

    // A/D 결과값을 Digit 값으로 반환합니다.
    if ceaiDigit_Get ( nAiChNo, @nDigitVal ) <> ceERR_NONE then
    begin
            ShowMessage ( 'ceaiDigit_Get has been failed' );
    end;
end;
```

NAME

ceaiVolt_Get

- 대상 아날로그 입력 채널의 입력 전압(Volt) 값을 반환

INFORMATION	
1	Analog Input Control
!	VC++ (6, 7, 8)/VB
	BCB/Delphi
:	Level 1
J	위험 요소 없음

SYNOPSIS

r VT_I4 ceaiVolt_Get ([in] VT_I4 Channel, [out] VT_PR8 fVolt)

DESCRIPTION

대상 아날로그 입력 채널에 대하여 A/D 변환을 수행하고, 그 값을 전압(Volt) 값으로 반환합니다.

PARAMETER

▶ ***Channel*** **:** 채널 번호. 통합 채널로 관리되는 채널 번호를 의미하며, 상수 값으로 0 (Zero Based) 이상, (최대 통합 채널수 - 1) 이하의 값을 채널 번호로 설정할 수 있습니다.

▶ ***fVolt*** **:** 이 매개변수를 통하여 A/D 결과값을 전압(Volt) 값으로 반환합니다.

RETURN VALUE

Value	Meaning
음수	수행 실패. 자세한 내용은 '에러처리' 편을 참고합니다.
0 (ceERR_NONE)	수행 성공.

EXAMPLE

C/C++

```
#include “ceSDK.h”
#include “ceSDKDef.h”

long nAiChNo = 1;              // 아날로그 입력 채널
double fVoltVal;               // A/D 결과를 전압 값으로 반환

// 전압 입력 모드를 3 번 (0~10V, 0 ~ 20mA) 로 설정
ceaiVoltRangeMode_Set ( nAiChNo, 3 );

// A/D 결과값을 전압 값으로 반환합니다.
if ( ceaiVolt_Get ( nAiChNo, &fVoltVal ) != ceERR_NONE )
{
    OutputDebugString ( "ceaiVolt_Get has been failed" );
}
```

Visual Basic

```
Dim nAiChNo As Long          ‘ 아날로그 입력 채널
Dim nVoltVal As Double       ‘ A/D 결과를 전압 값으로 반환

nAiChNo = 1

‘ 전압 입력 모드를 3 번 (0~10V, 0 ~ 20mA) 로 설정
Call ceaiVoltRangeMode_Set ( nAiChNo, 3 )

‘ A/D 결과값을 전압 값으로 반환합니다.
If ceaiVolt_Get ( nAiChNo, fVoltVal ) <> ceERR_NONE Then

    MsgBox ( "ceaiVolt_Get has been failed" )
End If
```

Delphi

```
var
    nAiChNo : LongInt;         // 아날로그 입력 채널
    nVoltVal : Double;         // A/D 결과를 전압 값으로 반환

begin
    nAiChNo := 1;

    // 전압 입력 모드를 3 번 (0~10V, 0 ~ 20mA) 로 설정
    ceaiVoltRangeMode_Set ( nAiChNo, 3 );

    // A/D 결과값을 전압 값으로 반환합니다.
    if ceaiVolt_Get ( nAiChNo, @fVoltVal ) <> ceERR_NONE then
    begin
            ShowMessage ( ‘ceaiVolt_Get has been failed’ );
    end;
end;
```

NAME

ceaiCurrent_Get

- 대상 아날로그 입력 채널의 입력 전류(Current) 값을 반환

INFORMATION	
1	Analog Input Control
!	VC++ (6, 7, 8)/VB BCB/Delphi
:	Level 1
J	위험 요소 없음

SYNOPSIS

r VT_I4 ceaiCurrent_Get ([in] VT_I4 Channel, [out] VT_PR8 fCurrent)

DESCRIPTION

대상 아날로그 입력 채널에 대하여 A/D 변환을 수행하고, 그 값을 전류(Current) 값으로 반환합니다.

PARAMETER

▶ ***Channel*** : 채널 번호. 통합 채널로 관리되는 채널 번호를 의미하며, 상수 값으로 0 (Zero Based) 이상, (최대 통합 채널수 - 1) 이하의 값을 채널 번호로 설정할 수 있습니다.

▶ ***fCurrent*** : A/D 결과값을 전류(Current) 값으로 반환합니다. 단, 전압 범위 설정 모드를 3 번 (0~20mA) 또는 6 번 (4~20mA) 로 설정해야 합니다.

Value	Meaning
-1	전류 입력 모드(3 번, 6 번 모드) 가 아닌 경우의 전달되는 값.
유효한 값	3 번 모드 : 0~20 (mA) 6 번 모드 : 4~20 (mA)

RETURN VALUE

Value	Meaning
음수	수행 실패. 자세한 내용은 '에러처리' 편을 참고합니다.
0 (ceERR_NONE)	수행 성공.

EXAMPLE

C/C++

```
#include “ceSDK.h”
#include “ceSDKDef.h”

long nAiChNo = 1;              // 아날로그 입력 채널
double fCurrentVal;            // A/D 결과를 전류 값으로 반환

// A/D 결과값을 전류 값으로 반환합니다.
if ( ceaiCurrent_Get ( nAiChNo, &fCurrentVal ) == ceERR_NONE )
{
    if ( fnCurrentVal == -1 )     // 전압 범위 입력 모드가 전류 모드가 아닙니다.
    {
            // 전압 입력 모드를 3 번 (0~10V, 0 ~ 20mA) 로 설정합니다.
            ceaiVoltRangeMode_Set ( nAiChNo, 3 );
    }
}
```

Visual Basic

```
Dim nAiChNo As Long          ‘ 아날로그 입력 채널
Dim fCurrentVal As Double    ‘ A/D 결과를 전류 값으로 반환

nAiChNo = 1

‘ A/D 결과값을 전류 값으로 반환합니다.
If ceaiVolt_Get ( nAiChNo, fCurrentVal ) = ceERR_NONE Then

    If fCurrentVal = -1 Then     ‘ 전압 범위 입력 모드가 전류 모드가 아닙니다.

            ‘ 전압 입력 모드를 3 번 (0~10V, 0 ~ 20mA) 로 설정
            Call ceaiVoltRangeMode_Set ( nAiChNo, 3 )
    End If
End If
```

Delphi

```
var
    nAiChNo : LongInt;          // 아날로그 입력 채널
    fCurrentVal : Double;       // A/D 결과를 전류 값으로 반환

begin
    nAiChNo := 1;

    // A/D 결과값을 전류 값으로 반환합니다.
    if ceaiVolt_Get ( nAiChNo, @fCurrentVal ) = ceERR_NONE then
    begin
```

```
            if fCurrentVal = -1 then      // 전압 범위 입력 모드가 전류 모드가 아닙니다.
            begin
                    ' 전압 입력 모드를 3 번 (0~10V, 0 ~ 20mA) 로 설정
                    ceaiVoltRangeMode_Set ( nAiChNo, 3 );
            end;
    end;

end;
```

14.2 아날로그 출력 (Analog Output)

㈜ 커미조아의 D/A 모듈은 하나의 모듈에서 아날로그 출력 채널을 통해 ±10V 전압 출력과 4~20mA 전류 출력 기능을 제공하여 사용자가 선택적으로 사용할 수 있도록 함으로써 장비의 구성에 편의를 더하고 있습니다.

또한 외부로 제공되는 24V 전원에 대해 Over-current shutdown 기능을 제공합니다. 이 기능은 외부 배선 시 전원이 단락 되는 등의 과전류 사용 시 5A 이상의 전류를 사용할 경우 자동으로 전원을 suutdown 하고, 과전류로 인한 제품의 손상을 방지합니다.

14.2.1 함수 요약

아날로그 출력(D/A) 기능에 관련된 함수는 다음과 같습니다.

Summary of Functions
r VT_I4 ceaoDigit_Out ([in] VT_I4 Channel, [in] VT_I4 Digit) 대상 아날로그 출력 채널을 통해 Digit 값을 출력합니다.
r VT_I4 ceaoVolt_Out ([in] VT_I4 Channel, [in] VT_R8 fVolt) 대상 아날로그 출력 채널을 통해 전압(Volt) 값을 출력합니다.
r VT_I4 ceaoCurrent_Out ([in] VT_I4 Channel, [in] VT_R8 fCurrent) 대상 아날로그 출력 채널을 통해 전류(Current) 값을 출력합니다.

14.2.2 함수 설명

NAME

ceaoDigit_Out

- 대상 아날로그 출력 채널을 통해 Digit 값 출력

INFORMATION	
1	Analog Output Control
!	VC++ (6, 7, 8)/VB BCB/Delphi
:	Level 1
J	위험 요소 없음

SYNOPSIS

r VT_I4 ceaoDigit_Out ([in] VT_I4 Channel, [in] VT_I4 Digit)

DESCRIPTION

대상 아날로그 출력 채널에 대하여 지정한 Digit 값을 출력합니다.

PARAMETER

▶ ***Channel*** **:** 채널 번호. 통합 채널로 관리되는 채널 번호를 의미하며, 상수 값으로 0 (Zero Based) 이상, (최대 통합 채널수 - 1) 이하의 값을 채널 번호로 설정할 수 있습니다.

▶ ***Digit*** **:** 매개변수를 통하여 대상 출력 채널을 통해 Digit 값을 출력합니다. D/A 데이터는 부호를 갖는 16Bit Data (-32768 ~ 32767)로 출력할 수 있습니다.

RETURN VALUE

Value	Meaning
음수	수행 실패. 자세한 내용은 '에러처리' 편을 참고합니다.
0 (ceERR_NONE)	수행 성공.

EXAMPLE

C/C++

```
#include “ceSDK.h”
#include “ceSDKDef.h”

long nAoChNo = 1;                // 아날로그 출력 채널

// 아날로그 출력 채널에 10V 를 출력 합니다.
if ( ceaoDigit_Out ( nAoChNo, 32767 ) != ceERR_NONE )
{
    OutputDebugString ( "ceaoDigit_Out has been failed" );
}
```

Visual Basic

```
Dim nAoChNo As Long          ‘ 아날로그 출력 채널
nAoChNo = 1

‘ 아날로그 출력 채널에 10V 를 출력 합니다.
If ceaoDigit_Out ( nAoChNo, 32767 ) <> ceERR_NONE Then

    MsgBox ( "ceaoDigit_Out has been failed" )
End If
```

Delphi

```
var
    nAoChNo : LongInt;          // 아날로그 출력 채널

begin
    nAoChNo := 1;

    // 아날로그 출력 채널에 10V 를 출력 합니다.
    if ceaoDigit_Out ( nAoChNo, 32767 ) <> ceERR_NONE then
    begin
            ShowMessage ( ‘ceaoDigit_Out has been failed’ );
    end;
end;
```

NAME

ceaoVolt_Out

- 대상 아날로그 출력 채널을 통해 전압(Volt) 값 출력

INFORMATION	
1	Analog Output Control
!	VC++ (6, 7, 8)/VB
	BCB/Delphi
:	Level 1
J	위험 요소 없음

SYNOPSIS

r VT_I4 ceaoVolt_Out ([in] VT_I4 Channel, [in] VT_PR8 fVolt)

DESCRIPTION

대상 아날로그 출력 채널에 대하여 지정한 전압(Volt) 값을 출력합니다.

PARAMETER

▶ ***Channel*** : 채널 번호. 통합 채널로 관리되는 채널 번호를 의미하며, 상수 값으로 0 (Zero Based) 이상, (최대 통합 채널수 - 1) 이하의 값을 채널 번호로 설정할 수 있습니다.

▶ ***fVolt*** : 이 매개변수를 통하여 대상 출력 채널을 통해 전압(Volt) 값을 출력합니다. 출력 가능한 전압 값의 범위는 -10V ~ 10V 입니다.

RETURN VALUE

Value	Meaning
음수	수행 실패. 자세한 내용은 '에러처리' 편을 참고합니다.
0 (ceERR_NONE)	수행 성공.

EXAMPLE

C/C++

```
#include “ceSDK.h”
#include “ceSDKDef.h”

long nAoChNo = 1;                // 아날로그 출력 채널

// 아날로그 출력 채널에 10V 를 출력 합니다.
if ( ceaoVolt_Out ( nAoChNo, 10 ) != ceERR_NONE )
{
    OutputDebugString ( "ceaoVolt_Out has been failed" );
}
```

Visual Basic

```
Dim nAoChNo As Long          ‘ 아날로그 출력 채널
nAoChNo = 1

‘ 아날로그 출력 채널에 10V 를 출력 합니다.
If ceaoVolt_Out ( nAoChNo, 10 ) <> ceERR_NONE Then

    MsgBox ( "ceaoVolt_Out has been failed" )
End If
```

Delphi

```
var
    nAoChNo : LongInt;          // 아날로그 출력 채널

begin
    nAoChNo := 1;

    // 아날로그 출력 채널에 10V 를 출력 합니다.
    if ceaoVolt_Out ( nAoChNo, 10 ) <> ceERR_NONE then
    begin
            ShowMessage ( ‘ceaoVolt_Out has been failed’ );
    end;
end;
```

NAME

ceaoCurrent_Out

- 대상 아날로그 출력 채널을 통해 전류(Current) 값 출력

INFORMATION	
1	Analog Output Control
!	VC++ (6, 7, 8)/VB
	BCB/Delphi
:	Level 1
J	위험 요소 없음

SYNOPSIS

r VT_I4 ceaoCurrent_Out ([in] VT_I4 Channel, [out] VT_PR8 fCurrent)

DESCRIPTION

대상 아날로그 출력 채널에 대하여 지정한 전류(Current) 값을 출력합니다.

PARAMETER

▶ ***Channel*** **:** 채널 번호. 통합 채널로 관리되는 채널 번호를 의미하며, 상수 값으로 0 (Zero Based) 이상, (최대 통합 채널수 - 1) 이하의 값을 채널 번호로 설정할 수 있습니다.

▶ ***fCurrent*** **:** 이 매개변수를 통하여 대상 출력 채널을 통해 전류(Current) 값을 출력합니다. 출력 가능한 전류 값의 범위는 4mA ~ 20mA 입니다.

RETURN VALUE

Value	Meaning
음수	수행 실패. 자세한 내용은 '에러처리' 편을 참고합니다.
0 (ceERR_NONE)	수행 성공.

EXAMPLE

C/C++

```
#include “ceSDK.h”
#include “ceSDKDef.h”

long nAoChNo = 1;                // 아날로그 출력 채널

// 아날로그 출력 채널에 20mA 를 출력 합니다.
if ( ceaoCurrent_Out ( nAoChNo, 10 ) != ceERR_NONE )
{
    OutputDebugString ( "ceaoCurrent_Out has been failed" );
}
```

Visual Basic

```
Dim nAoChNo As Long          ‘ 아날로그 출력 채널
nAoChNo = 1

‘ 아날로그 출력 채널에 20mA 를 출력 합니다.
If ceaoCurrent_Out ( nAoChNo, 20 ) <> ceERR_NONE Then

    MsgBox ( "ceaoCurrent_Out has been failed" )
End If
```

Delphi

```
var
    nAoChNo : LongInt;          // 아날로그 출력 채널

begin
    nAoChNo := 1;

    // 아날로그 출력 채널에 20mA 를 출력 합니다.
    if ceaoCurrent_Out ( nAoChNo, 20 ) <> ceERR_NONE then
    begin
            ShowMessage ( ‘ceaoCurrent_Out has been failed’ );
    end;
end;
```

Chapter

15

SERIAL Functions

커미조아의 시리얼 통신 모듈의 다양한 시리얼 통신 방식 지원은 고객 여러분들의 장비 구성에 편의를 더하고 있습니다.

이 단원에서는 시리얼 통신과 관련된 함수를 소개합니다. ㈜ 커미조아의 시리얼 통신 모듈은 4 채널의 RS232 / RS422 / RS485 시리얼 통신 방식을 지원하며, 해당 채널의 기능을 선택적으로 사용할 수 있어 통신 종류에 구애받지 않고 모듈을 구성할 수 있습니다.

15 시리얼(Serial) 통신 편

이 단원에서는 시리얼 통신과 관련된 함수를 소개합니다. ㈜ 커미조아의 시리얼 통신 모듈은 4 채널의 RS232 / RS422 / RS485 시리얼 통신 방식을 지원하며, 해당 채널의 기능을 선택적으로 사용할 수 있어 통신 종류에 구애받지 않고 모듈을 구성할 수 있습니다.

또한 외부로 제공되는 24V 전원에 대해 Over-current shutdown 기능을 제공합니다. 이 기능은 외부 배선 시 전원이 단락 되는 등의 과전류 사용 시 5A 이상의 전류를 사용할 경우 자동으로 전원을 suutdown 하고, 과전류로 인한 제품의 손상을 방지합니다.

15.1 함수 요약

시리얼 통신에 관련된 함수는 다음과 같습니다.

Summary of Functions
r VT_I4 cesOpenPort ([in] VT_I4 Channel, [in] VT_I4 SerType, [in] VT_I4 BaudId, [in] VT_I4 DataBits, [in] VT_I4 StopBits, [in] VT_I4 Parity) 해당 채널(Port) 을 할당하고 시리얼 통신 설정을 통해 통신 포트 오픈(Open) 후 사용할 준비를 합니다.
r VT_I4 cesClosePort ([in] VT_I4 Channel) 해당 채널의 할당을 해제하고 시리얼 통신 연결을 닫습니다.
r VT_I4 cesIsDataReady ([in] VT_I4 Channel, [out] VT_PI4 IsReady) 해당 채널을 통해 데이터가 수신 되었는지 확인 합니다.
r VT_I4 cesSetTimeout ([in] VT_I4 Channel, [in] VT_I4 Timeout) 해당 채널의 시리얼 통신 타임아웃 시간을 밀리초(ms) 단위로 설정합니다.
r VT_I4 cesGetTimeout ([in] VT_I4 Channel, [out] VT_PI4 Timeout) 해당 채널의 설정된 시리얼 통신 타임아웃 시간을 반환합니다.
r VT_I4 cesRxReset ([in] VT_I4 Channel) 해당 채널의 수신 버퍼를 비웁니다. 수신 버퍼 카운트를 0 으로 Clear 시킵니다.
r VT_I4 cesTxReset ([in] VT_I4 Channel) 해당 채널의 송신 버퍼를 비웁니다. 송신 버퍼 카운트를 0 으로 Clear 시킵니다.
r VT_I4 cesGetUnreadDataSize ([in] VT_I4 Channel, [out] VT_PI4 DataSize) 해당 채널의 수신 버퍼로부터 읽지 않은 데이터 크기를 바이트 단위로 얻어옵니다.
r VT_I4 cesPeekByte ([in] VT_I4 Channel, [out] VT_PI4 byData) 해당 채널의 수신 버퍼로부터 1 바이트의 데이터를 읽습니다. 버퍼 카운트를 감소 시키지 않습니다.
r VT_I4 cesPeekByteEx ([in] VT_I4 Channel, [in] VT_I4 ByteIdx, [out] VT_PI4 byData) 해당 채널의 수신 버퍼로부터 원하는 위치의 1 바이트의 데이터를 읽습니다. 버퍼 카운트를 감소 시키지 않습니다.
r VT_I4 cesReadByte ([in] VT_I4 Channel, [out] VT_PI4 byData) 해당 채널의 수신 버퍼로부터 1 바이트 데이터를 읽습니다. 버퍼 카운트를 1 만큼 감소 시킵니다.

r VT_I4 cesWriteByte ([in] VT_I4 Channel, [in] VT_I4 byData) 해당 채널의 송신 버퍼에 전송을 위한 1 바이트 데이터를 추가합니다.
r VT_I4 cesPeekString ([in] VT_I4 nChannel, [in] VT_I4 NumBytes , [out] VT_PSTR String) 해당 채널의 수신 버퍼로부터 지정한 바이트 수 만큼의 데이터를 읽습니다. 버퍼 카운트를 감소 시키지 않습니다.
r VT_I4 cesReadString ([in] VT_I4 Channel, [in] VT_I4 NumBytes, [out] VT_PSTR String) 해당 채널의 수신 버퍼로부터 지정한 바이트 수 만큼의 데이터를 읽습니다. 버퍼 카운트를 지정 바이트 수 만큼 감소 시킵니다.
r VT_I4 cesWriteString ([in] VT_I4 Channel, [in] VT_I4 NumBytes, [in] VT_STR String) 해당 채널의 송신 버퍼에 지정한 바이트 수 만큼의 데이터를 추가합니다. 버퍼 카운트를 지정 바이트 수 만큼 증가 시킵니다.
r VT_I4 cesReadDword ([in] VT_I4 Channel, [out] VT_PI4 dwData) 해당 채널의 수신 버퍼로부터 4 바이트 데이터를 읽습니다. 버퍼 카운트를 4 만큼 감소 시킵니다.
r VT_I4 cesWriteDword ([in] VT_I4 Channel, [in] VT_I4 NumWords, [in] VT_PI4 dwData) 해당 채널의 송신 버퍼에 전송을 위한 4 바이트 데이터를 추가합니다. 버퍼 카운트를 4 만큼 증가 시킵니다.
r VT_I4 cesCommit ([in] VT_I4 nChannel) 해당 채널의 송신 버퍼의 내용을 시리얼 통신 칩의 전송 버퍼에 담아 전송 시킵니다.

15.2 함수 설명

NAME

cesOpenPort

- 시리얼 통신 포트(Port) 오픈

INFORMATION	
1	Serial
!	VC++ (6, 7, 8)/VB
	BCB/Delphi
:	Level 7
J	위험 요소 없음

SYNOPSIS

r VT_I4 cesOpenPort ([in] VT_I4 Channel, [in] VT_I4 SerType, [in] VT_I4 BaudId, [in] VT_I4 DataBits, [in] VT_I4 StopBits, [in] VT_I4 Parity)

DESCRIPTION

해당 채널(Port) 을 할당하고 시리얼 통신 설정을 통해 통신 포트 Open 후 사용할 준비를 합니다.

PARAMETER

▶ ***Channel*** : 채널 번호. 통합 채널로 관리되는 채널 번호를 의미하며, 상수 값으로 0 (Zero Based) 이상, (최대 통합 채널수 - 1) 이하의 값을 채널 번호로 설정할 수 있습니다.

▶ ***SerType*** : 시리얼 통신 모드. 지원되는 통신 모드는 다음과 같습니다.

Value	Meaning
0 [Default]	RS-232
1	RS-422
2	RS-485

▶ ***BaudId*** : 시리얼 통신시 사용할 기 정의된 Baud Rate 인식 번호.

Value	Meaning
0 (BAUD_2400)	Baud Rate 2400
1 (BAUD_4800)	Baud Rate 4800
2 (BAUD_9600)	Baud Rate 9600
3 (BAUD_14400)	Baud Rate 14400
4 (BAUD_19200)	Baud Rate 19200
5 (BAUD_38400)	Baud Rate 38400
6 (BAUD_57600)	Baud Rate 57600
7 (BAUD_115200)	Baud Rate 115200

▶ ***DataBits*** : Data Bits 에 대한 설정 값이며 5 ~ 8 의 값을 설정하실 수 있습니다.

▶ ***StopBits*** : Stop Bits 에 대한 설정 값이며 1 또는 2 의 값을 설정하실 수 있습니다.

▶ ***Parity*** : Parity Bits 에 대한 설정 값이며 0(패리티 없음), 1(홀수 패리티), 2(짝수 패리티) 의 값을 설정하실 수 있습니다.

RETURN VALUE

Value	Meaning
음수	수행 실패. 자세한 내용은 '에러처리' 편을 참고합니다.
0 (ceERR_NONE)	수행 성공.

EXAMPLE

C/C++

```
#include “ceSDK.h”
#include “ceSDKDef.h”

long nSerialChNo = 0;              // 시리얼 채널(Port) 번호
long nSerType = 0;                 // 시리얼 통신 모드. 0: RS232, 1: RS422, 2: RS485
long nBaudId = BAUD_9600;          // Baud Rate 설정. 0 ~ 7 번 모드
long nDataBits = 8;                // Data Bits 설정. 5 ~ 8
long nStopBits = 1;                // Stop Bits 설정. 1 or 2
long nParity = 1;                  // Parity Bits 설정. 0: None, 1: Odd, 2: Even

// 0 번 시리얼 포트를 Open 합니다.
if ( cesOpenPort ( nSerialChNo, nSerType, BAUD_9600, nDataBits, nStopBits, nParity ) != ceERR_NONE )
{
    OutputDebugString ( "cesOpenPort has been failed" );
}
```

Visual Basic

```
Dim nSerialChNo As Long     ‘ 시리얼 채널(Port) 번호
Dim nSerType As Long        ‘ 시리얼 통신 모드. 0: RS232, 1: RS422, 2: RS485
Dim nBaudId As Long         ‘ Baud Rate 설정. 0 ~ 7 번 모드
Dim nDataBits As Long       ‘ Data Bits 설정. 5 ~ 8
Dim nStopBits As Long       ‘ Stop Bits 설정. 1 or 2
Dim nParity As Long         ‘ Parity Bits 설정. 0: None, 1: Odd, 2: Even

nSerialChNo = 0
nSerType = 0
nBaudId = BAUD_9600
nDataBits = 8
nStopBits = 1
nParity = 1
```

```
‘ 0 번 시리얼 포트를 Open 합니다.
If cesOpenPort ( nSerialChNo, nSerType, BAUD_9600, nDataBits, nStopBits, nParity ) <> ceERR_NONE Then
    MsgBox ( "cesOpenPort has been failed" )
End If
```

Delphi

```
var
    nSerialChNo, nSerType, nBaudId, nDataBits, nStopBits, nParity : LongInt;

begin
    nSerialChNo := 0;           // 시리얼 채널(Port) 번호
    nSerType := 0;              // 시리얼 통신 모드. 0: RS232, 1: RS422, 2: RS485
    nBaudId := BAUD_9600;       // Baud Rate 설정. 0 ~ 7 번 모드
    nDataBits := 8;             // Data Bits 설정. 5 ~ 8
    nStopBits := 1;             // Stop Bits 설정. 1 or 2
    nParity := 1;               // Parity Bits 설정. 0: None, 1: Odd, 2: Even

    // 0 번 시리얼 포트를 Open 합니다.
    if cesOpenPort ( nSerialChNo, nSerType, BAUD_9600, nDataBits, nStopBits, nParity )
                        <> ceERR_NONE then
    begin
            ShowMessage ( ‘cesOpenPort has been failed’ );
    end;
end;
```

NAME

cesClosePort

- 시리얼 통신 포트 닫기

INFORMATION
1 Serial
! VC++ (6, 7, 8)/VB
BCB/Delphi
: Level 7
J 위험 요소 없음

SYNOPSIS

r VT_I4 cesClosePort ([in] VT_I4 Channel)

DESCRIPTION

해당 시리얼 채널(Port) 의 할당을 해제하고 시리얼 통신 연결을 닫습니다.

PARAMETER

▶ *Channel* : 채널 번호. 통합 채널로 관리되는 채널 번호를 의미하며, 상수 값으로 0 (Zero Based) 이상, (최대 통합 채널수 - 1) 이하의 값을 채널 번호로 설정할 수 있습니다.

RETURN VALUE

Value	Meaning
음수	수행 실패. 자세한 내용은 '에러처리' 편을 참고합니다.
0 (ceERR_NONE)	수행 성공.

EXAMPLE

C/C++

```
#include “ceSDK.h”
#include “ceSDKDef.h”

long nSerialChNo = 0;          // 시리얼 채널(Port) 번호

// 0 번 시리얼 포트를 Close 합니다.
if ( cesClosePort ( nSerialChNo ) != ceERR_NONE )
{
    OutputDebugString ( "cesClosePort has been failed" );
}
```

Visual Basic

```
Dim nSerialChNo As Long     ‘ 시리얼 채널(Port) 번호
nSerialChNo = 0

‘ 0 번 시리얼 포트를 Close 합니다.
If cesClosePort ( nSerialChNo ) <> ceERR_NONE Then

    MsgBox ( "cesClosePort has been failed" )
End If
```

Delphi

```
var
    nSerialChNo : LongInt;      // 시리얼 채널(Port) 번호

begin
    nSerialChNo := 0;

    // 0 번 시리얼 포트를 Close 합니다.
    if cesClosePort ( nSerialChNo ) <> ceERR_NONE then
    begin
            ShowMessage ( ‘cesClosePort has been failed’ );
    end;
end;
```

NAME

cesIsDataReady

- 데이터 수신 확인

INFORMATION
1 Serial
! VC++ (6, 7, 8)/VB
BCB/Delphi
: Level 7
J 위험 요소 없음

SYNOPSIS

r VT_I4 cesIsDataReady ([in] VT_I4 Channel, [out] VT_PI4 IsReady)

DESCRIPTION

해당 채널(Port) 를 통해 데이터가 수신 되었는지 확인 합니다.

PARAMETER

▶ *Channel* : 채널 번호. 통합 채널로 관리되는 채널 번호를 의미하며, 상수 값으로 0 (Zero Based) 이상, (최대 통합 채널수 - 1) 이하의 값을 채널 번호로 설정할 수 있습니다.

▶ *IsReady* : 이 매개변수를 통하여 데이터가 수신 되었는지 확인합니다.

Value	Meaning
0 (CE_FALSE)	수신 데이터 없음.
1 (CE_TRUE)	수신 데이터 준비됨.

RETURN VALUE

Value	Meaning
음수	수행 실패. 자세한 내용은 '에러처리' 편을 참고합니다
0 (ceERR_NONE)	수행 성공

EXAMPLE

C/C++

```
#include “ceSDK.h”
#include “ceSDKDef.h”

long nSerialChNo = 0;          // 시리얼 채널(Port) 번호
long nIsReady;                 // 수신 데이터 정보

// 0 번 시리얼 포트에 데이터가 수신되었는지 확인합니다.
if ( cesIsDataReady ( nSerialChNo, &nIsReady ) != ceERR_NONE )
{
    if ( nIsReady == CE_TRUE )
    {
            OutputDebugString ( "데이터 수신" );
    }
}
```

Visual Basic

```
Dim nSerialChNo As Long     ‘ 시리얼 채널(Port) 번호
Dim nIsReady As Long        ‘ 수신 데이터 정보

nSerialChNo = 0             ‘ 시리얼 채널(Port) 번호

‘ 0 번 시리얼 포트에 데이터가 수신되었는지 확인합니다.
If cesIsDataReady ( nSerialChNo, nIsReady ) <> ceERR_NONE Then

    If nIsReady = CE_TRUE Then
            MsgBox ( "데이터 수신" )
    End If
End If
```

Delphi

```
var
    nSerialChNo : LongInt;      // 시리얼 채널(Port) 번호
    nIsReady : LongInt;         // 수신 데이터 정보

begin
    nSerialChNo := 0;           // 시리얼 채널(Port) 번호

    // 0 번 시리얼 포트에 데이터가 수신되었는지 확인합니다.
    if cesIsDataReady ( nSerialChNo, @nIsReady ) <> ceERR_NONE then
    begin
            if nIsReady = CE_TRUE then
            begin
                    ShowMessage ( ‘데이터 수신’ );
            end;
    end;
end;
```

NAME

cesSetTimeout / cesGetTimeout
- 타임아웃 설정

INFORMATION
1 Serial
! VC++ (6, 7, 8)/VB
BCB/Delphi
: Level 7
J 위험 요소 없음

SYNOPSIS

r VT_I4 cesSetTimeout ([in] VT_I4 Channel, [in] VT_I4 Timeout)

r VT_I4 cesGetTimeout ([in] VT_I4 Channel, [out] VT_PI4 Timeout)

DESCRIPTION

시리얼 통신 중 타임아웃을 설정하거나 설정된 타임아웃 값을 반환합니다.

PARAMETER

▶ *Channel* : 채널 채널 번호. 통합 채널로 관리되는 채널 번호를 의미하며, 상수 값으로 0 (Zero Based) 이상, (최대 통합 채널수 - 1) 이하의 값을 채널 번호로 설정할 수 있습니다.

▶ *TimeOut* : 이 매개변수를 통하여 타임 아웃 시간을 밀리초(ms) 단위로 설정 혹은 설정 시간을 반환합니다.

RETURN VALUE

Value	Meaning
음수	수행 실패. 자세한 내용은 '에러처리' 편을 참고합니다
0 (ceERR_NONE)	수행 성공

EXAMPLE

C/C++

```
#include “ceSDK.h”
#include “ceSDKDef.h”

long nSerialChNo = 0;          // 시리얼 채널(Port) 번호
long nTimeout;                 // 통신 Timeout 정보

// 0 번 시리얼 포트에 대해 통신 Timeout 을 확인 후, 1000 밀리초로 설정합니다.
if ( cesGetTimeout ( nSerialChNo, &nTimeout ) == ceERR_NONE )
{
    // 설정된 Timeout 값을 확인합니다.
    if ( nTimeout != 1000 )
```

```
    {
        cesSetTimeout ( nSerialChNo, 1000 );
    }
}
```

Visual Basic

```
Dim nSerialChNo As Long     ' 시리얼 채널(Port) 번호
Dim nTimeout As Long        ' 통신 Timeout 정보

nSerialChNo = 0

' 0 번 시리얼 포트에 대해 통신 Timeout 을 확인 후, 1000 밀리초로 설정합니다.
If cesGetTimeout ( nSerialChNo, nTimeout ) = ceERR_NONE Then

    ' 설정된 Timeout 값을 확인합니다.
    If nTimeout <> 1000 Then
        Call cesSetTimeout ( nSerialChNo, 1000 )
    End If
End If
```

Delphi

```
var
    nSerialChNo : LongInt;      // 시리얼 채널(Port) 번호
    nTimeout : LongInt;         // 통신 Timeout 정보

begin
    nSerialChNo := 0;

    // 0 번 시리얼 포트에 대해 통신 Timeout 을 확인 후, 1000 밀리초로 설정합니다.
    if cesGetTimeout ( nSerialChNo, @nTimeout ) = ceERR_NONE then
    begin
        // 설정된 Timeout 값을 확인합니다.
        if nTimeout <> 1000 then
        begin
            cesSetTimeout ( nSerialChNo, 1000 );
        end;
    end;
end;
```

NAME

cesRxReset / cesTxReset

- RX / TX 버퍼 초기화

INFORMATION
1 Serial
! VC++ (6, 7, 8)/VB
BCB/Delphi
: Level 7
J 위험 요소 없음

SYNOPSIS

r VT_I4 cesRxReset ([in] VT_I4 Channel)

r VT_I4 cesTxReset ([in] VT_I4 Channel)

DESCRIPTION

수신(RX) 버퍼 또는 송신(TX) 버퍼를 초기화 합니다.

PARAMETER

▶ *Channel* : 채널 번호. 통합 채널로 관리되는 채널 번호를 의미하며, 상수 값으로 0 (Zero Based) 이상, (최대 통합 채널수 - 1) 이하의 값을 채널 번호로 설정할 수 있습니다.

RETURN VALUE

Value	Meaning
음수	수행 실패. 자세한 내용은 '에러처리' 편을 참고합니다.
0 (ceERR_NONE)	수행 성공.

EXAMPLE

C/C++

```
#include “ceSDK.h”
#include “ceSDKDef.h”

long nSerialChNo = 0;          // 시리얼 채널(Port) 번호


// 0 번 시리얼 포트의 송신 버퍼를 비웁니다.
if ( cesTxReset ( nSerialChNo ) != ceERR_NONE )
{
    OutputDebugString ( "cesTxReset has been failed" );
}

// 0 번 시리얼 포트의 수신 버퍼를 비웁니다.
if ( cesRxReset ( nSerialChNo ) != ceERR_NONE )
{
    OutputDebugString ( "cesRxReset has been failed" );
```

```
}
```

Visual Basic

```
Dim nSerialChNo As Long      ‘ 시리얼 채널(Port) 번호

nSerialChNo = 0


‘ 0 번 시리얼 포트의 송신 버퍼를 비웁니다.
If cesTxReset ( nSerialChNo ) <> ceERR_NONE Then
    MagBox ( "cesTxReset has been failed" )
End If

‘ 0 번 시리얼 포트의 수신 버퍼를 비웁니다.
If cesRxReset ( nSerialChNo ) <> ceERR_NONE Then
    MsgBox ( "cesRxReset has been failed" )
End If
```

Delphi

```
var
    nSerialChNo : LongInt;      // 시리얼 채널(Port) 번호

begin

    nSerialChNo := 0;

    // 0 번 시리얼 포트의 송신 버퍼를 비웁니다.
    if cesTxReset ( nSerialChNo ) <> ceERR_NONE then
    begin
            ShowMessage ( ‘cesTxReset has been failed’ );
    end;

    // 0 번 시리얼 포트의 수신 버퍼를 비웁니다.
    if cesRxReset ( nSerialChNo ) <> ceERR_NONE then
    begin
            ShowMessage ( ‘cesRxReset has been failed’ );
    end;
end;
```

NAME

cesGetUnreadDataSize

- 읽지 않은 데이터 크기를 바이트 단위로 반환

INFORMATION	
1	Serial
!	VC++ (6, 7, 8)/VB
	BCB/Delphi
:	Level 7
J	위험 요소 없음

SYNOPSIS

r VT_I4 cesGetUnreadDataSize ([in] VT_I4 Channel, [out] VT_PI4 DataSize)

DESCRIPTION

수신 버퍼로부터 읽지 않은 데이터 크기를 바이트 단위로 반환합니다.

PARAMETER

▶ ***Channel*** : 채널 번호. 통합 채널로 관리되는 채널 번호를 의미하며, 상수 값으로 0 (Zero Based) 이상, (최대 통합 채널수 - 1) 이하의 값을 채널 번호로 설정할 수 있습니다.

▶ ***DataSize*** : 이 매개변수를 통하여 수신 버퍼에 읽지 않은 바이트 데이터 수를 반환합니다.

RETURN VALUE

Value	Meaning
음수	수행 실패. 자세한 내용은 '에러처리' 편을 참고합니다.
0 (ceERR_NONE)	수행 성공.

EXAMPLE

C/C++

```
#include “ceSDK.h”
#include “ceSDKDef.h”

long nSerialChNo = 0;          // 시리얼 채널(Port) 번호
long nDataSize;                // 수신 버퍼의 읽지 않은 데이터 수 정보

// 0 번 시리얼 포트의 수신 버퍼에서 읽지 않은 바이트 데이터 개수를 확인합니다.
if ( cesGetUnreadDataSize ( nSerialChNo, &nDataSize ) != ceERR_NONE )
{
    OutputDebugString ( "cesGetUnreadDataSize has been failed" );
}
```

Visual Basic

```
Dim nSerialChNo As Long     ‘ 시리얼 채널(Port) 번호
Dim nDataSize As Long       ‘ 수신 버퍼의 읽지 않은 데이터 수 정보

nSerialChNo = 0

‘ 0 번 시리얼 포트의 수신 버퍼에서 읽지 않은 바이트 데이터 개수를 확인합니다.
If cesGetUnreadDataSize ( nSerialChNo, nDataSize ) <> ceERR_NONE Then
    MsgBox ( "cesGetUnreadDataSize has been failed" )
End If
```

Delphi

```
var
    nSerialChNo : LongInt;      // 시리얼 채널(Port) 번호
    nDataSize : LongInt;        // 수신 버퍼의 읽지 않은 데이터 수 정보

begin
    nSerialChNo := 0;

    // 0 번 시리얼 포트의 수신 버퍼에서 읽지 않은 바이트 데이터 개수를 확인합니다.
    if cesGetUnreadDataSize ( nSerialChNo, @nDataSize ) <> ceERR_NONE then
    begin
            ShowMessage ( ‘cesGetUnreadDataSize has been failed’ );
    end;
end;
```

NAME

cesPeekByte / cesPeekByteEx

- Peek Byte Data

INFORMATION
1 Serial
! VC++ (6, 7, 8)/VB
BCB/Delphi
: Level 7
J 위험 요소 없음

SYNOPSIS

r VT_I4 cesPeekByte ([in] VT_I4 Channel, [out] VT_PI4 byData)

r VT_I4 cesPeekByteEx ([in] VT_I4 Channel, [in] VT_I4 ByteIdx, [out] VT_PI4 byData)

DESCRIPTION

cesPeekByte 함수는 수신 버퍼로부터 1 바이트의 데이터를 읽어옵니다.

cesPeekByteEx 함수는 수신 버퍼 내 원하는 위치의 1 바이트 데이터를 읽어옵니다.

두 함수 모두 버퍼 카운트를 감소 시키지 않습니다.

PARAMETER

▶ *Channel* : 채널 번호. 통합 채널로 관리되는 채널 번호를 의미하며, 상수 값으로 0 (Zero Based) 이상, (최대 통합 채널수 - 1) 이하의 값을 채널 번호로 설정할 수 있습니다.

▶ *ByteIdx* : cesPeekByteEx 함수의 인자로서 수신 버퍼에서 읽어올 바이트에 대한 위치 인덱스 값을 설정합니다.

▶ *byData* : 읽은 바이트 데이터를 반환합니다.

RETURN VALUE

Value	Meaning
음수	수행 실패. 자세한 내용은 '에러처리' 편을 참고합니다.
0 (ceERR_NONE)	수행 성공.

EXAMPLE

C/C++

```
#include “ceSDK.h”
#include “ceSDKDef.h”

long nSerialChNo = 0;          // 시리얼 채널(Port) 번호
BYTE byData;                   // 1 바이트 데이터 정보

// 0 번 시리얼 포트에 대해 수신 버퍼의 첫번째 바이트 데이터 확인
if ( cesPeekByte ( nSerialChNo, &byData) != ceERR_NONE )
{
    OutputDebugString ( "cesPeekByte has been failed" );
}

// 0 번 시리얼 포트에 대해 수신 버퍼의 임의의 위치의 바이트 데이터 확인
if ( cesPeekByteEx ( nSerialChNo, 5, &byData) != ceERR_NONE ) // zero base 로 6 번째 수신 데이터 확인
{
    OutputDebugString ( "cesPeekByteEx has been failed" );
}
```

Visual Basic

```
Dim nSerialChNo As Long     ‘ 시리얼 채널(Port) 번호
Dim byData As Byte;         ‘ 1 바이트 데이터 정보

nSerialChNo = 0

‘ 0 번 시리얼 포트에 대해 수신 버퍼의 첫번째 바이트 데이터 확인
If cesPeekByte ( nSerialChNo, byData) <> ceERR_NONE Then
    MsgBox ( "cesPeekByte has been failed" )
End If

‘ 0 번 시리얼 포트에 대해 수신 버퍼의 임의의 위치의 바이트 데이터 확인
‘ zero base 로 6 번째 수신 데이터를 확인합니다.
If cesPeekByteEx ( nSerialChNo, 5, byData) <> ceERR_NONE Then
    MsgBox ( "cesPeekByteEx has been failed" )
End If
```

Delphi

```
var
    nSerialChNo : LongInt;      // 시리얼 채널(Port) 번호
    byData : Byte;              // 1 바이트 데이터 정보

begin
    nSerialChNo := 0;
```

```
    // 0 번 시리얼 포트에 대해 수신 버퍼의 첫번째 바이트 데이터 확인
    if cesPeekByte ( nSerialChNo, @byData) <> ceERR_NONE then
    begin
            ShowMessage ( 'cesPeekByte has been failed' );
    end;

    // 0 번 시리얼 포트에 대해 수신 버퍼의 임의의 위치의 바이트 데이터 확인
    // zero base 로 6 번째 수신 데이터를 확인합니다.
    if cesPeekByteEx ( nSerialChNo, 5, @byData) <> ceERR_NONE then
    begin
            ShowMessage ( 'cesPeekByteEx has been failed' );
    end;
end;
```

NAME

cesReadByte / cesWriteByte

- Read / Write Byte Data

INFORMATION
1 Serial
! VC++ (6, 7, 8)/VB
BCB/Delphi
: Level 7
J 위험 요소 없음

SYNOPSIS

r VT_I4 cesReadByte ([in] VT_I4 Channel, [out] VT_PI4 byData)

r VT_I4 cesWriteByte ([in] VT_I4 Channel, [in] VT_I4 byData)

DESCRIPTION

cesReadByte 함수는 수신 버퍼로부터 1 바이트의 데이터를 읽어옵니다.

cesWriteByte 함수는 송신 버퍼에 전송을 위한 1 바이트 의 데이터를 추가합니다.

두 함수 모두 버퍼 카운트를 각각 1 만큼 감소 및 증가 시킵니다.

PARAMETER

▶ *Channel* : 채널 번호. 통합 채널로 관리되는 채널 번호를 의미하며, 상수 값으로 0 (Zero Based) 이상, (최대 통합 채널수 - 1) 이하의 값을 채널 번호로 설정할 수 있습니다.

▶ *byData* : 바이트 데이터.

RETURN VALUE

Value	Meaning
음수	수행 실패. 자세한 내용은 '에러처리' 편을 참고합니다
0 (ceERR_NONE)	수행 성공

EXAMPLE

C/C++

```
#include “ceSDK.h”
#include “ceSDKDef.h”

long nSerialChNo = 0;          // 시리얼 채널(Port) 번호
BYTE byData;                   // 1 바이트 데이터 정보

// 0 번 시리얼 포트에 대해 수신 버퍼의 첫번째 바이트 데이터 확인
if ( cesReadByte ( nSerialChNo, &byData ) == ceERR_NONE )
{
    if ( byData != 0x2 )       // STX 가 아니면
    {
            OutputDebugString ( "cesReadByte has been failed" );
    }
}

// 0 번 시리얼 포트에 대해 송신 버퍼에 1 바이트 데이터 추가
if ( cesWriteByte ( nSerialChNo, ‘B’ ) == ceERR_NONE )
{
    cesCommit ( nSerialChNo );          // 송신 버퍼의 내용을 전송 버퍼에 담아 전송
}
```

Visual Basic

```
Dim nSerialChNo As Long     ‘ 시리얼 채널(Port) 번호
Dim byData As Byte          ‘ 1 바이트 데이터 정보

nSerialChNo = 0

‘ 0 번 시리얼 포트에 대해 수신 버퍼의 첫번째 바이트 데이터 확인
If cesReadByte ( nSerialChNo, byData ) = ceERR_NONE Then
    If byData <> &H2 Then      ‘ STX 가 아니면
            MsgBox ( "cesReadByte has been failed" )
    End If
End If

‘ 0 번 시리얼 포트에 대해 송신 버퍼에 1 바이트 데이터 추가
If cesWriteByte ( nSerialChNo, ‘B’ ) = ceERR_NONE Then

    Call cesCommit ( nSerialChNo )      ‘ 송신 버퍼의 내용을 전송 버퍼에 담아 전송
End If
```

Delphi

```
var
    nSerialChNo : LongInt;      // 시리얼 채널(Port) 번호
```

```
    byData : Byte:                  // 1 바이트 데이터 정보

begin
    nSerialChNo := 0;

    // 0 번 시리얼 포트에 대해 수신 버퍼의 첫번째 바이트 데이터 확인
    if cesReadByte ( nSerialChNo, @byData ) = ceERR_NONE then
    begin
            if byData <> $2 Then        // STX 가 아니면
            begin
                    ShowMessage ( 'cesReadByte has been failed' );
            end;
    end;
end;

    // 0 번 시리얼 포트에 대해 송신 버퍼에 1 바이트 데이터 추가
    if cesWriteByte ( nSerialChNo, 'B' ) = ceERR_NONE then
    begin
            cesCommit ( nSerialChNo )           // 송신 버퍼의 내용을 전송 버퍼에 담아 전송
    end;
end;
```

NAME

cesPeekString / cesReadString / cesWriteString

- Peek / Read / Write String Data

INFORMATION	
1	Serial
!	VC++ (6, 7, 8)/VB
	BCB/Delphi
:	Level 7
J	위험 요소 없음

SYNOPSIS

r VT_I4 cesPeekString ([in] VT_I4 nChannel, [in] VT_I4 NumBytes, [out] VT_PSTR String)

r VT_I4 cesReadString ([in] VT_I4 Channel, [in] VT_I4 NumBytes, [out] VT_PSTR String)

r VT_I4 cesWriteString ([in] VT_I4 Channel, [in] VT_I4 NumBytes, [in] VT_STR String)

DESCRIPTION

cesPeekString 함수는 수신 버퍼로부터 지정한 바이트 수 만큼의 데이터를 읽어옵니다. 수신 버퍼 카운트는 감소 시키지 않습니다.

cesReadString 함수는 수신 버퍼로부터 지정한 바이트 수 만큼의 데이터를 읽어옵니다. 수신 버퍼 카운트는 읽은 바이트 수 만큼 감소 시킵니다.

cesWriteString 함수는 송신 버퍼에 지정된 바이트 수 만큼의 데이터를 추가합니다. 송신 버퍼 카운트를 해당 바이트 수 만큼 증가 시킵니다.

PARAMETER

▶ *Channel* : 채널 번호. 통합 채널로 관리되는 채널 번호를 의미하며, 상수 값으로 0 (Zero Based) 이상, (최대 통합 채널수 - 1) 이하의 값을 채널 번호로 설정할 수 있습니다.

▶ *NumBytes* : 바이트 수.

▶ *String* : 1 바이트 이상의 바이트열 데이터.

RETURN VALUE

Value	Meaning
음수	수행 실패. 자세한 내용은 '에러처리' 편을 참고합니다
0 (ceERR_NONE)	수행 성공

EXAMPLE

C/C++

```
#include “ceSDK.h”
#include “ceSDKDef.h”

long nSerialChNo = 0;          // 시리얼 채널(Port) 번호
BYTE abyData[8];               // 바이트열 데이터 정보
int nCheckSum;

// 0 번 시리얼 포트에 대해 수신 버퍼로부터 8 바이트 크기의 데이터 확인
if ( cesPeekString ( nSerialChNo, 8, &abyData) != ceERR_NONE )
{
    OutputDebugString ( "cesPeekString has been failed" );
}

// 0 번 시리얼 포트에 대해 수신 버퍼로부터 8 바이트 크기의 데이터 읽기
if ( cesReadString (SER_PORT0, 8, &abyData) != ceERR_NONE )
{
    OutputDebugString ( "cesPeekString has been failed" );
}

/* 전달 패킷 생성 */
abyData[0] = 0x2;     // STX
abyData[1] = ‘A’;
abyData[2] = ‘B’;
abyData[3] = ‘C’;
abyData[4] = ‘D’;
abyData[5] = ‘E’;

for ( int I = 0; I <= 5; i++ )
{
    nCheckSum += abyData[i];
}

abyData[6] = nCheckSum;
abyData[7] = 0x3;     // ETX

if ( cesWriteString ( nSerialChNo, 8, abyData ) == ceERR_NONE )
{
    cesCommit ( nSerialChNo );
}
```

Visual Basic

```
Dim nSerialChNo As Long     ‘ 시리얼 채널(Port) 번호
Dim abyData(8) As Byte      ‘ 바이트열 데이터 정보
Dim nCheckSum As Integer    ‘ CheckSum 정보

nSerialChNo = 0
nCheckSum = 0

‘ 0 번 시리얼 포트에 대해 수신 버퍼로부터 8 바이트 크기의 데이터 확인
If cesPeekString ( nSerialChNo, 8, abyData(0) ) <> ceERR_NONE Then
```

```
    MsgBox ( "cesPeekString has been failed" )
End If

‘ 0 번 시리얼 포트에 대해 수신 버퍼로부터 8 바이트 크기의 데이터 읽기
If cesReadString ( nSerialChNo, 8, abyData(0) ) <> ceERR_NONE Then
    MsgBox ( "cesPeekString has been failed" )
End If

‘ 전달 패킷 생성
abyData(0) = &H2     ‘ STX
abyData(1) = ‘A’
abyData(2) = ‘B’
abyData(3) = ‘C’
abyData(4) = ‘D’
abyData(5) = ‘E’

For i = 0 To 5
    nCheckSum = nCheckSum + abyData(i)
Next i

abyData(6) = nCheckSum
abyData(7) = &H3     ‘ ETX

If cesWriteString ( nSerialChNo, 8, abyData(0) ) = ceERR_NONE Then

    cesCommit ( nSerialChNo )
End If
```

```
Delphi

var
    nSerialChNo : LongInt;              // 시리얼 채널(Port) 번호
    abyData : Array[0..7] of Byte       // 바이트열 데이터 정보
    nCheckSum : Integer                 // CheckSum 정보

begin
    nSerialChNo := 0;
    nCheckSum := 0;

    // 0 번 시리얼 포트에 대해 수신 버퍼로부터 8 바이트 크기의 데이터 확인
    if cesPeekString ( nSerialChNo, 8, @abyData ) <> ceERR_NONE then
    begin
            ShowMessage ( ‘cesPeekString has been failed’ );
    end;

    // 0 번 시리얼 포트에 대해 수신 버퍼로부터 8 바이트 크기의 데이터 읽기
    if cesReadString ( nSerialChNo, 8, @abyData ) <> ceERR_NONE then
    begin
            ShowMessage ( ‘cesPeekString has been failed’ );
    end;

    // 전달 패킷 생성
    abyData[0] := $2;           // STX
    abyData[1] := ‘A’;
    abyData[2] := ‘B’;
    abyData[3] := ‘C’;
    abyData[4] := ‘D’;
```

```
    abyData[5] := 'E';

    For i := 0 to 5 do
    begin
            nCheckSum := nCheckSum + abyData[i];
    end;

    abyData[6] := nCheckSum;
    abyData[7] := $3;           // ETX

    if cesWriteString ( nSerialChNo, 8, @abyData ) = ceERR_NONE then
    begin
            cesCommit ( nSerialChNo );
    end;
end;
```

NAME

cesReadDword / cesWriteDword

- Read / Write Double Word Data

INFORMATION
1 Serial
! VC++ (6, 7, 8)/VB
BCB/Delphi
: Level 7
J 위험 요소 없음

SYNOPSIS

r VT_I4 cesReadDword ([in] VT_I4 Channel, [in] VT_I4 NumWords, [out] VT_PI4 dwData)

r VT_I4 cesWriteDword ([in] VT_I4 Channel, [in] VT_I4 NumWords, [in] VT_PI4 dwData)

DESCRIPTION

cesReadDword 함수는 수신 버퍼로부터 Double Word (4 바이트) 단위로 데이터를 읽어옵니다.

cesWriteDword 함수는 송신 버퍼에 전송을 위한 Double Word (4 바이트) 단위로 데이터를 추가합니다.

두 함수 모두 버퍼 카운트를 Double Word 개수에 해당하는 바이트 수만큼 각각 감소 및 증가 시킵니다.

PARAMETER

▶ ***Channel*** : 채널 번호. 통합 채널로 관리되는 채널 번호를 의미하며, 상수 값으로 0 (Zero Based) 이상, (최대 통합 채널수 - 1) 이하의 값을 채널 번호로 설정할 수 있습니다.

▶ ***NumWords*** : Double Word(4 바이트) 단위의 데이터 개수를 설정합니다.

▶ ***Dword*** : 1 개 이상의 Double Word 데이터.

RETURN VALUE

Value	Meaning
음수	수행 실패. 자세한 내용은 '에러처리' 편을 참고합니다.
0 (ceERR_NONE)	수행 성공.

EXAMPLE

C/C++

```
#include “ceSDK.h”
#include “ceSDKDef.h”

long nSerialChNo = 0;          // 시리얼 채널(Port) 번호
DWORD adwData[5];              // DWORD 열 데이터 정보

// 0 번 시리얼 포트에 대해 수신 버퍼로부터 5 DWORD 크기의 데이터 확인
if ( cesReadDword ( nSerialChNo, 5, &adwData ) != ceERR_NONE )
{
    OutputDebugString ( "cesReadDword has been failed" );
}

/* 전달 패킷 생성 */
adwData[0] = ‘@’;
adwData[1] = ‘A’;
adwData[2] = ‘B’;
adwData[3] = ‘C’;
abwData[4] = ‘!’;

if ( cesWriteDword ( nSerialChNo, 5, adwData) == ceERR_NONE )
{
    cesCommit ( nSerialChNo );
}
```

Visual Basic

```
Dim nSerialChNo As Long     ‘ 시리얼 채널(Port) 번호
Dim adwData[5] As Long      ‘ DWORD 열 데이터 정보

nSerialChNo = 0

‘ 0 번 시리얼 포트에 대해 수신 버퍼로부터 5 DWORD 크기의 데이터 확인
If cesReadDword ( nSerialChNo, 5, adwData ) <> ceERR_NONE Then
    MsgBox ( "cesReadDword has been failed" )
End If

‘ 전달 패킷 생성
adwData(0) = ‘@’
adwData(1) = ‘A’
adwData(2) = ‘B’
adwData(3) = ‘C’
abwData(4) = ‘!’

If cesWriteDword ( nSerialChNo, 5, adwData) = ceERR_NONE Then
    Call cesCommit ( nSerialChNo )
End If
```

Delphi

```
var
    nSerialChNo : LongInt;                  // 시리얼 채널(Port) 번호
    adwData : Array[0..4] of DWORD;    // DWORD 열 데이터 정보

begin
    nSerialChNo := 0;

    // 0 번 시리얼 포트에 대해 수신 버퍼로부터 5 DWORD 크기의 데이터 확인
    if cesReadDword ( nSerialChNo, 5, @adwData ) <> ceERR_NONE then
    begin
            ShowMessage ( 'cesReadDword has been failed' );
    end;


    // 전달 패킷 생성
    adwData[0] = '@';
    adwData[1] = 'A';
    adwData[2] = 'B';
    adwData[3] = 'C';
    adwData[4] = '!';

    if cesWriteDword ( nSerialChNo, 5, @adwData) = ceERR_NONE then
    begin
            cesCommit ( nSerialChNo );
    end;
end;
```

NAME

cesCommit

- Commit and Send Data

INFORMATION	
1	Serial
!	VC++ (6, 7, 8)/VB
	BCB/Delphi
:	Level 7
J	위험 요소 없음

SYNOPSIS

r VT_I4 cesCommit ([in] VT_I4 nChannel)

DESCRIPTION

송신 버퍼의 내용을 시리얼 통신 칩의 전송 버퍼에 담아 전송 시키는 작업을 수행합니다.

PARAMETER

▶ *Channel* : 채널 번호. 통합 채널로 관리되는 채널 번호를 의미하며, 상수 값으로 0 (Zero Based) 이상, (최대 통합 채널수 - 1) 이하의 값을 채널 번호로 설정할 수 있습니다.

RETURN VALUE

Value	Meaning
음수	수행 실패. 자세한 내용은 '에러처리' 편을 참고합니다.
0 (ceERR_NONE)	수행 성공.

EXAMPLE

C/C++

```
#include “ceSDK.h”
#include “ceSDKDef.h”

long nSerialChNo = 0;          // 시리얼 채널(Port) 번호
BYTE byData;                   // 바이트 데이터 정보
BYTE abyData[8];               // 바이트열 데이터 정보
int nCheckSum = 0;

// 0 번 시리얼 포트에 대해 수신 버퍼로부터 1 바이트 크기의 데이터 확인
if ( cesPeekByte ( nSerialChNo, &byData ) == ceERR_NONE )
{
    if ( byData != 0x2 )       // STX 가 아니면 원하는 패킷이 아님.
    {
            return;
    }
}
```

```
// 0 번 시리얼 포트에 대해 수신 버퍼로부터 8 바이트 크기의 데이터 읽기
if ( cesReadString ( nSerialChNo, 8, &abyData ) != ceERR_NONE )
{
    OutputDebugString ( "cesPeekString has been ailed" );
}

// 전달해 줄 패킷 생성
abyData[0] = 0x2;     // STX
abyData[1] = 'A';
abyData[2] = 'B';
abyData[3] = 'C';
abyData[4] = 'D';
abyData[5] = 'E';

for ( int I = 0; I <= 5; i++ )
{
    nCheckSum += abyData[i];
}

abyData[6] = nCheckSum;
abyData[7] = 0x3;     // ETX

if ( cesWriteString ( nSerialChNo, 8, abyData) == ceERR_NONE )
{
    cesCommit ( nSerialChNo );
}
```

Visual Basic

```
Dim nSerialChNo As Long     ' 시리얼 채널(Port) 번호
Dim byData As Byte          ' 바이트 데이터 정보
Dim abyData(8) As Byte      ' 바이트열 데이터 정보
Dim nCheckSum As Integer    ' CheckSum 정보

nSerialChNo = 0
nCheckSum = 0

' 0 번 시리얼 포트에 대해 수신 버퍼로부터 1 바이트 크기의 데이터 확인
If cesPeekByte ( nSerialChNo, byData ) = ceERR_NONE Then
    If byData <> &H2 Then     ' STX 가 아니면 원하는 패킷이 아님.
            MsgBox ( "cesPeekByte has been failed" )
    End If
End If

' 0 번 시리얼 포트에 대해 수신 버퍼로부터 8 바이트 크기의 데이터 읽기
If cesReadString ( nSerialChNo, 8, abyData(0) ) <> ceERR_NONE Then
    MsgBox ( "cesPeekString has been ailed" )
End If

' 전달해 줄 패킷 생성
abyData(0) = &H2              ' STX
abyData(1) = 'A'
abyData(2) = 'B'
abyData(3) = 'C'
abyData(4) = 'D'
abyData(5) = 'E'
```

```
For i = 0 To 5
    nCheckSum = nCheckSum + abyData(i)
Next i

abyData(6) = nCheckSum
abyData(7) = &H3                ' ETX

If cesWriteString ( nSerialChNo, 8, abyData(0) ) = ceERR_NONE Then
    Call cesCommit ( nSerialChNo )
End If
```

Delphi

```
var
    nSerialChNo : LongInt;              // 시리얼 채널(Port) 번호
    byData : Byte;                      // 바이트 데이터 정보
    abyData : Array[0..7] of Byte;      // 바이트열 데이터 정보
    nCheckSum : Integer;                // CheckSum 정보

begin
    nSerialChNo := 0;
    nCheckSum := 0;

    ' 0 번 시리얼 포트에 대해 수신 버퍼로부터 1 바이트 크기의 데이터 확인
    if ( cesPeekByte ( nSerialChNo, @byData ) = ceERR_NONE then
    begin
            if byData <> $2 then        // STX 가 아니면 원하는 패킷이 아님.
            begin
                    ShowMessage ( 'cesPeekByte has been failed' );
            end;
    end;

    // 0 번 시리얼 포트에 대해 수신 버퍼로부터 8 바이트 크기의 데이터 읽기
    if cesReadString ( nSerialChNo, 8, @abyData ) <> ceERR_NONE then
    begin
            ShowMessage ( 'cesPeekString has been ailed' );
    end;

    // 전달해 줄 패킷 생성
    abyData[0] = $2;            // STX
    abyData[1] = 'A';
    abyData[2] = 'B';
    abyData[3] = 'C';
    abyData[4] = 'D';
    abyData[5] = 'E';

    For i := 0 to 5 do
    begin
            nCheckSum := nCheckSum + abyData[i];
    end;

    abyData[6] := nCheckSum;
    abyData[7] := $3;           // ETX

    if cesWriteString ( nSerialChNo, 8, @abyData ) = ceERR_NONE then
    begin
```

```
            cesCommit ( nSerialChNo );
    end;
end;
```

Chapter

16

INTERLOCK Functions

커미조아의 cEIP 모듈은 각각의 모듈 기본 기능의 충실함은 물론, 응용 프로그램의 오류 및 시스템의 이상 상태로 인해 시스템의 제어가 불가능한 비상 사태 발생 시 적절한 조치를 통해 더 큰 손실로부터 시스템 및 장비를 보호할 수 있습니다.

이 단원에서는 인터락 기능에 대해 설명합니다. 인터락이란 소프트웨어 적인 제어 불능 상태 및 하드웨어적인 제어 불능 상태에 대해 미리 정해놓은 동작을 수행하여 시스템의 오동작을 방지하는 기능입니다. 본 단원을 통해 인터락 기능들을 자세히 살펴보고, 활용할 수 있도록 합니다.

16 인터락(Interlock) 편

16.1 함수 요약

"Interlock Functions"에 관련된 함수는 다음과 같습니다.

Summary of Functions
r VT_I4 ceil_Set ([in] VT_I4 NodeID, [in] VT_I4 Interlock_Type, [in] VT_I4 bEnable) 해당 원격 노드에 인터락 기능을 설정합니다.
r VT_I4 ceil_Get ([in] VT_I4 NodeID, [in] VT_I4 Interlock_Type, [out] VT_PI4 bIsEnabled) 해당 원격 노드의 인터락 기능 설정 상태를 반환합니다.
r VT_I4 ceilDisconnectTimeout_Set ([in] VT_I4 NodeID, [in] VT_I4 Disconnect_Type, [in] VT_I4 dwTimeOut) 해당 원격 노드의 Disconnect 인터락 체크 시 타임 아웃 시간을 밀리초 단위로 설정합니다.
r VT_I4 ceilDisconnectTimeout_Get ([in] VT_I4 NodeID, [in] VT_I4 Disconnect_Type, [out] VT_PI4 pdwTimeOut) 해당 원격 노드의 Disconnect 인터락 체크 타임 아웃 설정 시간을 반환합니다.
r VT_I4 ceilActionModeOne_Set ([in] VT_I4 NodeID, [in] VT_I4 ModuleType, [in] VT_I4 Channel, [in] VT_I4 ActionMode) 해당 원격 노드 내 슬레이브 모듈의 단일 채널에 대한 인터락 동작 모드를 설정합니다.
r VT_I4 ceilActionModeOne_Get ([in] VT_I4 NodeID, [in] VT_I4 ModuleType, [in] VT_I4 Channel, [out] VT_PI4 pdwActionMode) 해당 원격 노드 내 슬레이브 모듈의 단일 채널에 대한 인터락 동작 모드 설정 상태를 반환합니다.
r VT_I4 ceilActionModeMulti_Set ([in] VT_I4 NodeID, [in] VT_I4 ModuleType, [in] VT_I4 IniChan, [in] VT_I4 NumChan, [in] VT_I4 ActionModes) 해당 원격 노드 내 슬레이브 모듈의 다중 채널에 대한 인터락 동작 모드를 설정합니다.
r VT_I4 ceilActionModeMulti_Get ([in] VT_I4 NodeID, [in] VT_I4 ModuleType, [in] VT_I4 IniChan, [in] VT_I4 NumChan, [out] VT_PI4 ActionModes) 해당 원격 노드 내 슬레이브 모듈의 다중 채널에 대한 인터락 동작 모드 설정 상태를 반환합니다

16.2 함수 설명

NAME

ceil_Set / ceil_Get

- 인터락 기능 설정 및 반환

INFORMATION	
1	Interlock
!	VC++ (6, 7, 8)/VB
	BCB/Delphi
:	Level 7
J	위험 요소 없음

SYNOPSIS

r VT_I4 ceil_Set ([in] VT_I4 NodeID, [in] VT_I4 Interlock_Type, [in] VT_I4 bEnable)

r VT_I4 ceil_Get ([in] VT_I4 NodeID, [in] VT_I4 Interlock_Type, [out] VT_PI4 bIsEnabled)

DESCRIPTION

ceil_Set 함수를 통하여 해당 원격 노드에 인터락 기능을 설정합니다. 원격 노드와 마스터 PC 와의 연결 해제 상황 및 특정 센서 감지시에 대해 인터락 기능을 설정할 수 있습니다.

ceil_Get 함수를 통하여 인터락 기능의 활성/비활성 상태를 확인할 수 있습니다.

PARAMETER

▶ *NodeID* : 원격 노드 ID. 인터락 기능을 설정 할 원격 노드 ID 를 설정합니다.

▶ *Interlock_Type* : 인터락 종류를 선택합니다.

Value	Meaning
0 (INTLK_DISCONN)	노드와 마스터 PC 와의 연결 해제 상황에 대한 조건.
1 (INTLK_SENSOR)	센서 감지에 의한 조건.

▶ *bEnable* : 인터락 기능의 활성/비활성 상태를 설정합니다.

Value	Meaning
0 (CE_FALSE) [Default]	인터락 기능을 비활성 상태로 설정합니다.
1 (CE_TRUE)	인터락 기능을 활성 상태로 설정합니다.

RETURN VALUE

Value	Meaning
음수	수행 실패. 자세한 내용은 '에러처리' 편을 참고합니다
0 (ceERR_NONE)	수행 성공

EXAMPLE

C/C++

```
#include “ceSDK.h”
#include “ceSDKDef.h”

void OnSetInterlock ()
{
    long nNodeID = 1;           // 원격 노드 ID 설정
    long nIsEnabled;            // 인터락 기능 활성/비활성 정보

    /* Disconnect 인터락 활성 / 비활성 상태를 확인한 후, 활성 상태로 설정합니다. */

    if ( ceil_Get ( nNodeID, INTLK_DISCONN, &nIsEnabled ) = ceERR_NONE )
    {
            if ( nIsEnabled != CE_TRUE )
            {
                    ceil_Set ( nNodeID,             // 인터락 기능을 설정할 원격 노드 ID
                            INTLK_DISCONN,          // 0 (INTLK_DISCONN) : 연결 끊김 인터락
                                                    // 1 ( INTLK_SENSOR) : 센서 인터락
                            CE_TRUE                 // 0 (CE_FALSE) : 인터락 기능 비활성
                                                    // 1 (CE_TRUE) : 인터락 기능 활성
                            );
            }
    }
}
```

NAME

ceilDisconnectTimeout_Set /

ceilDisconnectTimeout_Get

- Disconnect 인터락 설정 및 반환

INFORMATION	
1	Interlock
!	VC++ (6, 7, 8)/VB
	BCB/Delphi
:	Level 7
J	위험 요소 없음

SYNOPSIS

r VT_I4 ceilDisconnectTimeout_Set ([in] VT_I4 NodeID, [in] VT_I4 Disconnect_Type, [in] VT_I4 dwTimeOut)

r VT_I4 ceilDisconnectTimeout_Get ([in] VT_I4 NodeID, [in] VT_I4 Disconnect_Type, [out] VT_PI4 pdwTimeOut)

DESCRIPTION

ceilDisconnectTimeout_Set 함수는 해당 원격 노드의 Disconnect 인터락 처리 시 Disconnect 인터락 종 류 및 타임 아웃 시간을 밀리초(ms) 단위로 설정합니다.

ceilDisconnectTimeout_Get 함수는 설정되어있는 Disconnect 인터락 처리 시 Disconnect 체크 타임 아웃 시간을 반환합니다.

PARAMETER

▶ ***NodeID*** : 원격 노드 ID. Disconnect 인터락 기능을 설정 할 원격 노드 ID 를 설정합니다.

▶ ***Disconnect_Type*** : Disconnect 인터락 종류를 선택합니다.

Value	Meaning
0 (DT_ PYSICAL)	물리적 Disconnect 인터락. 랜선 끊김 등 물리적인 요인으로 통신 연결이 끊기는 경우에 해당합니다.
1 (DT_ LOGICAL)	논리적 Disconnect 인터락. ceSDK 서비스 데몬 다운 등 소프트웨어적인 요인으로 통신 연결이 끊기는 경우에 해당합니다.

▶ ***dwTimeOut*** : Disconnect 인터락 처리 시 연결 해제에 대한 타임 아웃 체크 시간을 밀리초(ms) 단위로 설정합니다. 타임 아웃 체크 시간의 초기값은 500ms 입니다.

RETURN VALUE

Value	Meaning
음수	수행 실패. 자세한 내용은 '에러처리' 편을 참고합니다
0 (ceERR_NONE)	수행 성공

EXAMPLE

C/C++

```
#include “ceSDK.h”
#include “ceSDKDef.h”

#define NODE_ID     1 // 노드 IP 를 192.168.1.1 로 지정
enum {INTLK_DISCONN, INTLK_SENSOR}; // Interlock type : 연결 끊김 인터락(0), 센서 인터락(1)
enum {DT_PYSICAL, DT_LOGICAL}; // Disconnect type : 물리적(0), 논리적(1)

VOID CInterlockTestDlg::OnBtnSetDisconnectCheckTimeout()
{
    long val;

    long nResult = ceilDisconnectTimeout_Set(NODE_ID, DT_PYSICAL, 500); // 500ms 초로 Timeout 설정
    nResult = ceilDisconnectTimeout_Get(NODE_ID, DT_PYSICAL, &val);
}
```

NAME

ceilActionModeOne_Set /

ceilActionModeOne_Get

- 단일 채널에 대한 인터락 동작 모드 설정 및 반환

INFORMATION	
1	Interlock
!	VC++ (6, 7, 8)/VB
	BCB/Delphi
:	Level 7
J	위험 요소 없음

SYNOPSIS

r VT_I4 ceilActionModeOne_Set ([in] VT_I4 NodeID, [in] VT_I4 ModuleType, [in] VT_I4 Channel, [in] VT_I4 ActionMode)

r VT_I4 ceilActionModeOne_Get ([in] VT_I4 NodeID, [in] VT_I4 ModuleType, [in] VT_I4 Channel, [out] VT_PI4 ActionMode)

DESCRIPTION

ceilActionModeOne_Set 함수는 해당 원격 노드 슬레이브 모듈의 단일 채널에 대해 인터락 동작 모드를 설정합니다.

ceAI ActionModeOne_Get 함수는 해당 원격 노드 슬레이브 모듈의 단일 채널에 대해 설정되어있는 인터락 동작 모드를 반환합니다.

PARAMETER

▶ ***NodeID*** : 원격 노드 ID. 인터락 동작 모드를 설정할 채널이 속한 원격 노드 ID 를 설정합니다.

▶ ***Module_Type*** : 인터락 동작 모드를 설정할 대상 모듈 을 선택합니다.

Value	Meaning
0 (MOD_CPU)	마스터 모듈 (ceNM-SE) 을 선택합니다.
1 (MOD_AO)	슬레이브 모듈 중 아날로그 출력 모듈(ceAO02A) 을 선택합니다.
2 (MOD_DO)	슬레이브 모듈 중 디지털 출력 모듈(ceD16CM, ceDO32N) 을 선택합니다.
3 (MOD_MOT)	슬레이브 모듈 중 모션 모듈(ceMC02P)을 선택합니다.

▶ ***Channel*** : 인터락 동작 모드를 설정할 채널 번호. 통합 채널로 관리되는 채널 번호를 의미하며, 상수 값으로 0 (Zero Based) 이상, (최대 통합 채널수 - 1) 이하의 값을 채널 번호로 설정할 수 있습니다.

▶ *ActionMode* : 인터락 동작 모드를 설정합니다. Module_Type 에서 설정한 모듈 종류에 따라 인터락 동작 모드는 다음과 같이 구분됩니다.

Slave Module	Value	Meaning
ceNM-SE	0 (CPU_ACT_RESERVED)	Reserved.
	1 (CPU_ACT_REBOOT)	Reboot
	2 (CPU_ACT_KEEP)	현재 상태 유지(불변) [Default]
ceAO02N	0 (AO_ACT_MIN)	최소 전압출력 값(0 V, 0 mA) 출력 [Default]
	1 (AO_RESERVED1)	Reserved.
	2 (AO_ACT_KEEP)	현재 상태 유지(불변)
ceD16CN, ceDO32N	0 (DO_ACT_OFF)	OFF 출력 [Default]
	1 (DO_ACT_ON)	ON 출력
	2 (DO_ACT_KEEP)	현재 상태 유지(불변)
ceMC02P	0 (MOT_ACT_DECELSTOP)	감속후 정지 [Default]
	1 (MOT_ACT_EMGSTOP1)	즉시 정지
	2 (MOT_ACT_KEEP)	현재 상태 유지(불변)

RETURN VALUE

Value	Meaning
음수	수행 실패. 자세한 내용은 '에러처리' 편을 참고합니다
0 (ceERR_NONE)	수행 성공

EXAMPLE

C/C++

```
#include “ceSDK.h”
#include “ceSDKDef.h”

#define NODE_ID    1 // 노드 IP 를 192.168.1.1 로 지정

// Module type: ceNM-SE(0), ceAO02N(1), ceDxxN(2), ceMC02P(3)
enum {MOD_CPU, MOD_AO, MOD_DO, MOD_MOT};

// CPU (ceNM-SE) : reserved(0), reboot(1), 현재상태유지/불변(2) => reboot(1) 인 경우만 노드 마스터가 재부팅
됨.
enum {CPU_ACT_RESERVED, CPU_ACT_REBOOT, CPU_ACT_KEEP};

// Analog Output 모듈의 인터락 Action 정의 : 0: 최소 전압 값(0V, 0 mA)출력, 1: reserved, 2: 현재상태 유지(불변)
enum {AO_ACT_MIN, AO_RESERVED, AO_ACT_KEEP};

// Digital Output 모듈의 인터락 Action 정의 : off 출력(0), on 출력(1), 현출력상태 유지(2)
enum {DO_ACT_OFF, DO_ACT_ON, DO_ACT_KEEP};

// Motion 모듈의 인터락 Action 정의 : 0: 감속후 정지 (default), 1: 즉시 정지, 2: 현재상태 유지(불변)
enum {MOT_ACT_DECELSTOP, MOT_ACT_EMGSTOP, MOT_ACT_KEEP};
```

```
VOID CInterlockTestDlg::OnBtnActModeSetOne
{
    long val;

// 1. CPU 모듈 리셋 설정
    // Module Type => 0: ceNM-SE(CPU 모듈)
    // Channel => CPU 모듈인 경우 임의값, don't care
    // Action Mode => 1 이외의 값은 Reboot 하지 않음, 1: 재부팅
    long nResult = ceilActionModeOne_Set(NODE_ID, MOD_CPU, 0, CPU_ACT_REBOOT);

    // 2. CPU 모듈 리셋 설정 확인
    nResult = ceilActionModeOne_Get(NODE_ID, MOD_CPU, 0, &val);

// 3. AO 모듈 단일 채널의 인터락 동작 설정
    // Module Type => 1: 아날로그출력모듈
    // Channel => 0 번
    // Action Mode => AO_ACT_MIN : 최소값 출력 (0V, 0mA)
    nResult = ceilActionModeOne_Set(NODE_ID, MOD_AO, 0, AO_ACT_MIN);

    // 4. AO 모듈 단일 채널의 인터락 동작 설정 확인
    nResult = ceilActionModeOne_Get(NODE_ID, MOD_AO, 0, &val);

// 5. DO 모듈 단일 채널의 인터락 동작 설정
    // Module Type => 2: 디지털출력모듈
    // Channel => 0 번
    // Action Mode => DO_ACT_OFF : 출력 OFF
    nResult = ceilActionModeOne_Set(NODE_ID, MOD_DO, 0, DO_ACT_OFF);

    // 6. DO 모듈 단일 채널의 인터락 동작 설정 확인
    nResult = ceilActionModeOne_Get(NODE_ID, MOD_DO, 0, &val);

// 7. MOTION 모듈 단일 채널의 인터락 동작 설정
    // Module Type => 3: 모션 모듈
    // Channel => 0 번 축
    // Action Mode => MOT_ACT_EMGSTOP : 비상 정지 (Emergency Stop)
    nResult = ceilActionModeOne_Set(NODE_ID, MOD_MOT, 0, MOT_ACT_EMGSTOP);

    // 8. MOTION 모듈 단일 채널의 인터락 동작 설정 확인
    nResult = ceilActionModeOne_Get(NODE_ID, MOD_MOT, 0, &val);
}
```

NAME

ceilActionModeMulti_Set /

ceilActionModeMulti_Get

- 다중 채널에 대한 인터락 동작 모드 설정 및 반환

INFORMATION	
1	Interlock
!	VC++ (6, 7, 8)/VB
	BCB/Delphi
:	Level 7
J	위험 요소 없음

SYNOPSIS

r VT_I4 ceilActionModeMulti_Set ([in] VT_I4 NodeID, [in] VT_I4 ModuleType, [in] VT_I4 IniChan, [in] VT_I4 NumChan, [in] VT_I4 ActionModes)

r VT_I4 ceilActionModeMulti_Get ([in] VT_I4 NodeID, [in] VT_I4 ModuleType, [in] VT_I4 IniChan, [in] VT_I4 NumChan, [out] VT_PI4 ActionModes)

DESCRIPTION

ceilActionModeMulti_Set 함수는 해당 원격 노드 슬레이브 모듈의 다중 채널에 대해 인터락 동작 모드를 설정합니다.

ceAI ActionModeOne_Get 함수는 해당 원격 노드 슬레이브 모듈의 다중 채널에 대해 설정되어있는 인터락 동작 모드를 반환합니다.

PARAMETER

▶ ***NodeID*** : 원격 노드 ID. 인터락 동작 모드를 설정할 채널 속한 원격 노드 ID 를 설정합니다.

▶ ***Module_Type*** : 인터락 동작 모드를 설정할 대상 모듈 을 선택합니다.

Value	Meaning
0 (MOD_CPU)	마스터 모듈 (ceNM-SE) 을 선택합니다.
1 (MOD_AO)	슬레이브 모듈 중 아날로그 출력 모듈(ceAO02A) 을 선택합니다.
2 (MOD_DO)	슬레이브 모듈 중 디지털 출력 모듈(ceD16CM, ceDO32N) 을 선택합니다.
3 (MOD_MOT)	슬레이브 모듈 중 모션 모듈(ceMC02P)을 선택합니다.

▶ ***IniChan*** : 인터락 동작 모드를 설정할 시작 채널 번호. 통합 채널로 관리되는 채널 번호를 의미하며, 상수 값으로 0 (Zero Based) 이상, (최대 통합 채널수 - 1) 이하의 값을 채널 번호로 설정할 수 있습니다.

▶ ***NumChan*** : 시작 채널로부터 몇 개의 채널의 상태를 확인할 것인지에 대한 값을 전달합니다. (32 Bit 변수를 Bit Mask 로 사용하며, 채널당 모드는 0, 1, 2, 3 의 4 가지 상태에 대해 2 Bit 가 필요합니다. 즉, 한번에 최대 16 개의 채널에 대해 설정 가능합니다).

▶ ***ActionMode*** : 인터락 동작 모드를 설정합니다. Module_Type 에서 설정한 모듈 종류에 따라 인터락 동작 모드는 다음과 같이 구분됩니다. 복수개의 채널에 대한 Action Mode 를 Bit Mask 로 설정하시면 됩니다.

Slave Module	Value	Meaning
ceNM-SE	0 (CPU_ACT_RESERVED)	Reserved.
	1 (CPU_ACT_REBOOT)	Reboot
	2 (CPU_ACT_KEEP)	현재 상태 유지(불변) **[Default]**
ceAO02N	0 (AO_ACT_MIN)	최소 전압출력 값(0 V, 0 mA) 출력 **[Default]**
	1 (AO_RESERVED1)	Reserved.
	2 (AO_ACT_KEEP)	현재 상태 유지(불변)
ceD16CN, ceDO32N	0 (DO_ACT_OFF)	OFF 출력 **[Default]**
	1 (DO_ACT_ON)	ON 출력
	2 (DO_ACT_KEEP)	현재 상태 유지(불변)
ceMC02P	0 (MOT_ACT_DECELSTOP)	감속후 정지 **[Default]**
	1 (MOT_ACT_EMGSTOP1)	즉시 정지
	2 (MOT_ACT_KEEP)	현재 상태 유지(불변)

RETURN VALUE

Value	Meaning
음수	수행 실패. 자세한 내용은 '에러처리' 편을 참고합니다
0 (ceERR_NONE)	수행 성공

EXAMPLE

C/C++

```
#include “ceSDK.h”
#include “ceSDKDef.h”

#define NODE_ID    1 // 노드 IP 를 192.168.1.1 로 지정

// Module type: ceNM-SE(0), ceAO02N(1), ceDxxN(2), ceMC02P(3)
enum {MOD_CPU, MOD_AO, MOD_DO, MOD_MOT};

// CPU (ceNM-SE) : reserved(0), reboot(1), 현재상태유지/불변(2) => reboot(1) 인 경우만 노드 마스터가 재부팅
됨.
enum {CPU_ACT_RESERVED, CPU_ACT_REBOOT, CPU_ACT_KEEP};

// Analog Output 모듈의 인터락 Action 정의 : 0: 최소 전압 값(0V, 0 mA)출력, 1: reserved, 2: 현재상태 유지(불변)
enum {AO_ACT_MIN, AO_RESERVED, AO_ACT_KEEP};
```

```
// Digital Output 모듈의 인터락 Action 정의 : off 출력(0), on 출력(1), 현출력상태 유지(2)
enum {DO_ACT_OFF, DO_ACT_ON, DO_ACT_KEEP};

// Motion 모듈의 인터락 Action 정의 : 0: 감속후 정지 (default), 1: 즉시 정지, 2: 현재상태 유지(불변)
enum {MOT_ACT_DECELSTOP, MOT_ACT_EMGSTOP, MOT_ACT_KEEP};

VOID CInterlockTestDlg::OnBtnActModeSetOne
{
    long val;

// 1. AO 모듈 다중 채널의 인터락 동작 설정
    // Module Type => 1: 아날로그출력모듈
    // IniChan => 0 번 채널부터
    // NumChan => 2 개 (0, 1 번 채널에 대해 설정)
    // Action Mode => 채널별로 다르게 설정

// 0 번채널: AO_ACT_KEEP (2), 1 번채널:AO_ACT_OFF (0)
long nResult = ceilActionModeMulti_Set (NODE_ID, MOD_AO, 0, 2, 0x2); // 이진수로 0010 으로 설정.

    // 2. AO 모듈 다중 채널의 인터락 동작 설정 확인
    nResult = ceilActionModeMulti_Get (NODE_ID, MOD_AO, 0, 2, &val);

// 3. DO 모듈 다중 채널의 인터락 동작 설정
    // Module Type => 2: 디지털출력모듈
    // IniChan => 0 번 채널부터
    // NumChan => 16 개 (0 ~15 번 채널에 대해 설정)
    // Action Mode => 상위 8 개 채널은 OFF (0), 하위 8 개 채널은 기존 출력을 유지(2) 하도록 설정
// 이진수로 00000000000000001010101010101010
    nResult = ceilActionModeMulti_Set (NODE_ID, MOD_DO, 0, 16, 0x0000AAAA);

    // 4. DO 모듈 다중 채널의 인터락 동작 설정 확인
    nResult = ceilActionModeMulti_Get (NODE_ID, MOD_DO, 0, 16, &val);

// 5. MOTION 모듈 다중 채널의 인터락 동작 설정
    // Module Type => 3: 모션 모듈
    // IniChan => 0 번 채널부터
    // NumChan => 4 개 (0~3 번 축에 대해 설정)
    // Action Mode => 축 0 번:비상정지(1), 축 1:감속후정지(0),축 2:현상태유지(2),축 3:비상정지(1) 로 설정
// 이진수로 01100001
    nResult = ceilActionModeMulti_Set (NODE_ID, MOD_MOT, 0, 4, 0x61);

    // 6. MOTION 모듈 다중 채널의 인터락 동작 설정 확인
    nResult = ceilActionModeMulti_Get (NODE_ID, MOD_MOT, 0, 4, &val);
}
```

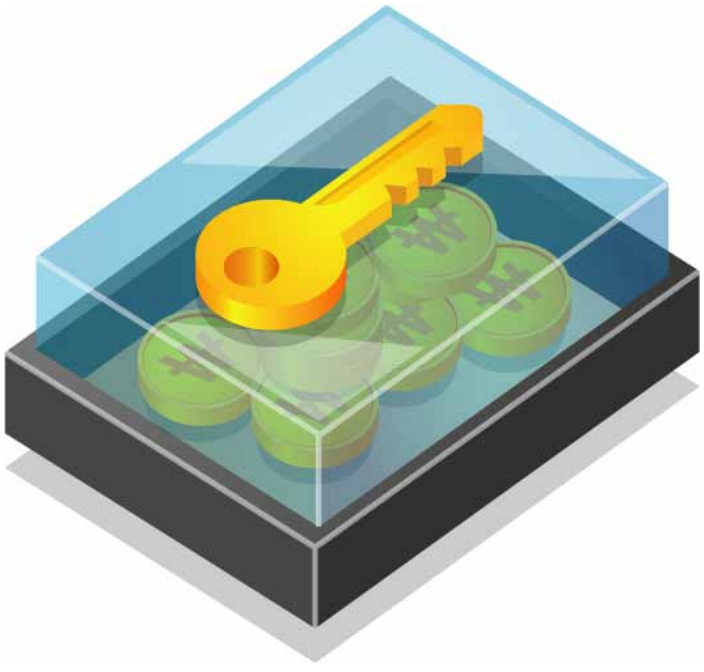
Chapter

17

Utility Functions

고객님의 소중한응용프로그램에서 라이브러리가 그 기능의 일부를 담당한다면, 안전성과 신뢰성이 결여될 수는 없습니다. 수년간의 다양한 개발 요건들을 충족시켜온 저희 ㈜커미조아의 라이브러리 기술은 이제 기능적 라이브러리의 기능을 넘어, 감각적 라이브러리의 기능을 추구하고 있습니다. 모션 라이브러리가 가질 수 있는 진정한 세계로 지금 고객님을 초대합니다.

최단 시간내에 개발되어야 하는 고객사의 제품 개발에는 분명 개발 과정의 다양한 변수들이 존재하고 있습니다. ceSDK 는 개발 과정 전후에 발생할 수 있는 모든 소프트웨어의 장애물들을 제거하기 위해 존재하는 강력한 디버그 관련 기능을 제공합니다. 또한 실제적인 윈도우 응용프로그램에서 필요한 각종 유틸리티 함수 지원을 통해 고객 여러분들에 조력자 역할을 아끼지 않겠습니다.



17 유틸리티 함수 편

㈜커미조아의 ceSDK 라이브러리는 개발자에게 특수한 처리를 위한 유틸리티 함수를 제공합니다. 이 함수들은 시스템의 직접적인 제어 기능에는 관련이 없으나, 고객님들께서 구현하시는 응용 프로그램이나 관련된 응용 범위에서 유용하게 사용할 수 있는 주요한 기능으로 준비되었습니다. 최적화된 ceSDK 라이브러리에 다양한 유틸리티 함수들을 이용함으로써 최적의 소프트웨어를 구현할 수 있도록 안내하여 드립니다.

17.1 함수 요약

ceSDK 에서 제공하는 유틸리티 함수들은 다음과 같습니다. 해당 함수 목록은 실제 응용 프로그램의 개발에 있어, 고객님들의 프로그램과 보다 효과적으로 작용할 수 있는 기능을 제공합니다. 강력한 ceSDK 라이브러리 기능을 바탕으로 제공되는 유틸리티 함수는 고객님들의 응용 프로그램에서 뛰어난 성능을 발휘합니다.

유틸리티 기능에 관련된 함수는 다음과 같습니다.

Summary of Functions
r VT_I4 ceutlUserData_Set ([in] VT_I4 NodeID, [in] VT_I4 NumByte, [in] VT_PSTR szText) 해당 원격 노드(Node)에 사용자 정의 데이터를 기록합니다.
r VT_I4 ceutlUserData_Get ([in] VT_I4 NodeID, [out] VT_PI4 pNumByte, [out] VT_PSTR szText) 해당 원격 노드에 저장되어 있는 사용자 정의 데이터를 반환합니다.
r VT_I4 ceutlUserVersion_Set ([in] VT_I4 NodeID, [in] VT_I4 Version) 해당 원격 노드에 사용자 정의 버전을 기록합니다.
r VT_I4 ceutlUserVersion_Get ([in] VT_I4 NodeID, [out] VT_PI4 pVersion) 해당 원격 노드의 사용자 정의 버전을 읽어옵니다.
r VT_I4 ceutlNodeVersion_Get ([in] VT_I4 NodeID, [out] VT_PI4 pVersion) 해당 원격 노드의 펌웨어 버전을 읽어옵니다.
r VT_I4 ceutlLibVersion_Get ([out] VT_PI4 pVersionMS, [out] VT_PI4 pVersionLS) ceSDK Library 버전을 읽어옵니다. 상위 및 하위 4 바이트의 주소에 각 2 바이트씩 버전을 할당합니다. 버전의 자릿수는 총 4 자리 입니다.
r VT_I4 ceutlPumpSingleMessage ([none] VT_EMPTY) 단일 윈도우 메시지를 처리합니다.
r VT_I4 ceutlPumpMultiMessage ([in] VT_I4 nTimeout) 다중 윈도우 메시지를 처리합니다.
r VT_I4 ceutlSyncCount_Get ([in] VT_I4 NodeID, [out] VT_PI4 pSyncCount) 해당 원격 노드의 명령을 동기화 하기 위한 동기 카운트를 반환합니다.

r VT_I4 ceutlIOSyncCount_Get ([in] VT_I4 NodeID, [out] VT_PI4 pSyncCount) 해당 원격 노드의 명령을 동기화 하기 위한 I/O 메시지 동기 카운트를 반환합니다.
r VT_I4 ceutlSyncWait ([in] VT_I4 NodeID, [in] VT_I4 IsBlocking) 해당 원격 노드의 노드와 상위 응용 프로그램의 명령 동기화에 대해 처리합니다.

17.2 함수 설명

NAME

ceutlUserData_Set / ceutlUserData_Get

- 원격 노드에 사용자 정의 데이터 기록 및 반환

INFORMATION	
1	Utility
!	VC++ (6, 7, 8)/VB
	BCB/Delphi
:	Level 7
J	위험 요소 없음

SYNOPSIS

r VT_I4 ceutlUserData_Set ([in] VT_I4 NodeID, [in] VT_I4 NumByte, [in] VT_PSTR szText)

r VT_I4 ceutlUserData_Get

([in] VT_I4 NodeID, [out] VT_PI4 pNumByte, [out] VT_PSTR szText)

DESCRIPTION

ceutlUserData_Set 함수는 사용자 정의 데이터를 해당 원격 노드(Node) 에 기록합니다.

ceutlUserData_Get 함수는 해당 원격 노드에 저장되어 있는 사용자 정의 데이터를 반환합니다.

PARAMETER

▶ ***NodeID*** : 원격 노드의 ID 를 설정합니다.

▶ ***NumByte*** : ceutlUserData_Set 함수에서 사용하며, szText 의 바이트 개수를 설정합니다. 최대 32 바이트 까지 설정할 수 있습니다.

▶ ***pNumByte*** : ceutlUserData_Get 함수에서 사용하며, szText 의 바이트 개수를 반환합니다.

▶ ***szText*** : 사용자 정의 문자열.

RETURN VALUE

Value	Meaning
음수	수행 실패. 자세한 내용은 '에러처리' 편을 참고합니다.
0 (ceERR_NONE)	수행 성공.

NAME

ceutlUserVersion_Get/ ceutlUserVersion_Get

- 원격 노드에 사용자 정의 버전 기록 및 반환

INFORMATION	
1	Utility
!	VC++ (6, 7, 8)/VB
	BCB/Delphi
:	Level 7
J	위험 요소 없음

SYNOPSIS

r VT_I4 ceutlUserVersion_Set ([in] VT_I4 NodeID, [in] VT_I4 Version)

r VT_I4 ceutlUserVersion_Get ([in] VT_I4 NodeID, [out] VT_PI4 pVersion)

DESCRIPTION

ceutlUserVersion_Set 함수는 대상 원격 노드에 사용자 정의 버전을 기록합니다.

ceutlUserVersion_Get 함수는 대상 원격 노드의 사용자 정의 버전을 읽어옵니다.

PARAMETER

▶ ***NodeID*** : 원격 노드의 ID 를 설정합니다.

▶ ***Version*** : ceutlUserVersion_Set 함수에서 사용하며, 기록할 사용자 정의 버전을 설정합니다.

▶ ***pVersion*** : ceutlUserVersion_Get 함수에서 사용하며, 기록된 사용자 정의 버전을 반환합니다.

RETURN VALUE

Value	Meaning
음수	수행 실패. 자세한 내용은 '에러처리' 편을 참고합니다.
0 (ceERR_NONE)	수행 성공.

NAME

ceutlNodeVersion_Get

- 원격 노드의 펌웨어(Firmware) 버전을 반환

INFORMATION
1 Utility
! VC++ (6, 7, 8)/VB
BCB/Delphi
: Level 7
J 위험 요소 없음

SYNOPSIS

r VT_I4 ceutlNodeVersion_Get ([in] VT_I4 NodeID, [out] VT_PI4 pVersion)

DESCRIPTION

해당 원격 노드의 펌웨어(Firmware) 버전을 읽어옵니다.

PARAMETER

▶ ***NodeID*** : 원격 노드의 ID 를 설정합니다.

▶ ***pVersion*** : 해당 원격 노드의 펌웨어 버전을 반환합니다.

RETURN VALUE

Value	Meaning
음수	수행 실패. 자세한 내용은 '에러처리' 편을 참고합니다.
0 (ceERR_NONE)	수행 성공.

NAME

ceutlLibVersion_Get

- ceSDK 라이브러리의 버전 확인

INFORMATION	
1	Utility
!	VC++ (6, 7, 8)/VB
	BCB/Delphi
:	Level 7
J	위험 요소 없음

SYNOPSIS

r VT_I4 ceutlLibVersion_Get ([out] VT_PI4 pVersionMS, [out] VT_PI4 pVersionLS)

DESCRIPTION

ceSDK 라이브러리 버전을 반환합니다. 상위 및 하위 4 바이트의 주소에 각 2 바이트씩 버전 정보를 할당합니다. 버전의 자릿수는 총 4 자리입니다.

PARAMETER

▶ ***pVersionMS*** : 라이브러리 버전 정보 중 상위 비트 버전 정보를 반환합니다.

▶ ***pVersionLS*** : 라이브러리 버전 정보 중 하위 비트 버전 정보를 반환합니다.

RETURN VALUE

Value	Meaning
음수	수행 실패. 자세한 내용은 '에러처리' 편을 참고합니다.
0 (ceERR_NONE)	수행 성공.

NAME

ceutlPumpSingleMessage

- 단일 윈도우 메시지 처리

INFORMATION
1 Utility
! VC++ (6, 7, 8)/VB
BCB/Delphi
: Level 7
J 위험 요소 없음

SYNOPSIS

r VT_I4 ceutlPumpSingleMessage ([none] VT_EMPTY)

DESCRIPTION

단일 윈도우 메시지를 처리합니다.

RETURN VALUE

Value	Meaning
음수	수행 실패. 자세한 내용은 '에러처리' 편을 참고합니다.
0 (ceERR_NONE)	수행 성공.

SEE ALSO

ceutlPumpMultiMessage

NAME

ceutlPumpMultiMessage

- 다중 윈도우 메시지 처리

INFORMATION	
1	Utility
!	VC++ (6, 7, 8)/VB
	BCB/Delphi
:	Level 7
J	위험 요소 없음

SYNOPSIS

r VT_I4 ceutlPumpMultiMessage ([in] VT_I4 nTimeout)

DESCRIPTION

다중 윈도우 메시지를 처리합니다.

응용 프로그램 혹은 쓰레드(Thread) 가 메시지 큐(Queue) 에 있는 윈도우 메시지 처리에 너무 많은 시간을 할당하지 않도록 nTimeout 을 밀리초(millisecond) 단위로 설정할 수 있습니다. 이 시간 안에 지정된 다중 윈도우 메시지가 처리 됩니다.

PARAMETER

▶ ***nTimeout*** : 지정된 타임 아웃 시간 안에 다중 윈도우 메시지를 처리합니다. 단, nTimeout 을 CN_INFINITE 로 설정하게 되면, 모든 윈도우 메시지를 처리한 후 반환됩니다. 단위는 ms 입니다.

RETURN VALUE

Value	Meaning
음수	수행 실패. 자세한 내용은 '에러처리' 편을 참고합니다
0 (ceERR_NONE)	수행 성공

SEE ALSO

ceutlPumpSingleMessage

NAME

ceutlSyncCount_Get

- 원격 노드 명령을 동기화 하기 위한 동기 카운트 반환

INFORMATION	
1	Utility
!	VC++ (6, 7, 8)/VB
	BCB/Delphi
:	Level 7
J	위험 요소 없음

SYNOPSIS

r VT_I4 ceutlSyncCount_Get ([in] VT_I4 NodeID, [out] VT_PI4 pSyncCount)

DESCRIPTION

해당 원격 노드의 명령을 동기화 하기 위한 동기 카운트를 반환합니다.

PARAMETER

▶ *NodeID* : 원격 노드의 ID 를 설정합니다.

▶ *pSyncCount* : 대상 원격 노드의 현재 카운트 정보를 반환합니다.

RETURN VALUE

Value	Meaning
음수	수행 실패. 자세한 내용은 '에러처리' 편을 참고합니다.
0 (ceERR_NONE)	수행 성공.

SEE ALSO

ceutlIOSyncCount_Get

NAME

ceutlIOSyncCount_Get

- 원격 노드 명령을 동기화 하기 위한 I/O 메시지 동기 카운트를 반환

INFORMATION	
1	Utility
!	VC++ (6, 7, 8)/VB
	BCB/Delphi
:	Level 7
J	위험 요소 없음

SYNOPSIS

r VT_I4 ceutlIOSyncCount_Get ([in] VT_I4 NodeID, [out] VT_PI4 pSyncCount)

DESCRIPTION

해당 원격 노드의 명령을 동기화 하기 위한 I/O 메시지 동기 카운트를 반환합니다.

PARAMETER

▶ ***NodeID*** : 원격 노드의 ID 를 설정합니다.

▶ ***pSyncCount*** : 대상 원격 노드의 현재 카운트 정보를 반환합니다.

RETURN VALUE

Value	Meaning
음수	수행 실패. 자세한 내용은 '에러처리' 편을 참고합니다
0 (ceERR_NONE)	수행 성공

SEE ALSO

ceutlSyncCount_Get

NAME

ceutlSyncWait

- 원격 노드와 상위 응용 프로그램간 명령 동기화 처리

INFORMATION	
1	Utility
!	VC++ (6, 7, 8)/VB
	BCB/Delphi
:	Level 7
J	위험 요소 없음

SYNOPSIS

r VT_I4 ceutlSyncWait ([in] VT_I4 NodeID, [in] VT_I4 IsBlocking)

DESCRIPTION

원격 노드와 상위 응용 프로그램의 명령 동기화에 대해 처리합니다. 실제 명령된 명령 개수와 주기적인 메시지의 동기화를 위해 대기합니다.

PARAMETER

▶ *NodeID* : 원격 노드의 ID 를 설정합니다.

▶ *IsBlocking* : 동기를 위해 대기하는 동안 윈도우 메시지를 블록할 것인지를 결정합니다.

RETURN VALUE

Value	Meaning
음수	수행 실패. 자세한 내용은 '에러처리' 편을 참고합니다
0 (ceERR_NONE)	수행 성공

Chapter

18

Advanced and Extended Interface

다양한 고급 기능 지원을 위해 ceSDK 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 다양한 제어를 위해 사용되어집니다.

모션 고급 기능은 커미조아의 다양한 확장 모션 제어에 주로 이용되어 집니다. 정확한 모션 제어와 안정적인 모션제어를 위해 제공되는 고급 모션 기능들은 ㈜ 커미조아 기술 진들이나 관련 기술 지원 협약점을 통해서 문의해주시기 바랍니다.

18 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 문의해 주시기 바랍니다.

고급 기능 함수들의 이름은 모두 중간 첨두어 “Adv”를 포함하며 이에 해당하는 함수들의 리스트는 다음의 표와 같습니다.

18.1 함수 요약

r VT_I4 cemAdvGetNodeInformation ([in] VT_I4 nNode, [out] PTNode pTargetNode)
Undocument Function 입니다. 이 함수는 기술 지원이나 고객 지원용으로 사용됩니다.

r VT_I4 cemAdvGetAllNodeInformation ([out] PTNodeInformation pTargetNodes)
Undocument Function 입니다. 이 함수는 기술 지원이나 고객 지원용으로 사용됩니다.

r VT_I4 cemAdvErcOut ([in] VT_I4 Axis)
해당 축에 ERC 신호를 출력합니다.

r VT_I4 cemAdvErcReset ([in] VT_I4 Axis)
해당 축에 ERC 신호의 출력을 리셋합니다.

r VT_I4 cemAdvManualPacket ([in] VT_I4 NodeID, [in] VT_I4 CommandNo,
[in] VT_PR8 SendBuffer, [in] VT_I4 NumSendData, [out] VT_PR8 RecvBuffer,
[out] VT_PI4 NumRecvData, [in] VT_I4 SendFlag, [in] VT_I4 RecvFlag)
Undocument Function 입니다. 이 함수는 기술 지원이나 고객 지원용으로 사용됩니다.

Appendix

A

cEIP Runtime Environment

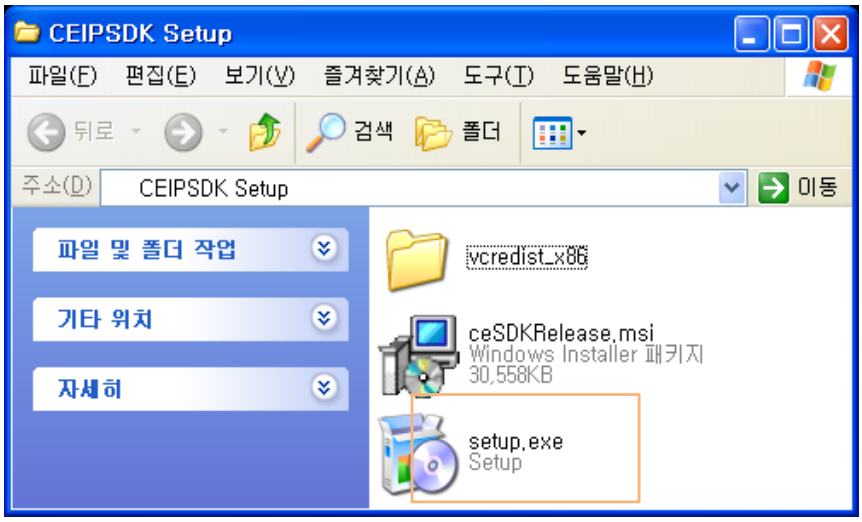
간편한 설치는 제품의 가용성을 증대 시키게 됩니다. ㈜ 커미조아 cEIP 제품은 쉽고 빠른 설치, 그리고 안정적인 설치를 위해 설계되었으며, 많은 분야에 접목될 수 있는 구조로 설계되어 있습니다. 이제 cEIP 제품을 통해 경쟁력 있는 제품 개발에 효과를 거두시기 바랍니다.

본 장은 초기 cEIP 구동 환경을 구축하는데 관련된 내용에 대해 안내하고 있습니다. 처음 cEIP 제품을 사용하시는 분들께서는 제품 사용에 어려움이 없도록 본 장의 내용을 숙지하여 주시기 바랍니다.

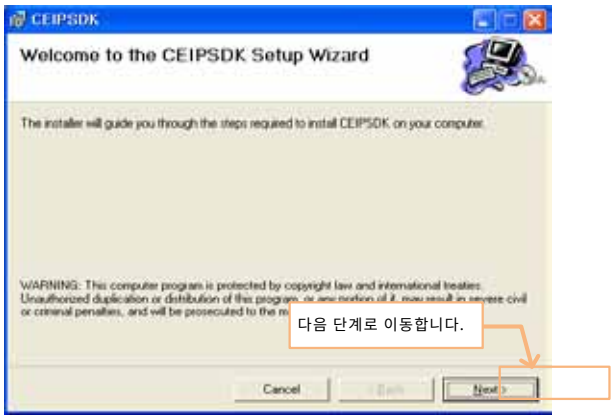
I cEIP 구동 환경 구축 안내

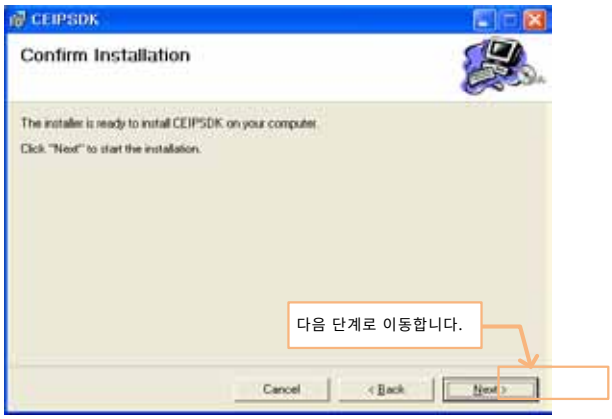
I.I cEIPSDK 설치

가. cEIPSDK 설치 실행 프로그램 파일을 실행하여 cEIPSDK 설치를 시작합니다.

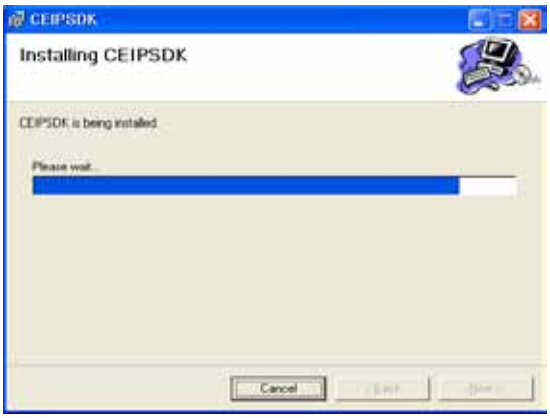


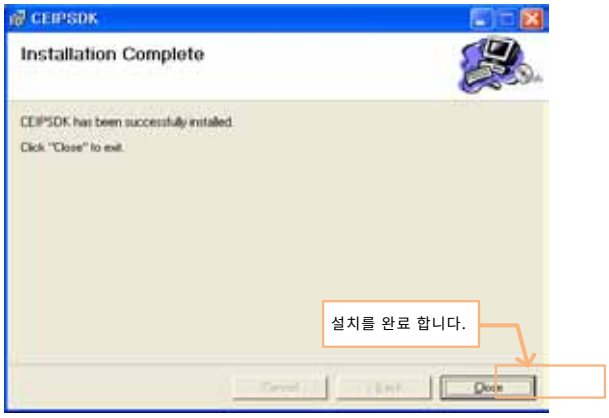
나. cEIPSDK Setup Wizard 가 실행되면 'Next' 버튼을 통하여 설치를 진행합니다.





다. 정상적으로 cEIPSDK 설치가 완료되면 'Close' 버튼을 통하여 설치 프로그램을 종료합니다.





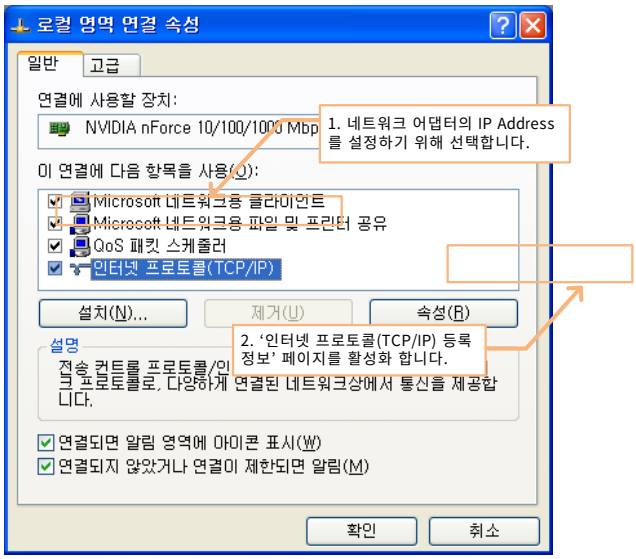
라. cEIPSDK 설치가 완료 후, 시스템을 재부팅 하면 자동으로 ceSDK Daemon Service Manager 가 실행됩니다. ceSDK Daemon Service Manager 는 cEIP 제품과 ceSDK 간에 인터페이스 역할을 담당합니다. 윈도우 작업 표시줄의 Tray 아이콘을 통해 ceSDK Daemon Service Manager 의 정상적인 실행 여부를 확인하실 수 있습니다.

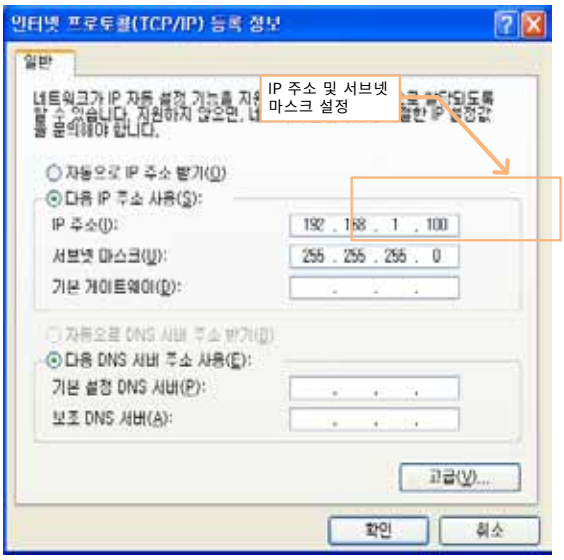
I.II 제어 시스템 IP Address 설정

cEIP 제품과 Ethernet 통신을 위한 네트워크 어댑터의 IP Address 를 설정합니다. cEIP 제품의 IP 사용 범위는 192.168.1.1 ~ 192.168.1.254 이며, 제어시스템의 IP Address 를 cEIP 제품의 가용 IP 범위에서 설정하여 주셔야 합니다. 다음 절차에 따라 IP Address 를 설정하여 주시기 바랍니다.

가. cEIP 제품과 통신할 네트워크 어댑터의 '로컬 영역 연결 속성' 페이지에서 '인터넷 프로토콜 (TCP/IP) '를 선택합니다.

나. 네트워크 어댑터의 IP Address 를 설정합니다. IP 주소는 192.168.1.1 ~ 192.168.1.254 범위에서 설정하시면 되며 서브넷 마스크는 255.255.255.0 으로 설정합니다.





I.III cEIP 노드 마스터 모듈의 IP Address 설정

cEIP 노드 마스터 모듈(ceNM-SE) 의 Hardware IP Address 를 설정합니다. Hardware IP 를 설정하기 위한 로터리 스위치가 제공되며, 설정 범위는 1 ~ 254 입니다.

	IP Setting	IP = MSB * 16 + LSB
	IP Range	1 ~ 254

2 개 이상의 노드 마스터 모듈 사용 시 Hardware IP 가 중복되지 않도록 주의하여 주시기 바랍니다. Hardware IP 가 중복될 경우 정상동작하지 않습니다.

I.IV cEIP 슬레이브 모듈의 ID 설정

cEIP 슬레이브 모듈은 모듈 ID 를 10 진 로터리 스위치를 통해 설정합니다. 모듈 ID 란 대상 노드 마스터 모듈에 연결된 슬레이브 모듈간에 서로를 구별할 수 있는 번호를 의미합니다. 따라서 대상 노드 마스터 내의 슬레이브 모듈간 모듈 ID 가 중복되지 않아야 합니다.

	ID Range	1 ~ 9

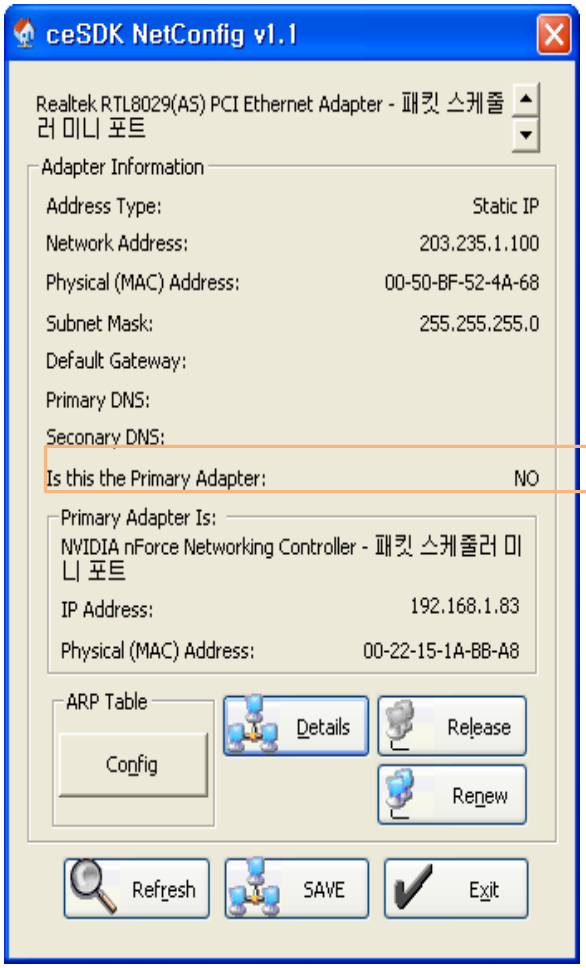
* 슬레이브 모듈의 ID 설정 범위는 1 ~ 9 입니니다. ID 를 ‘0’으로 설정 시 대상 슬레이브 모듈은 정상동작하지 않습니다.

I.V cEIP 대상 네트워크 어댑터 설정

ceSDK Daemon Service 는 주 네트워크 어댑터(Primary Adapter)를 통해 cEIP 제품과 통신을 하게됩니다. 따라서 제어 시스템에 2 개 이상의 네트워크 어댑터를 사용할 경우 cEIP 제품과 통신할 네트워크 어댑터를 설정하여 주셔야 하여, ceNetConfig 유틸리티를 통해 손쉽게 설정하실 수 있습니다.

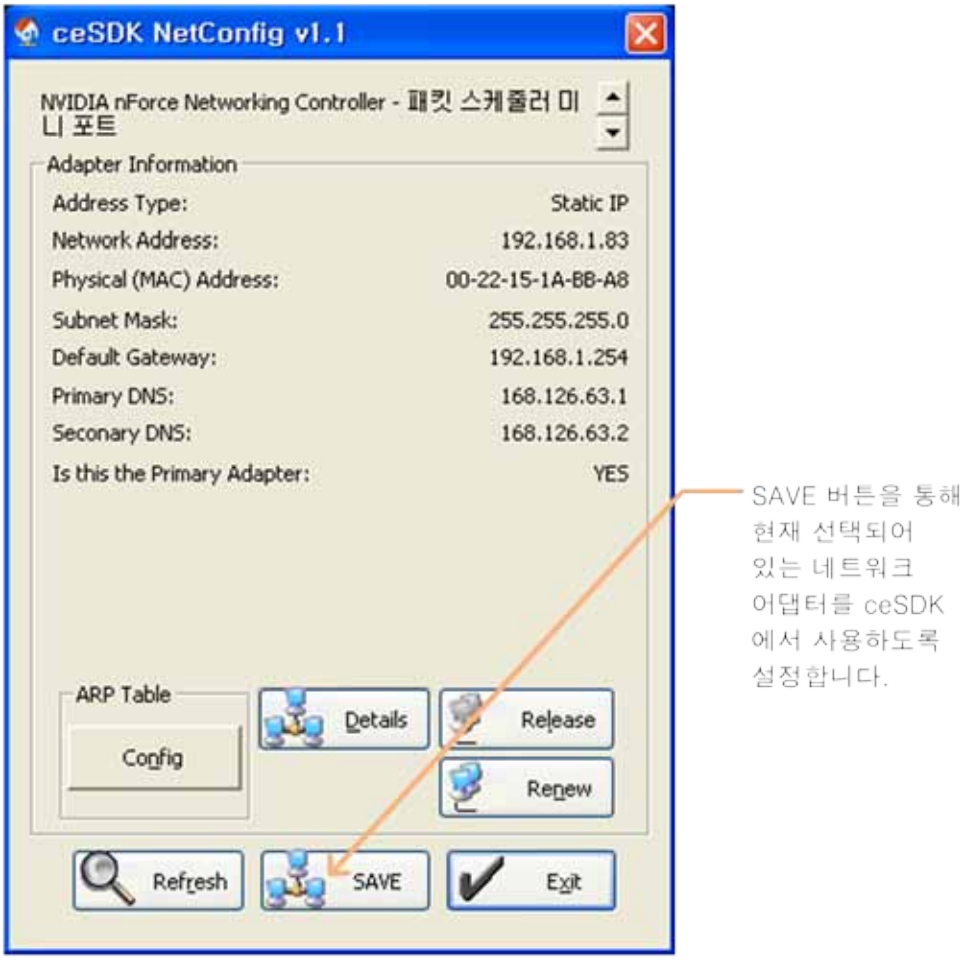
가. ceNetConfig 를 실행합니다. ceNetConfig 는 윈도우 바탕 화면의 'ceSDK' 바로가기 폴더를 통해 실행하거나 cEIPSDK 설치 폴더를 통해 실행하실 수 있습니다.

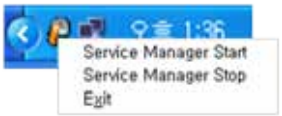
나. ceNetConfig 가 실행되면 자동으로 시스템의 네트워크 어댑터 정보를 얻어와 보여줍니다. 주 네트워크 어댑터는 'Is this the Primary Adapter: YES' 로 표시되며, ceSDK Deamon Service 는 주 네트워크 어댑터를 통해 cEIP 제품과 통신합니다.



여러 네트워크 어댑터가 존재할 경우 상·하 버튼으로 네트워크 어댑터를 변경할 수 있습니다.

다. cEIP 대상 네트웨크 어댑터를 변경하시려면 '네트워크 변경 버튼'으로 선택하신 후 'SAVE' 명령을 통해 설정하실 수 있습니다.



주의	ceSDK Daemon Service 는 제어 시스템 부팅 시 주 네트워크 어댑터를 통해 cEIP 제품과의 통신 환경을 준비합니다. 따라서, ceNetConfig 를 통해 cEIP 대상 네트워크 어댑터를 변경하시면 ceSDK Daemon Service 를 재시작 해주어 변경된 네트워크 어댑터를 통해 cEIP 제품과 통신할 수 있도록 하여야 합니다. ceSDK Daemon Service 를 재시작 하는 방법은 다음과 같습니다.  가. Tray 아이콘 위에서 마우스 우측 버튼을 눌러 'Service Manager Stop' 명령을 통해 ceSDK Daemon 을 정지합니다. 나. ceSDK Daemon 이 구동환경을 정지하기까지 1~2 초 정도의 시간이 필요합니다. 다. 'Service Manager Start' 명령을 통해 ceSDK Daemon 을 시작합니다. 라. ceSDK Damon 시작 시 ceNetConfig 를 통해 설정한 네트워크 어댑터를 통해 cEIP 제품과 통신을 합니다. * ceNetConfig 유틸리티의 자세한 설명은 'Appendix::B cEIP Utility - III ceNetConfig' 편을 참고하여 주시기 바랍니다.

Appendix

B

cEIP Utility

㈜ 커미조아는 고객님들께 다양한 cEIP 관련 유틸리티를 제공하여 편리한 개발환경을 지원하기 위해 노력하고 있습니다. cEIP 제품만의 다양한 유틸리티들은 편리한 기능 외에 고급 사용자 인터페이스를 제공하여 고객님들의 제품 개발에 도움을 드리고자 합니다.

본 장에서는 cEIP 제품 관련 각종 유틸리티에 대해 설명합니다. cEIP 노드 마스터 모듈의 Firmware 업데이트 및 Firmware 파일을 관리할 수 있는 cePowerFlasher, cEIP 네트워크 어댑터 정보를 관리할 수 있는 ceNetConfig, 연결되어있는 cEIP 모듈 정보를 한눈에 보여주는 ceNodeViewer, ceSDK 에러 코드에 대한 설명을 안내해주는 ceErrorLookup, cEIP 제어 및 테스트 프로그램인 ceMADIC 으로 구성되어 있습니다.

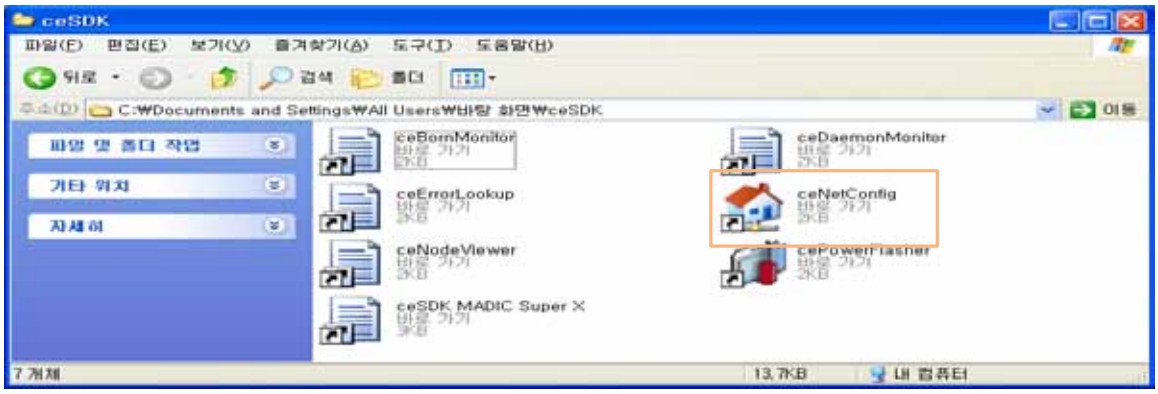
I cePowerFlasher

cePowerFlasher 는 cEIP 제품의 노드 마스터 모듈(ceNM-SE) Firmware 를 변경하기 위한 프로그램 입니다. 본 장을 통해 Firmware 업데이트 절차에 대해 안내합니다.

I.I cePowerFlasher 실행

펌웨어 업데이트 프로그램인 cePowerFlasher (ceNM-SE Module Firmware Update Program v2.0.0.x using ceSDK Library) 를 실행합니다.

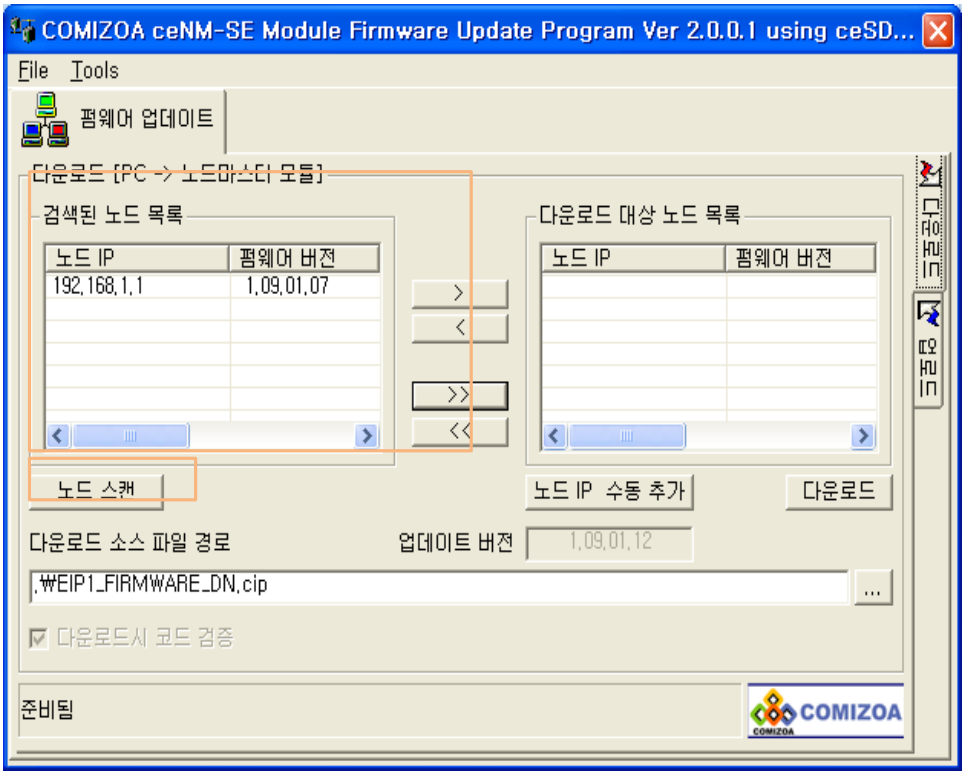
cePowerFlasher 는 cEIPSDK 설치 시 생성된 바탕화면의 [ceSDK] 폴더에서 'cePowerFlasher 바로가기' 를 통해 실행하거나,



프로그램 설치 경로의 cePowerFlasher 실행 파일을 실행합니다.

I.II 검색된 노드 목록 및 펌웨어 버전 확인

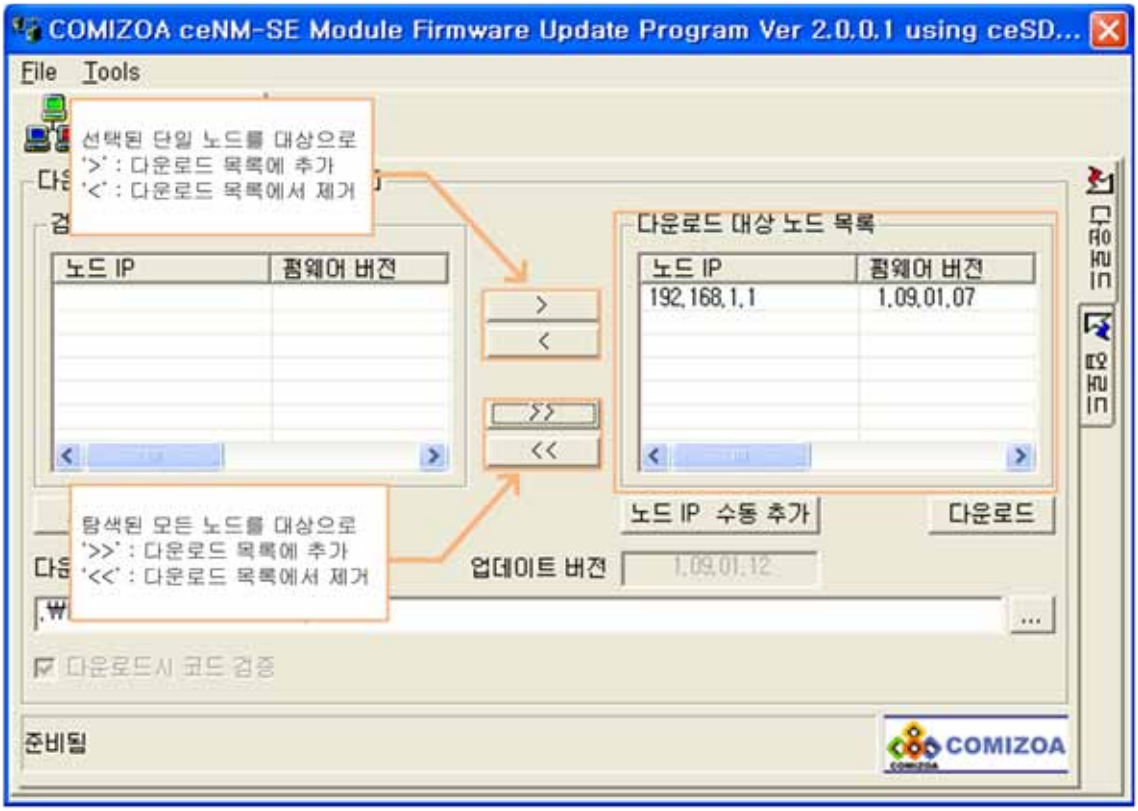
프로그램이 정상적으로 수행되면 아래와 같은 화면이 나타나며, 자동으로 노드 탐색을 수행합니다.



탐색된 결과는 '검색된 노드 목록' 의 리스트를 통해 확인하실 수 있으며, '노드 스캔' 명령을 통하여 노드를 재탐색 하실 수 있습니다.
'검색된 노드 목록'에는 해당 노드의 IP 주소 및 펌웨어 버전이 표시됩니다.

I.III 다운로드 대상 노드 선택 및 추가

'검색된 노드 목록' 의 리스트에서 펌웨어 업데이트를 수행하려는 노드를 선택한 뒤, '다운로드 대상 노드 목록'으로 옮기시면 다운로드 대상 노드 선택이 완료 됩니다.

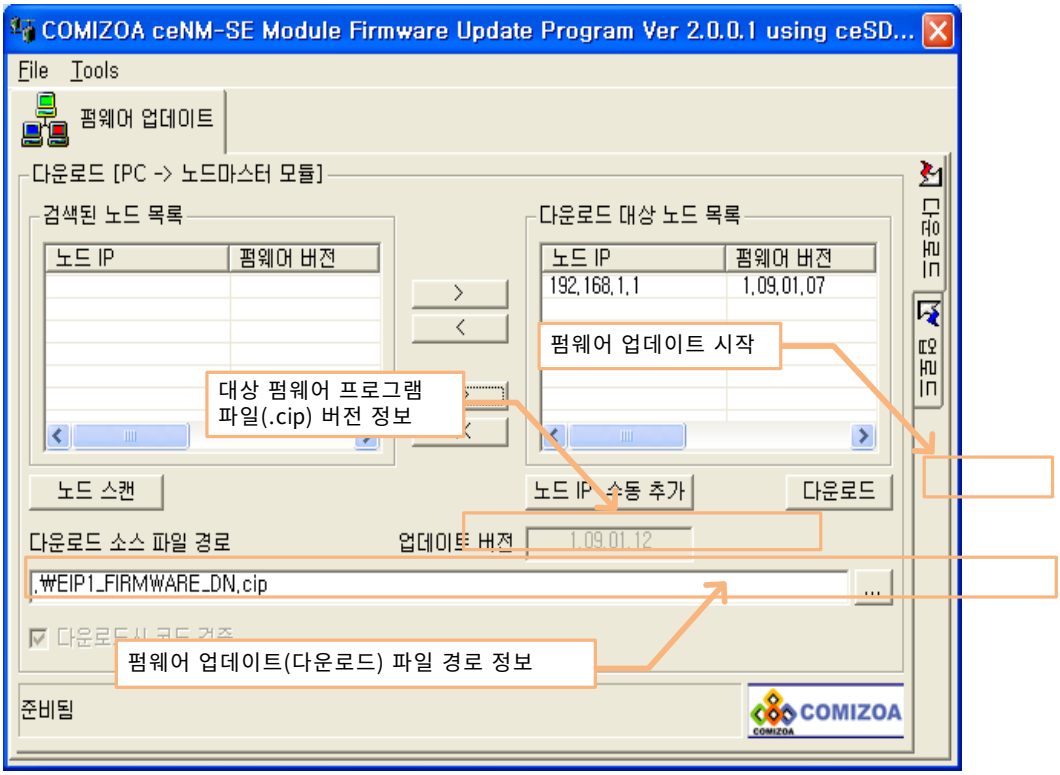


안내	노드가 검색되지 않은 경우 펌웨어 업데이트 안내.
	노드가 검색되지 않은 경우 '다운로드 대상 노드 목록' 하단의 '노드 IP 수동 추가' 명령을 통해 다운로드 대상 노드 IP 를 수동으로 입력해 주셔도 됩니다. 정상적인 경우라면 노드는 자동으로 검색이 되어 '검색된 노드 목록' 에 리스트됩니다. 다만 펌웨어 업데이트 작업이 중간에 취소되었거나, 실패한 경우에는 메인 펌웨어 코드가 정상적으로 동작하지 않고 부트로더 상태로 대기하고 있는 상황이 발생할 수 있습니다. 이 경우 수동으로 노드 IP 를 추가해서 해당 노드에 대한 펌웨어 업데이트가 가능합니다.

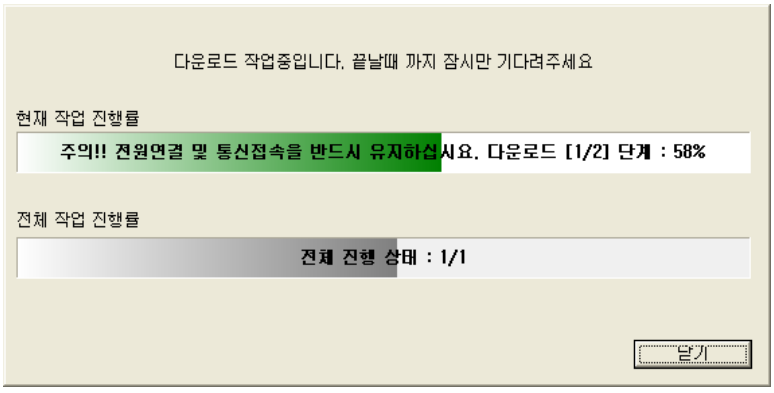
안내	펌웨어 업데이트 시 Active 및 Error LED 점등 상태에 대한 안내.
	참고로 펌웨어 업데이트 시작 시 부트로더 대기 상태로 되기 때문에 노드 마스터 모듈의 LED 중 Active 및 Error LED 는 함께 점등해 있습니다. 이 상태가 부트로더 대기 상태를 의미합니다. 부트로더 대기 상태는 노드 마스터 모듈에 24V 전원 공급 후 메인 펌웨어 프로그램이 정상적으로 수행되지 않는 경우나, 정상적으로 메인 펌웨어 프로그램동작 중 펌웨어 업데이트 프로그램을 사용해서 펌웨어 업데이트 시에 해당됩니다.

I.IV 펌웨어 업데이트 (다운로드)

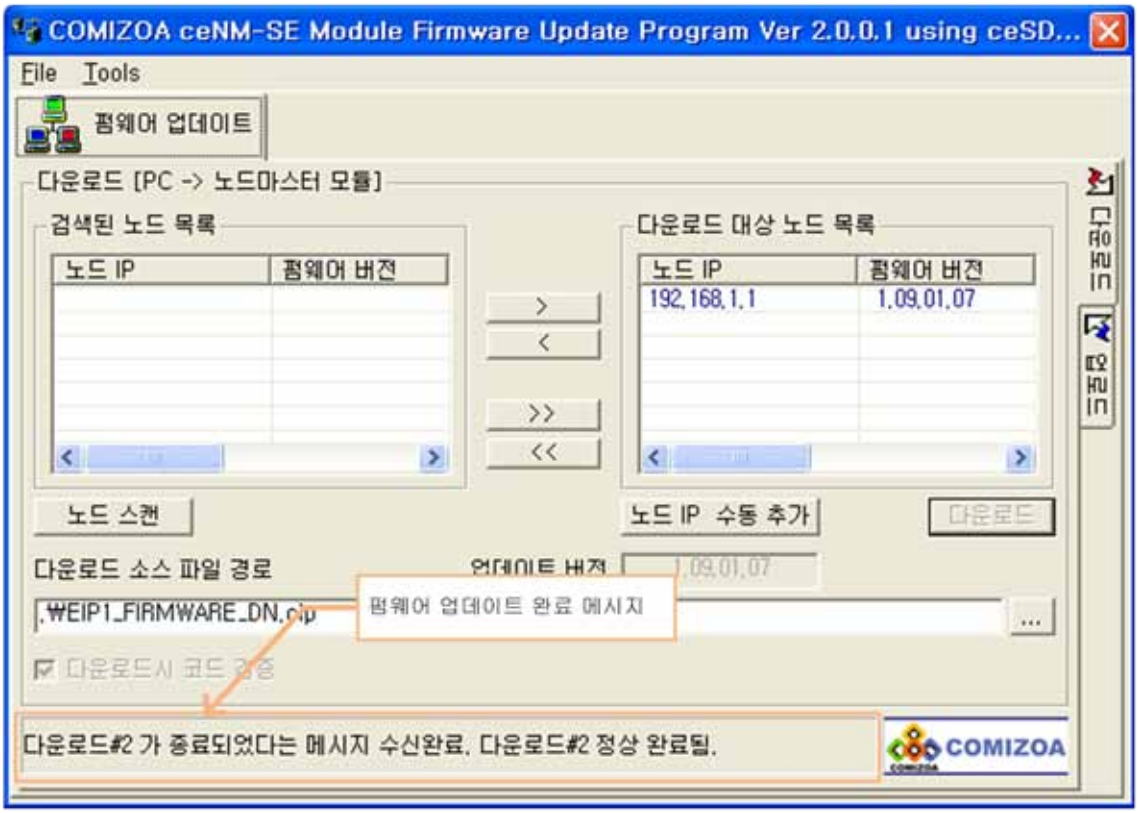
펌웨어 업데이트 대상 파일의 버전을 확인 후, '다운로드' 명령을 통해 펌웨어 업데이트를 진행합니다.



펌웨어 업데이트가 진행되면, 진행 상태 안내창을 통해 진행 상태를 확인하실 수 있습니다.



펌웨어 업데이트가 완료되면 업데이트 진행 창이 자동으로 사라지며 다음과 같이 다운로드 완료 메시지를 확인하실 수 있습니다.



안내	펌웨어 업데이트 파일의 경로 및 버전 정보 안내.
	프로그램 실행 시 자동으로 현재 경로에있는 EIP1_FIRMWARE_DN.cip 파일을 로드하며, 이 파일의 이름은 버전에 상관없이 동일하게 배포됩니다. 로드한 펌웨어 파일의 버전은 '업데이트 버전' 란을 통해 확인할 수 있습니다. 최신 펌웨어 업데이트 파일은 웹사이트 또는 기술지원팀을 통해 지원받으실 수 있습니다.

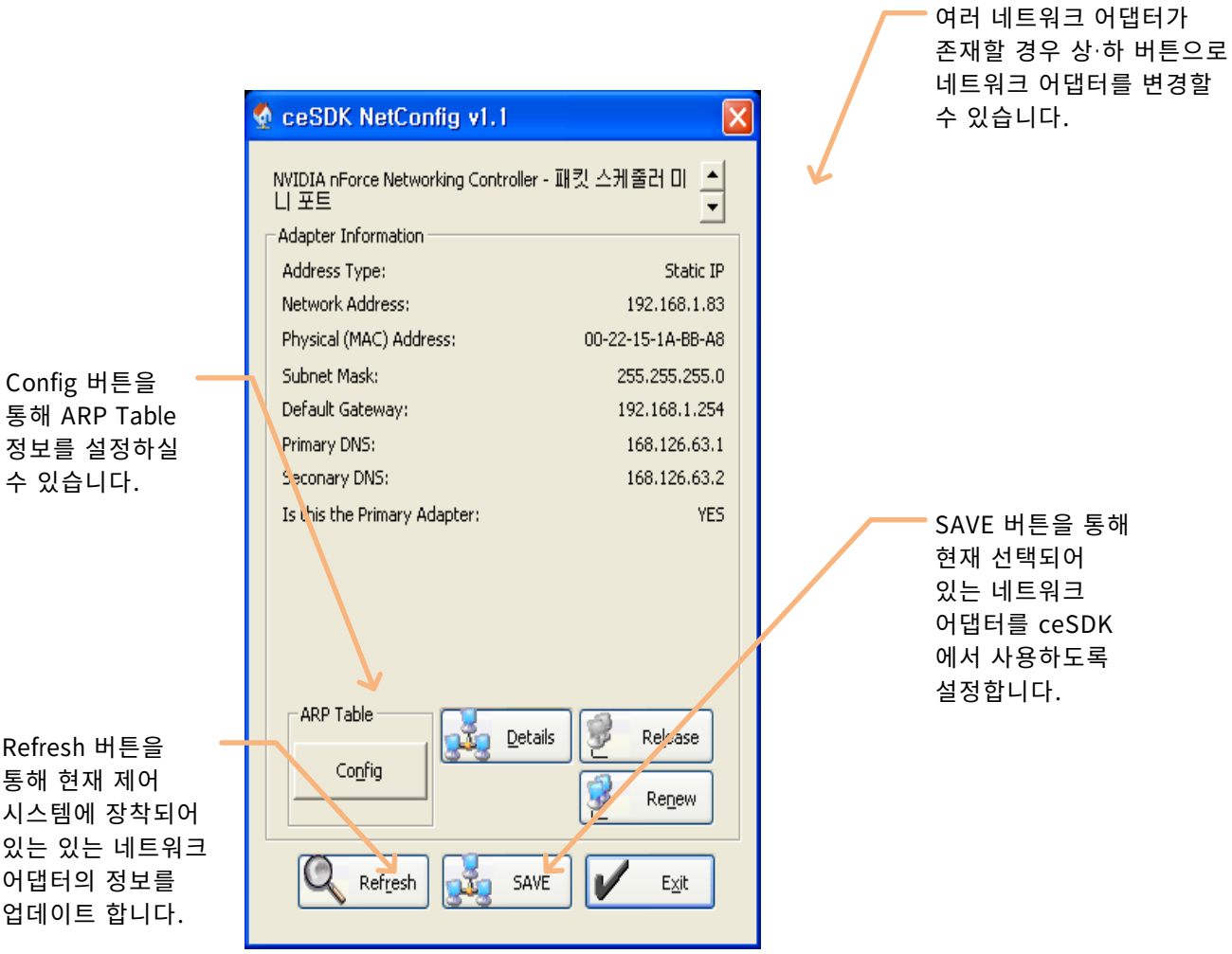
안내	부트로더 진입 체크 박스에 대한 설명.
	"부트로더 진입" 체크 박스는 디폴트가 Check 상태이며, ceNM-SE 노드마스터 모듈이 정상 동작하는 상태(Run)에서 부트로더로 진입하라는 트리거 명령을 전달하기 위해 필요한 경우 체크 합니다. 일반적으로 모듈을 정상적으로 사용하신다면 Check 상태를 유지하시면 됩니다. 다만, 업그레이드 실패 및 기타의 이유로 모듈이 이미 부트로더 모드로 머물러 있는 경우는 이 부분을 Uncheck 하셔야 부트로더로 재진입을 하지 않게 됩니다. 부트로더 모드와 정상동작 모드(Run 상태) 는 ceNM-SE 노드마스터 모듈의 전면부 패널에 Active LED 와 Error LED 상태를 통해 쉽게 확인하실 수 있습니다.

	펌웨어 업데이트가 완료 후 노드 마스터 모듈은 자동으로 재부팅이 되어 사용 가능한 상태가 되지만 다음과 같은 이유로 노드마스터 모듈의 24V 전원 공급을 Off 후 다시 On 시켜서 사용하시는 것을 권장합니다.
	아날로그 입력 모듈 및 확장시리얼 모듈은 전원이 Off 되었다 다시 On 되었을 때 완전히 초기화 되는 내부 상태 값을 가지고 있습니다. 따라서 펌웨어 업데이트 후에 자동으로 진행되는 Software Reset (Warm Booting) 으로는 이 상태 값들이 초기화 되지 않을 수 있습니다. 이로 인해 동작상에 문제는 없지만 펌웨어 업데이트 후에는 노드 마스터 모듈의 24V 전원을 Off 후 다시 On 시켜서(Cold Booting) Hardware Reset 을 해주실 것을 권장합니다.

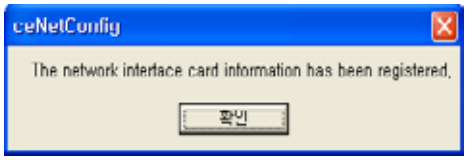
II ceNetConfig

ceNetConfig 응용프로그램은 다수의 네트워크 인터페이스를 갖는 시스템에서 ㈜커미조아의 cEIP 모듈과 통신을 하기 위해 네트워크 정보를 설정하는 유틸리티 프로그램입니다.

II.I 네트워크 어댑터 선택



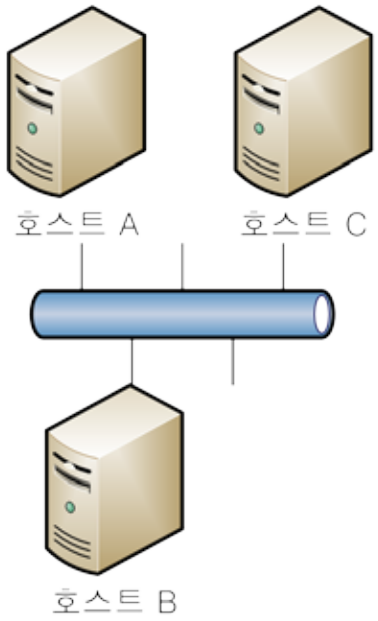
'SAVE' 명령을 통해 네크워크 어댑터 정보가 등록되며, 다음과 같은 메시지를 통해 정상적으로 정보가 등록되었을 확인할 수 있습니다.



II.II ARP Table 관리 및 유지 보수

ARP 는 Address Resolution Protocol 의 약자로서, TCP/IP 프로토콜 구성단에서 지역적으로 사용되는 레이어 중 L2 를 대상 하며, 이 레이어를 통해, MAC 주소와 IP Address 가 동일하게 구성이 되는데 같은 네트워크 내에서 호스트 및 라이터 간의 패킷 전달은 L2 주소를 가지고 전달이 됩니다. 이 내용은 기본 이더넷에 대한 배경 지식을 필요로 할 수 있습니다.

ARP 에 대한 패킷 전송의 절차는 다음과 같습니다. 이 내용에 대해서는 운용을 위한 주요 내용이 아니므로 생략하셔도 무방합니다.



네트워크 내에서 호스트 구성원 - 1

ARP 패킷 전송 절차

가. 호스트 A 와 호스트 B 는 같은 네트워크 클래스 내에서 존재합니다.

나. 호스트 A 는 호스트 B 에게 IP 패킷을 송신하려고 합니다.

다. 현재 호스트 A 는 호스트 B 의 IP 주소는 알고 있으며, 실제 호스트 B 의 네트워크 인터페이스 카드(Network Interface Card) 의 하드웨어적 물리 주소를 알고 있지는 않습니다.

라. 호스트 A 는 호스트 B 의 물리주소를 알기 위해서 ARP Request 라는 일정 형식의 패킷을 브로드 캐스트 방식으로 지역 네트워크의 모든 호스트에 송신합니다.

마. 여기서, 이 네트워크 구성원이었던 호스트 C 는 호스트 A 가 물리 주소를 알고 싶어하지는 않지만, 그것이 자신의 IP 주소는 대상이 아니기 때문에 해당 패킷을 무시합니다.

바. 동시에 호스트 B 는 호스트 A 로부터 송출된 ARP Request 패킷을 받아보니, 호스트 A 가 IP 주소를 통해 자신의 물리 주소(xx:xx:xx:xx:xx:xx) 를 알고 싶어하기 때문에 해당 물리 주소를 포함한 ARP Response 패킷을 주어 응답합니다.

사. 호스트 A 는 비로서 자신이 알고 있던 IP 주소를 통해 호스트 B 의 물리주소를 알게 되며, 호스트 A 는 이제 수신 측의 완전한 물리주소를 포함하여, 이더넷 패킷 프레임을 구성할 수 있게 됩니다.

아. 호스트 A 는 계속적으로 패킷을 호스트 B 로 보내게 될 경우 이제 위와 같은 ARP 에 의한 물리주소 탐색과정을 생략하고, 자신이 가지고 있는 ARP 캐쉬 테이블에서 해당 인터넷 주소와 반대측 물리주소를 찾아, 패킷을 송신하게 됩니다.

cEIP 제품의 NetConfig 응용프로그램에서 제공하는 ARP Table 관리 화면에서는 현재 운영체제가 ARP 를 통해 수신 받은 인터넷 주소와 해당 호스트의 물리 주소를 관리하는 역할을 합니다.

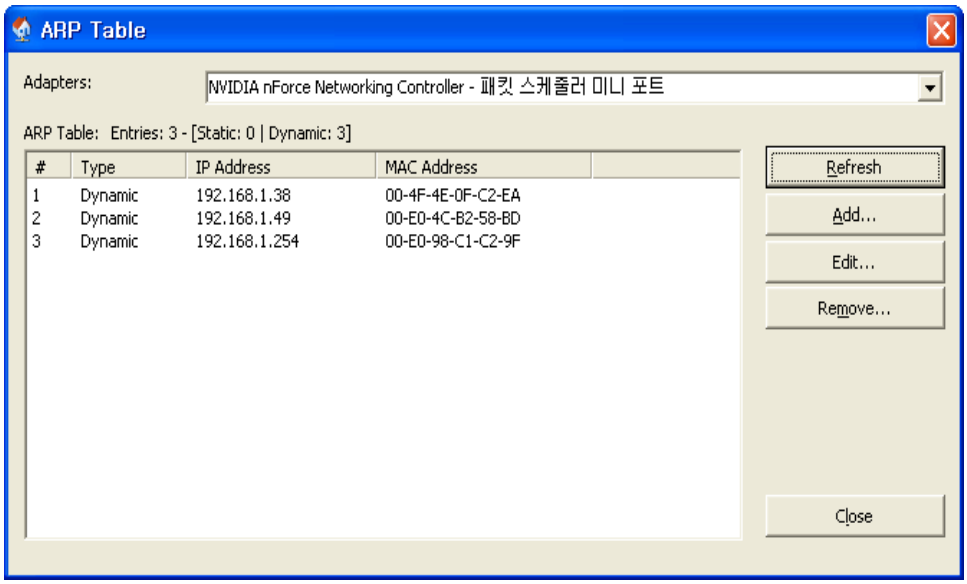
이 테이블의 데이터는 운영체제에서 지정된 시간 유지 후 삭제 될 수 있으며, 테이블 데이터의 갱신 역시 운영체제를 통해 그 관리가 귀속됩니다.

단, 호스트의 물리적인 주소를 직접적으로 관리하거나 해당 ARP 테이블의 IP Address 와 호스트의 MAC Address 를 직접적으로 제어해야 할 필요가 있는 경우에는 다음 그림에서 ARP 테이블의 호스트 구성원들을 갱신하거나, 추가, 수정 및 삭제 할 수 있습니다.

ARP 테이블을 실제 호스트의 지역 네트워크 상황과 관계없이, 임의로 추가 삭제 및 수정하게 되면, 네트워크 전송에 지연이나 문제를 발생시킬 수 있습니다. 숙련된 네트워크 운영자나 각 호스트간 네트워크 담당자 및 관할자에게 자세한 내용을 문의하기 바랍니다.

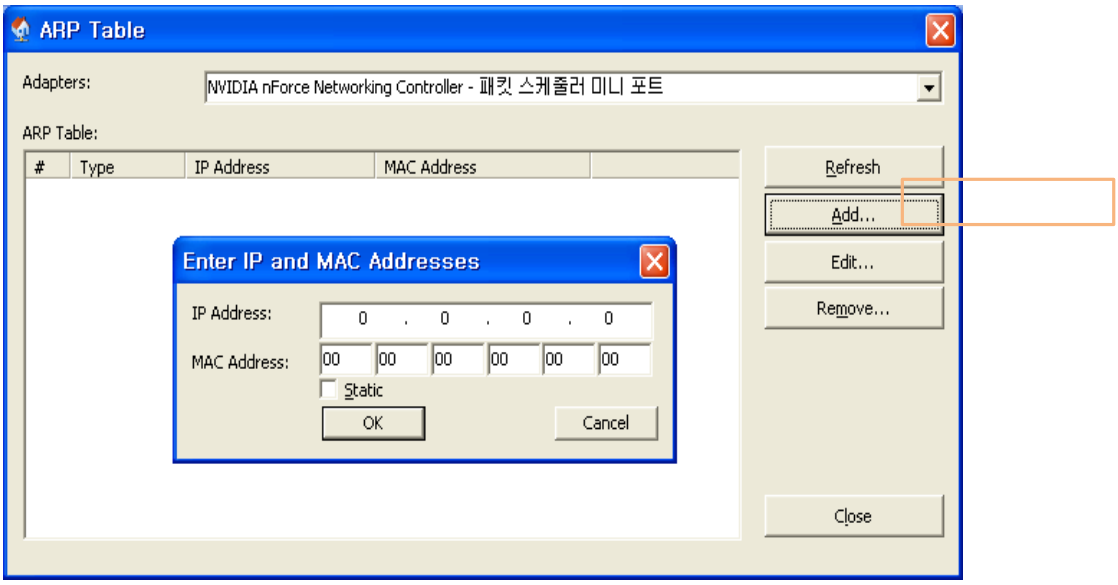
ARP 테이블 관리 방법에 대해서 안내하겠습니다.

가. ceSDK NetConfig 의 메인 화면에서 ARP Table 그룹 박스에 있는 'Config' 버튼을 눌러 다음화면을 활성화 합니다.



화면 상단부의 Adapters 에는 현재 Personal Computer 혹은 시스템에 장착되어 있는 네트워크 인터페이스 카드(Network Interface Card) 를 열거하게 됩니다. 1 개 이상의 네트워크 카드의 모델 이름 등으로 구분할 수 있습니다.

나. 우측 의 가장 첫번 째 위치한 ‘Refresh’ 버튼을 통해 현재 네트워크 어댑터의 ARP 캐쉬 테이블의 데이터를 화면 중앙에 갱신할 수 있습니다. ARP 캐쉬 테이블의 데이터를 갱신하는 경우는 가장 최신의 ARP 테이블을 확인하기 위한 용도로 사용할 수 있습니다.



다. ARP 테이블 관리 화면에서 ‘Add’ 명령을 통하여 위 그림과 같은 자식 윈도우를 화면에 표시하게 됩니다. 자식 윈도우는 실제 IP 주소와 호스트의 물리 주소를 직접 ARP 캐쉬 테이블 내에 추가할 수 있는 역할을 하며, 이 동작에서 하단 부의 ‘Static’ 을 선택하게 될 경우 이 IP 주소와 물리 주소는 운영체제가 임의로 삭제할 수 있는 대상이 아닌 고정 ARP 테이블 호스트 내역이 됩니다. 이 의미는 이 시스템이 종료될 때까지 ARP 테이블에 본 IP 주소와 MAC 주소의 데이터가 존속된다고 해석할 수 있습니다.

라. 이외의 Edit 및 Remove 버튼은 등록되어 있는 ARP 캐쉬 테이블의 IP 주소 혹은 호스트의 하드웨어 주소를 수정하거나 제거하는 역할을 합니다. 단지 이 수정 및 제거하는 역할일 뿐이며, 이 이외의 다른 추가 동작은 하지 않습니다.

마. 모든 ARP 테이블의 관리가 종료되면, ‘Close’ 버튼을 통해 ARP Table 관리 화면을 닫고 변경내용을 최종 적용하게 됩니다.

III ceNodeViewer

ceNodeView 는 연결된 원격 노드 정보를 보여줍니다.

노드 스캔을 위해서 하단 부분의 [Installed-Nodes] 에 연결된 물리적인 노드 개수를 입력하고, 아래 [Search-Time] 에 최대 검색 타임아웃 시간을 설정한 뒤 <Scan Nodes> 버튼을 클릭하시면 됩니다. 노드 개수는 최대 253 개까지 설정, 검색하실 수 있습니다.

사용자는 ceNodeView 프로그램을 통해 다음 정보를 확인하 실 수 있습니다.

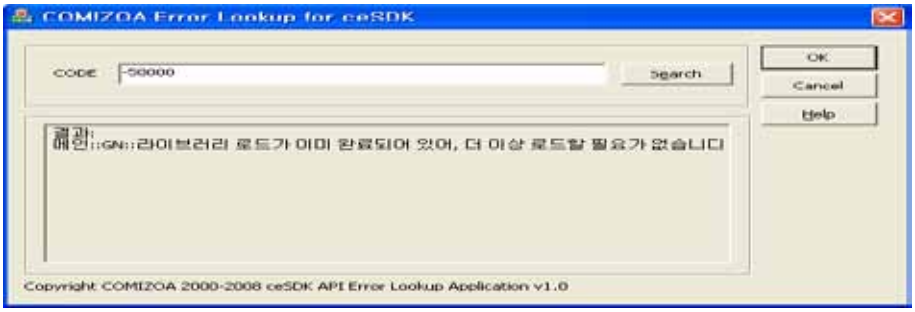
1. 원격 노드 및 슬레이브 모듈 개수.
2. 원격 노드의 IP Address, Firmware Version 정보.
3. ceSDK Version 정보.

IV ceErrorLookup

ceSDK 에서는 자세하고, 일괄적인 에러코드를 제공함으로써, 에러 상태에 대한 명확한 원인과 분석이 가능할 수 있도록 돕고 있습니다. ceErrorLookup 은 ceSDK 함수의 에러코드에 대한 설명을 안내합니다.

IV.I ceErrorLookup 을 통한 에러 코드 확인

'CODE 에디터 박스'에 확인하고자 하는 에러코드를 입력 후 'Search' 버튼을 누르면 해당 에러코드에 대한 설명을 확인하실 수 있습니다.



입력한 에러코드가 ceSDK 함수의 에러코드가 아닌 경우에는 다음과 같이 안내합니다.

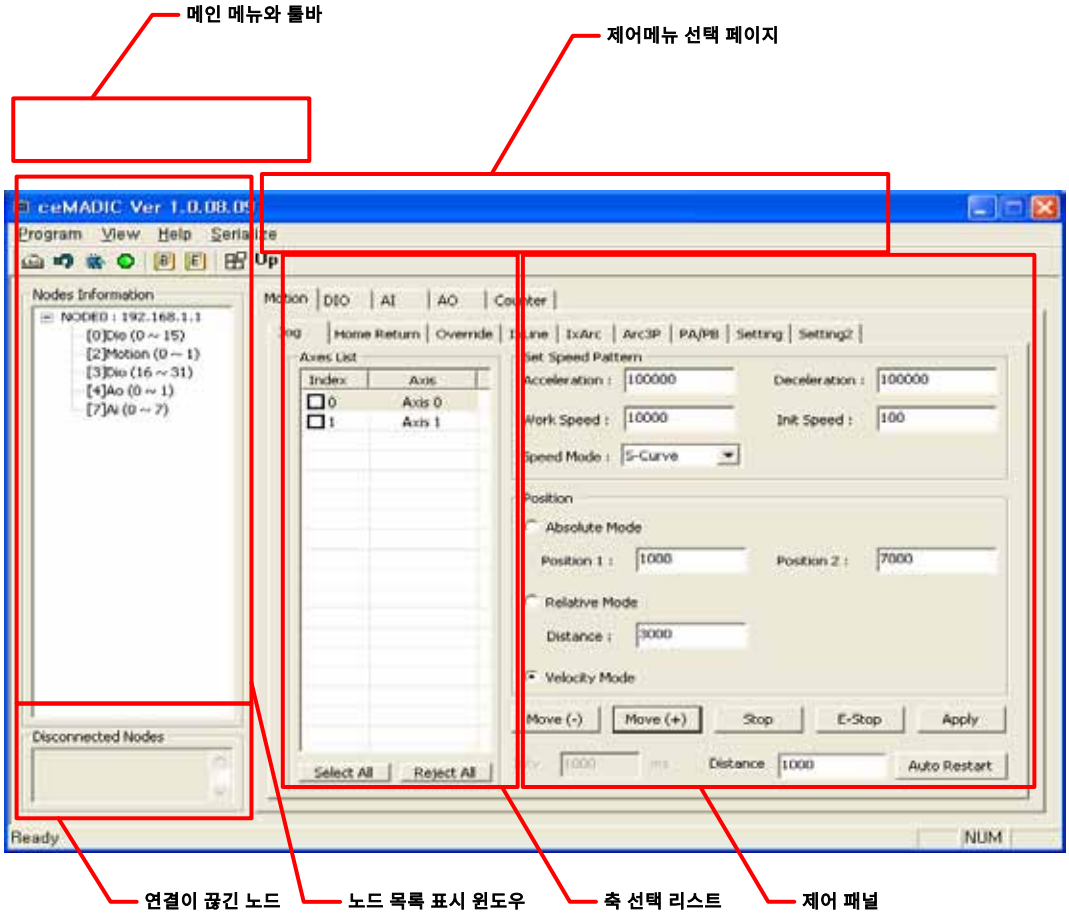
V ceMADIC

MADIC 을 사용하여 모션을 제어하고, 각 모듈의 상태를 확인하는 방법에 대해서 설명합니다. MADIC 은 cEIP 모듈을 제어하기 위해서 별도의 프로그램을 작성하지 않고도, cEIP 의 대부분의 기능을 사용하는 것이 가능합니다. 또한 상태 확인을 위한 윈도우를 표시하여 모션, DIO 의 상태를 바로 확인하고 테스트하는 것이 가능합니다.

V.I MADIC 사용자 인테페이스 설명

V.I.i User Interface

User Interface 는 메뉴와 노드 표시 창, 제어 패널로 이루어져 있습니다.



- Main Menus & Tool Bar : 메인 메뉴와 메인 메뉴를 빠르게 실행 할 수 있는 툴 바입니다.
- Control Mode Selection Tab : 구동할 모드를 선택하는 탭 페이지입니다.
- Control Panel : 실제 구동이나 설정을 편집하기 위한 패널입니다.

- Axis Selection List : 구동하거나 값을 설정할 축을 선택합니다. 체크가 된 축은 구동이나 설정이 적용될 축이며, 선택된 축은 편집을 위해 화면에 표시되는 축입니다.
- Node Tree : Tree 형식으로 노드와 하위 모듈을 표시합니다.
- Disconnected Nodes : 접속이 끊어진 노드를 표시합니다.

V.I.iiMain Menu

Main Menu 구성은 다음과 같습니다.

- Research Device : 노드를 재 탐색 합니다.
- Update All : 모든 축의 정보를 실제 모션에 설정된 값으로 업데이트 합니다.
- Motion : 탭 페이지에서 Motion 탭이 보이게 합니다.
- Dio : 탭 페이지에서 Dio 탭이 보이도록 합니다.
- AI : 탭 페이지에서 Analog Input 탭이 보이도록 합니다.
- AO : 탭 페이지에서 Analog Output 탭이 보이도록 합니다.
- Counter : 탭 페이지에서 Counter 탭이 보이도록 합니다.
- Exit : MADIC 을 종료합니다.

- Tool Bar : 툴바의 표시 여부를 결정합니다.
- Status Bar : Status Bar 의 표시 여부를 결정합니다.
- Status & Monitoring : Status Monitoring Window 의 표시 여부를 결정합니다.
- Motion Dio : Motion Dio 제어 윈도우의 표시 여부를 결정합니다.

- About Madic : MADIC 의 버전 정보를 표시합니다.

- Begin Serialize : Begin Serialize 버튼을 클릭 한 후 변경된 설정 값들에 대해서만 각 노드에 저장하는 모드를 시작합니다. End Serialize 를 클릭하면 변경된 값들만 각 모듈에 저장됩니다. 변경된 값들은 'Apply'버튼을 클릭하여 노드에 설정된 값들만 유효합니다.
- End Serialize : Begin Serialize 가 클릭 됐을 경우에는 변경된 값들만 저장을 시작합니다. Begin Serialize 가 클릭되지 않은 상태에서는 현재에 설정된 모든 값들을 노드의 Flash Memory 영역에 저장합니다. 플래쉬롬에 저장된 값은 전원이 꺼진 상태에서도 유지되며, 전원 Off 후 On 시 저장된 값들로 로드가 됩니다.

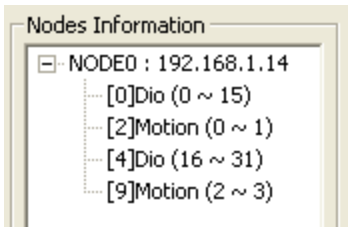
V.I.iii Tool Bar

메뉴의 기능을 간편하게 이용 할 수 있도록 툴바가 제공됩니다. 각 툴 버튼의 기능은 다음과 같습니다.

- Research Device : Research Device 메뉴를 실행합니다.
- Reset Device : 모든 노드들에 대해서 Reset 을 합니다.
- Status & Monitoring Window Show and Hide : Status & Monitoring Window 메뉴를 실행합니다.
- Motion Dio : Motion Dio Window 를 실행합니다.
- Begin Serialize : Begin Serial 메뉴를 실행합니다.
- End Serialize : End Serialize 메뉴를 실행합니다.
- Apply All Axis Mode : 전체 축의 설정값을 한번에 변경하고 싶을 경우에 Enable 시킵니다.
- Update All : Update All 메뉴를 실행합니다.

V.I.iv Node Tree

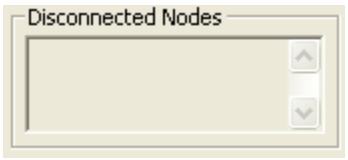
현재 연결되어 있는 원격 노드들의 정보를 표시합니다. 연결된 노드의 IP 를 표시하고, 트리 형식으로 노드에 있는 모듈들의 종류와 관련 정보를 표시합니다. 노드에 있는 컨트롤을 선택하여서 해당 모듈을 컨트롤 할 수 있는 제어판넬을 바로 화면에 표시하는 것이 가능합니다.



상위에 노드의 인덱스와 노드의 IP 가 표시되며, 하위에 모듈의 모듈 번호, 모듈의 타입, 모듈의 채널 범위가 표시됩니다.

V.I.v Disconnected Nodes

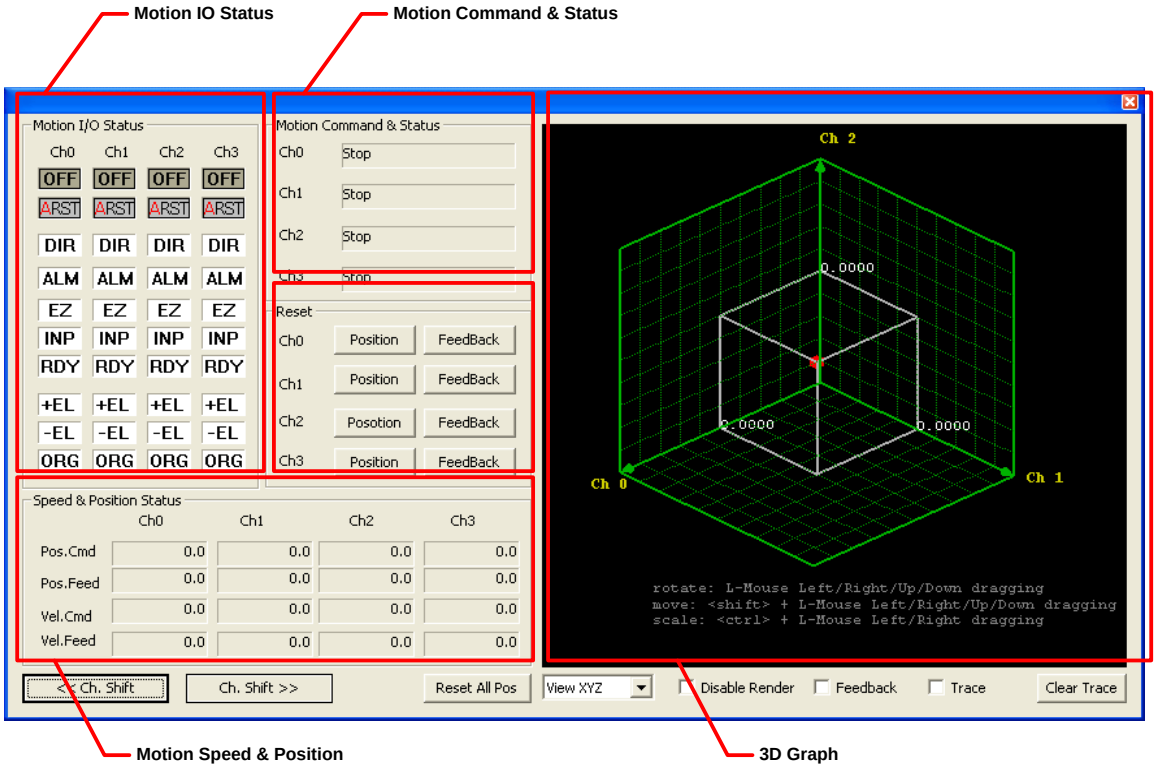
디바이스 로드 시 연결되어 있던 노드들 중 연결 상태가 불안정하거나, 다른 외부의 요인에 의해서 연결이 끊어진 노드의 ID 를 화면에 표시합니다. 끊어진 노드가 다시 복구 되면 화면에서 노드의 ID 가 사라집니다.



화면에 표시되는 숫자는 노드의 ID 이며, 노드의 ID 는 노드의 실제 IP 의 마지막 바이트 값입니다.
Ex) IP : 192.168.1.55 -> ID : 55

V.I.vi Status & Monitoring

모션의 상태를 한눈에 볼 수 있는 모니터링 윈도우 입니다. 모션의 Command Position, Feedback Position, Velocity Speed, Mio Status, 3D Graph 를 통한 모션의 위치 확인 기능을 제공합니다.



- Motion IO Status : 모션 IO 의 상태를 표시합니다.
- Motion Command & Status : 각 축의 현재 진행 중인 모션의 상태를 표시합니다.
- Motion Speed & Position : 모션의 Command Position 과 Feedback Position, Velocity Command Speed, Velocity Feedback Speed 를 축 별로 화면에 표시합니다.
- 3D Graph : 현재 화면에 표시 중인 4 축 중 3 축에 대해서 3 차원 공간에 이송 좌표를 표시합니다.

V.II MADIC 을 이용한 모션(Motion) 제어

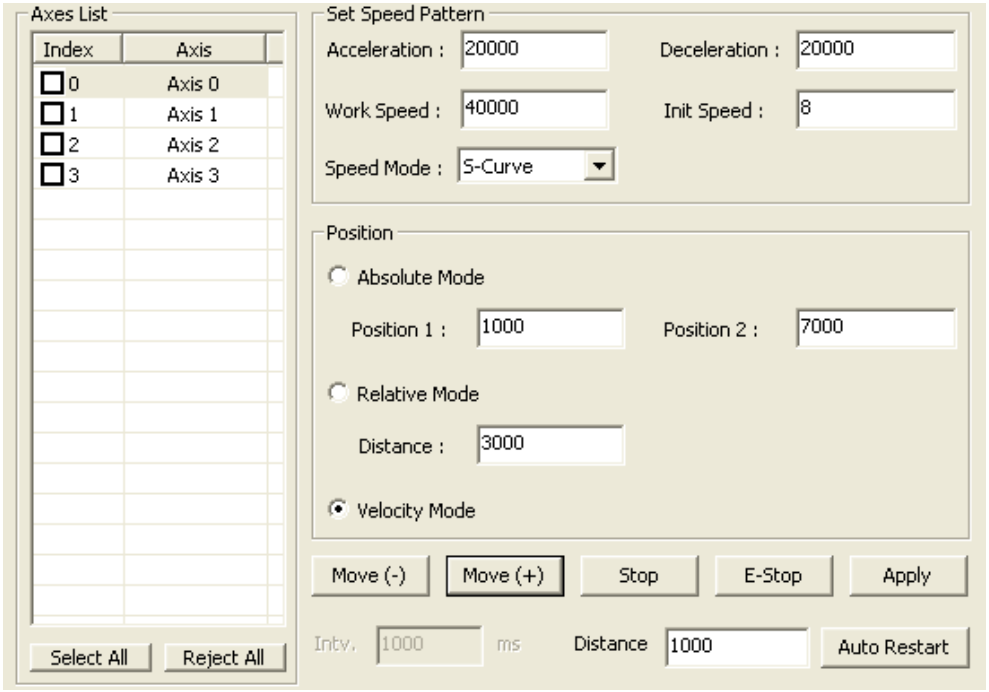
Motion 을 실제로 구동하거나 제어하는 부분입니다. Motion Tab 에서는 다축구동, 원점복귀, 오버라이드, 직선보간, 원호보간, 매뉴얼 펄사, 기본 모션 환경 설정이 가능합니다.

V.II.i Motion : Jog Panel

단축 및 다축을 절대좌표, 상대 좌표, 단방향 구동을 할 수 있습니다.

선택하는 축에 따라서 속도 및 이송 거리, 좌표 값이 다르게 설정됩니다. 현재 설정하고 있는 축이 왼쪽의 축 리스트에서 선택되어 있는지 주의 하셔야 합니다.

'Move'버튼을 통해서 구동하는 축은 축 리스트에서 선택된 축들이며, 각 축의 설정 값에 따라 다르게 구동합니다.



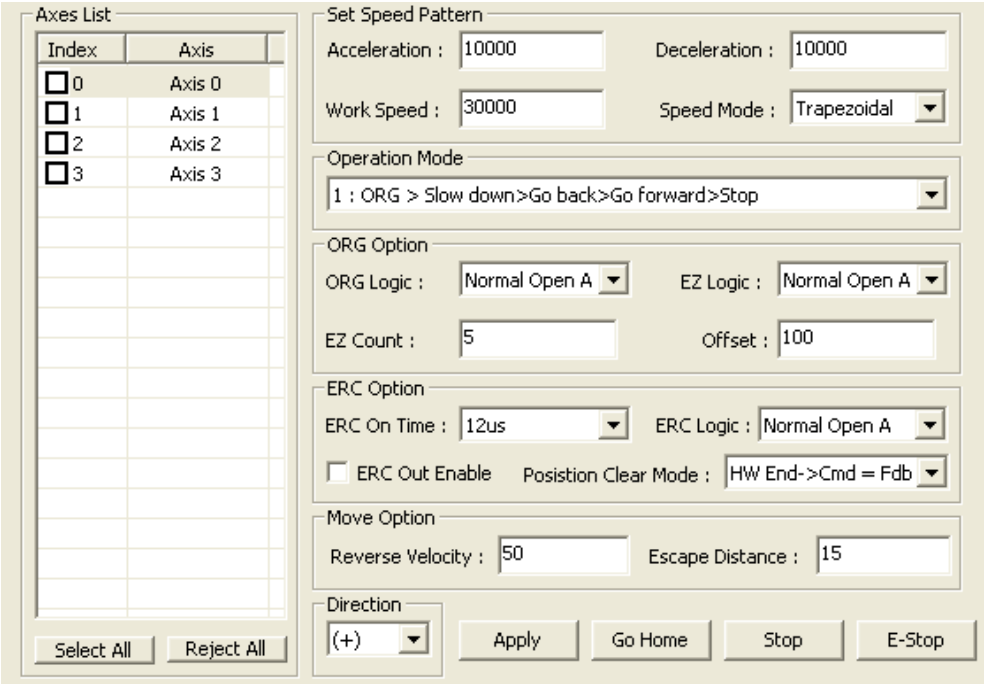
- Set Speed Pattern : 구동할 축의 속도 및 속도 패턴 모드를 설정합니다.
- Absolute Mode : 절대 좌표 이송을 할 좌표를 설정합니다. Position 1 은 Move(-)버튼에 클릭 시 이동할 좌표이며, Position 2 는 Move(+)버튼에 대응합니다.
- Relative Mode : 현재 위치를 기준으로 상대적으로 이송할 거리를 설정합니다. Move 버튼의 방향에 따라 이송하는 방향이 결정됩니다.
- Velocity Mode : Move 버튼의 방향에 따라 단방향으로만 이송합니다. 버튼을 누르고 있을 때만 이송하며, 버튼 올라오게 되면 정지하게 됩니다.
- Move : 왼쪽의 축 리스트에서 선택된 축들을 모두 각각의 설정 값에 따라서 구동시킵니다.
- Stop : 축 리스트에서 선택된 축들의 이송을 감속 후 정지시킵니다.

- E-Stop : 축 리스트에서 선택된 축들의 이송을 즉시 정지시킵니다.
- Apply : 축 리스트에서 선택된 축들의 설정된 값을 모션 모듈에 반영합니다. Serialize 를 통해서 변경된 값들을 저장하기 위해서 사용할 수 있습니다.
- Auto Restart : 선택된 축들을 지정된 상대 거리만큼 반복 이송을 합니다.

V.II.ii Motion : Home Return Panel

선택된 축들을 각축의 설정된 값에 따라 원점복귀 시킵니다.

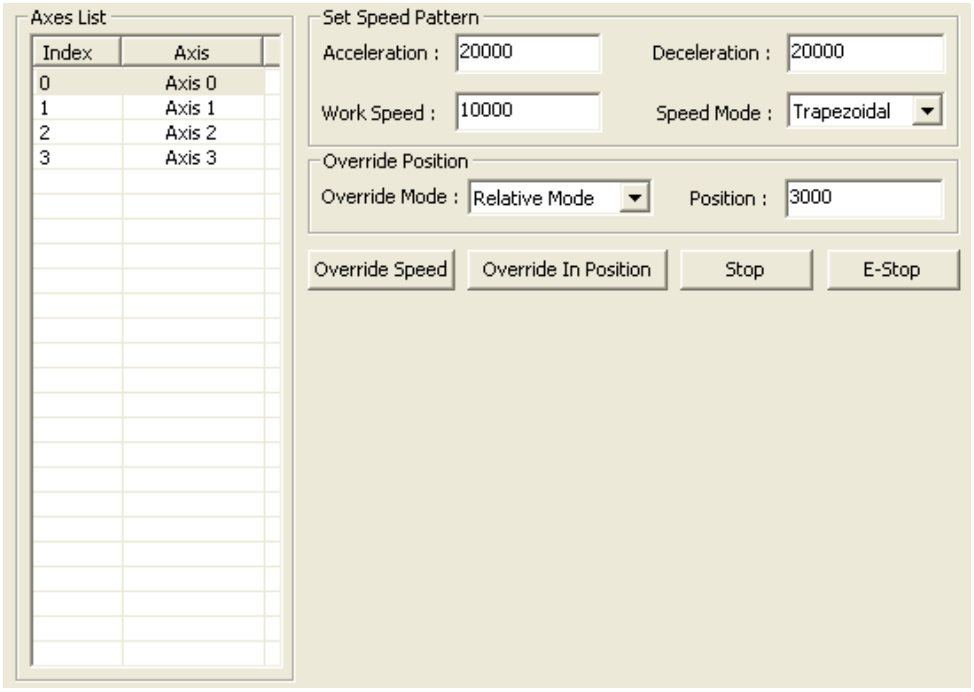
현재 설정하고 있는 축이 왼쪽의 축 리스트에서 선택되어 있는지 주의 하셔야 합니다.



- Set Speed Pattern : 원점복귀 할 축의 속도 및 속도 패턴 모드를 설정합니다.
- Operation Mode : 선택된 축의 원점복귀 모드를 설정합니다.
- ORG Logic : ORG 센서의 로직을 설정합니다.
- EZ Logic : EZ 센서의 로직을 설정합니다.
- EZ Count : 원점복귀 모드 중 EZ Count 에 따라 정지하는 모드에서 사용하는 값입니다.
- Offset : 원점복귀 완료 후 추가로 이송할 거리를 설정합니다.
- ERC Out Enable : 원점복귀 완료 후 ERC 신호 출력 기능 사용 여부를 설정합니다.
- ERC On Time : ERC 출력을 사용할 경우 출력을 유지할 시간을 설정합니다.
- ERC Logic : ERC 출력을 사용할 경우 출력 로직을 설정합니다.
- Position Clear Mode : Position Clear Mode 의 작동 방식을 설정합니다.
- Reverse Velocity : 원점복귀 역방향 속도(Vr)를 설정합니다.
- Escape Distance : 원점 탈출거리를 설정합니다.
- Direction : 원점복귀를 진행할 방향을 설정합니다.
- Apply : 설정된 값을 모션 모듈에 적용할 경우 사용합니다. 축 선택 리스트에서 선택된 축들만 설정 값들이 모션 모듈에 적용됩니다.
- Go Home : 선택된 축들에 대해서 원점복귀를 시작합니다.
- Stop : 선택된 축들의 원점복귀를 감속 후 정지합니다.
- E-Stop : 선택된 축들의 원점복귀를 즉시 정지합니다.

V.II.iii Motion : Override Panel

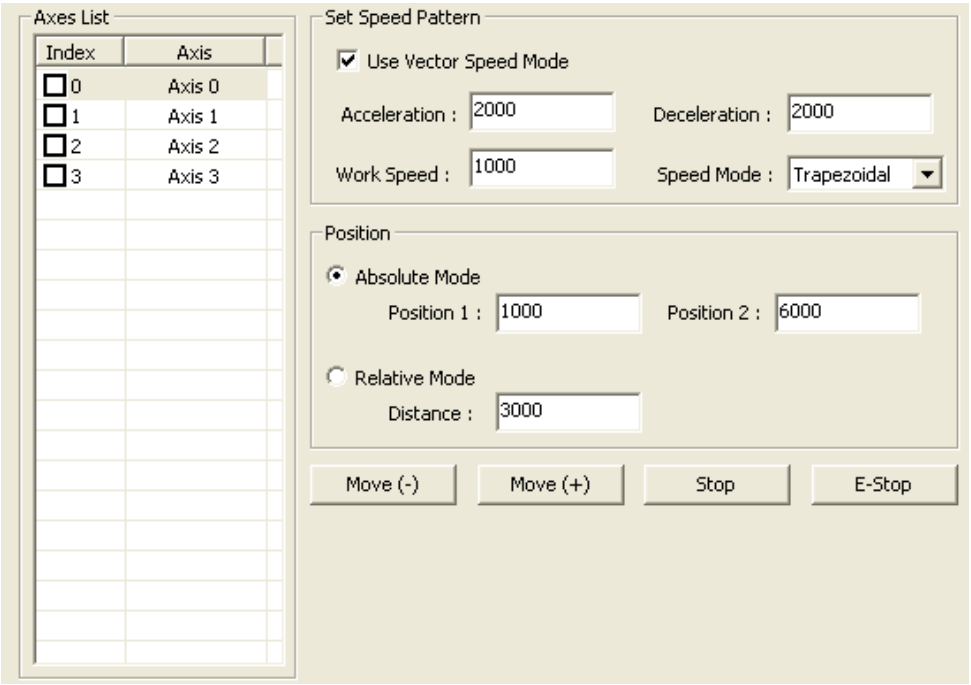
이송 중인 축의 속도 및 목표 좌표를 변경 할 때 사용합니다.



- Set Speed Pattern : 속도 오버라이드를 할 때 오버라이드 할 속도를 설정합니다.
- Override Mode : 위치 오버라이드를 할 경우 상대 좌표 모드인지 절대 좌표 모드인지 설정합니다.
- Position : 오버라이드 할 포지션을 설정합니다. 앞의 오버라이드 모드에 따라서 상대 거리인지 절대 좌표 인지 결정됩니다.
- Override Speed : 선택된 축을 속도 오버라이드 합니다.
- Override In Position : 선택된 축을 거리, 위치 오버라이드 합니다.
- Stop : 선택된 축을 감속 후 정지 합니다.
- E-Stop : 선택된 축을 즉시 정지합니다.

V.II.ivMotion : IxLine Panel

여러 축에 대한 직선보간 구동을 합니다. 동일 노드 상에 있는 모든 모션 축들에 대해서 사용하는 것이 가능합니다. 직선 보간을 구동하기 전에 축 리스트에서 선택된 축들이 동일 노드에 있는 축들인지 주의하셔야 합니다.

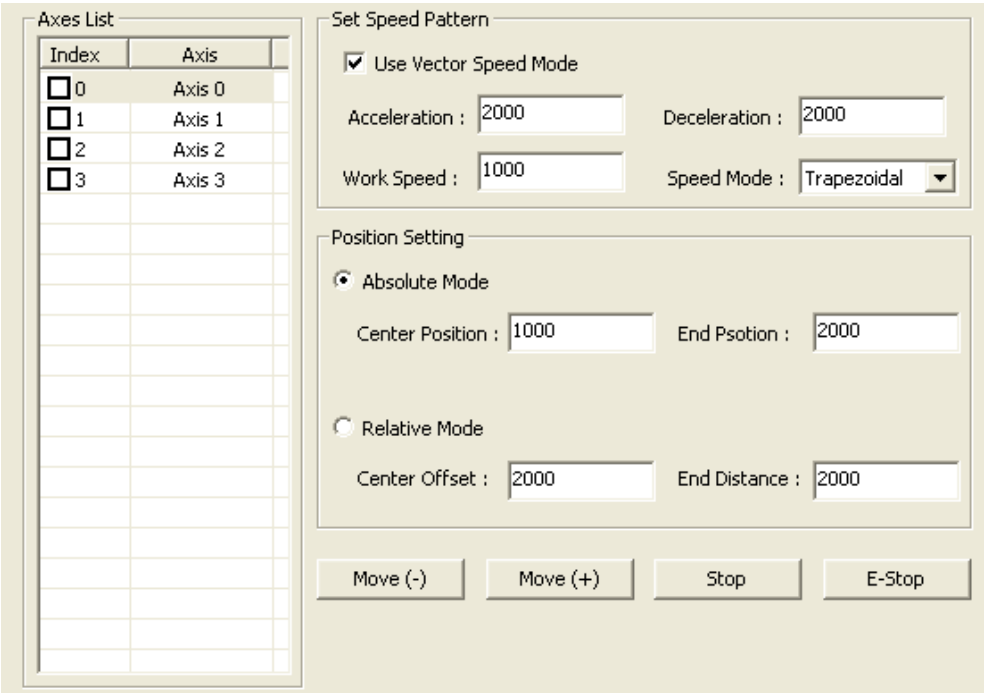


- Use Vector Speed Mode : 구동할 축들의 속도 패턴을 벡터 모드와 마스터 모드 중 하나로 설정합니다. 마스터 모드를 사용할 경우 속도 패턴의 각 값들은 Ratio(%)로 설정됩니다.
- Set Speed Pattern : 구동할 축들의 속도 패턴을 설정합니다. 속도 패턴은 선택된 축들에 모두 적용되며, 벡터 모드의 사용 여부에 따라 속도 혹은 비율로 설정되기 때문에 주의가 필요합니다.
- Absolute Mode : 선택된 축이 이송할 절대좌표를 설정합니다. 값들을 입력 할 때 축 리스트에서 선택된 축들마다 다르게 위치값이 저장됩니다.
- Relative Mode : 선택된 축이 이송할 상대거리를 설정합니다. 값들을 입력 할 때 축 리스트에서 선택된 축들마다 다르게 위치값이 저장됩니다.
- Move : 선택된 축들을 직선보간 이송 시킵니다. ‘Move(-)’버튼의 경우 Absolute Mode 에서는 Position 1 번의 좌표로 이송이며, Relative Mode 의 경우 (-)방향에 대한 상대 거리 이송입니다.
- Stop : 보간 이송 중인 축들을 감속 후 정지 시킵니다.
- E-Stop : 보간 이송 중인 축들을 즉시 정지 시킵니다.

V.II.v Motion : IxArc Panel

동일 모듈 내에 있는 두 축에 대해서 원호보간 이송을 합니다.

축 리스트에서 선택된 축들이 동일 모듈 내에 있는 축들인지 주의 하셔야 합니다.

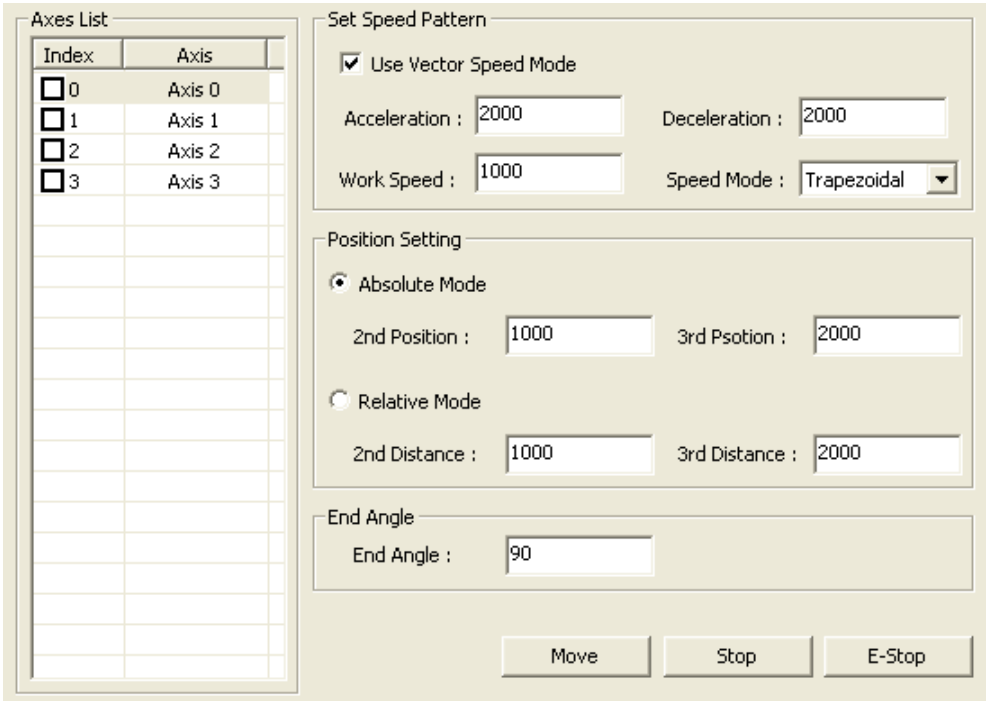


- Use Vector Speed Mode : 벡터 모드의 사용 여부를 결정합니다. 마스터 모드를 사용할 경우 속도 패턴의 각 값들은 Ratio(%)로 설정됩니다.
- Set Speed Pattern : 구동할 축들의 속도 패턴을 설정합니다. 속도 패턴은 선택된 축들에 모두 적용되며, 벡터 모드의 사용 여부에 따라 속도 혹은 비율로 설정되기 때문에 주의가 필요합니다.
- Absolute Mode : 선택된 축이 이송할 절대좌표를 설정합니다. 값들을 입력 할 때 축 리스트에서 선택된 축들마다 다르게 위치값이 저장됩니다.
- Relative Mode : 선택된 축이 이송할 상대거리를 설정합니다. 값들을 입력 할 때 축 리스트에서 선택된 축들마다 다르게 위치값이 저장됩니다.
- Move : 선택된 축들을 원호보간 이송 시킵니다. (-)버튼과 (+)버튼에 따라 원호를 그리는 방향이 결정됩니다.
- Stop : 원호 보간 이송 중인 축들을 감속 후 정지시킵니다.
- E-Stop : 원호 보간 이송 중인 축들을 즉시 정지시킵니다.

V.II.vi Motion : IxArc3P Panel

동일 모듈 내에 있는 두 축에 대해서 세점을 지나는 원호보간 이송을 합니다.

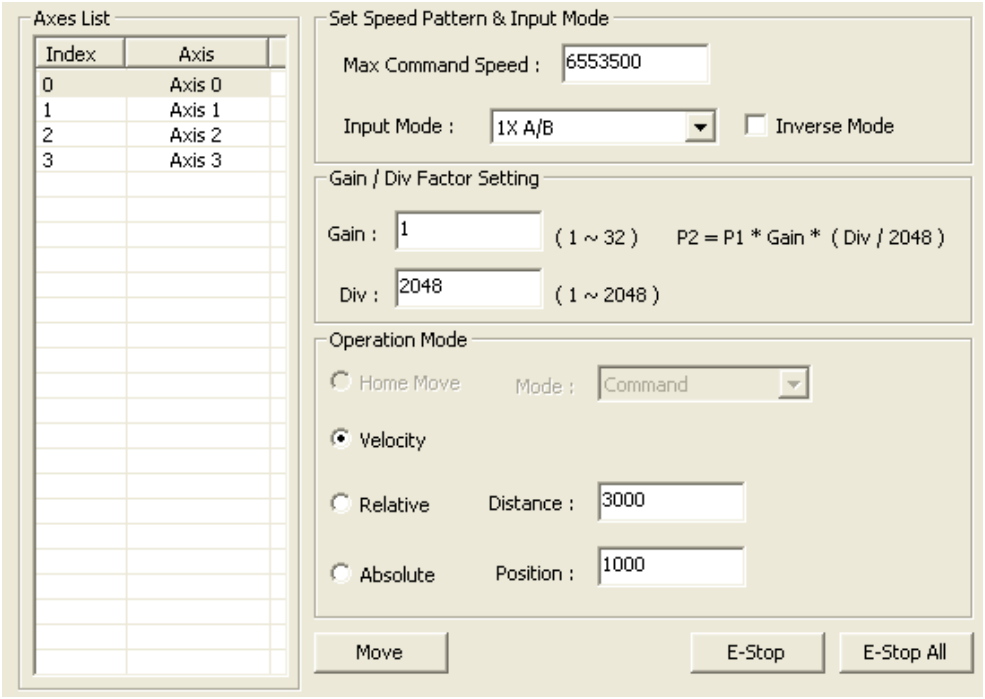
축 리스트에서 선택된 축들이 동일 모듈 내에 있는 축들인지 주의 하셔야 합니다.



- Use Vector Speed Mode : 벡터 모드의 사용 여부를 결정합니다. 마스터 모드를 사용할 경우 속도 패턴의 각 값들은 Ratio(%)로 설정됩니다.
- Set Speed Pattern : 구동할 축들의 속도 패턴을 설정합니다. 속도 패턴은 선택된 축들에 모두 적용되며, 벡터 모드의 사용 여부에 따라 속도 혹은 비율로 설정되기 때문에 주의가 필요합니다.
- Absolute Mode : 선택된 축이 이송할 절대좌표를 설정합니다. 값들을 입력 할 때 축 리스트에서 선택된 축들마다 다르게 위치값이 저장됩니다.
- Relative Mode : 선택된 축이 이송할 상대거리를 설정합니다. 값들을 입력 할 때 축 리스트에서 선택된 축들마다 다르게 위치값이 저장됩니다.
- End Angle : 원호 보간 이송을 할 때 회전할 각도 및 회전 방향을 결정합니다.
- Move : 선택된 축들의 원호 보간 이송을 시작합니다.
- Stop : 원호 보간 이송 중인 축들을 감속 후 정지시킵니다.
- E-Stop : 원호 보간 이송 중인 축들을 즉시 정지시킵니다.

V.II.vii Motion : PA / PB Panel

PA/PB 구동 페이지입니다. 선택된 축을 외부 입력 펄스에 의해서 구동하도록 합니다. Gain 및 Div 값, Input Mode 를 통해서 입력 대비 출력 펄스의 수를 조절 할 수 있습니다.

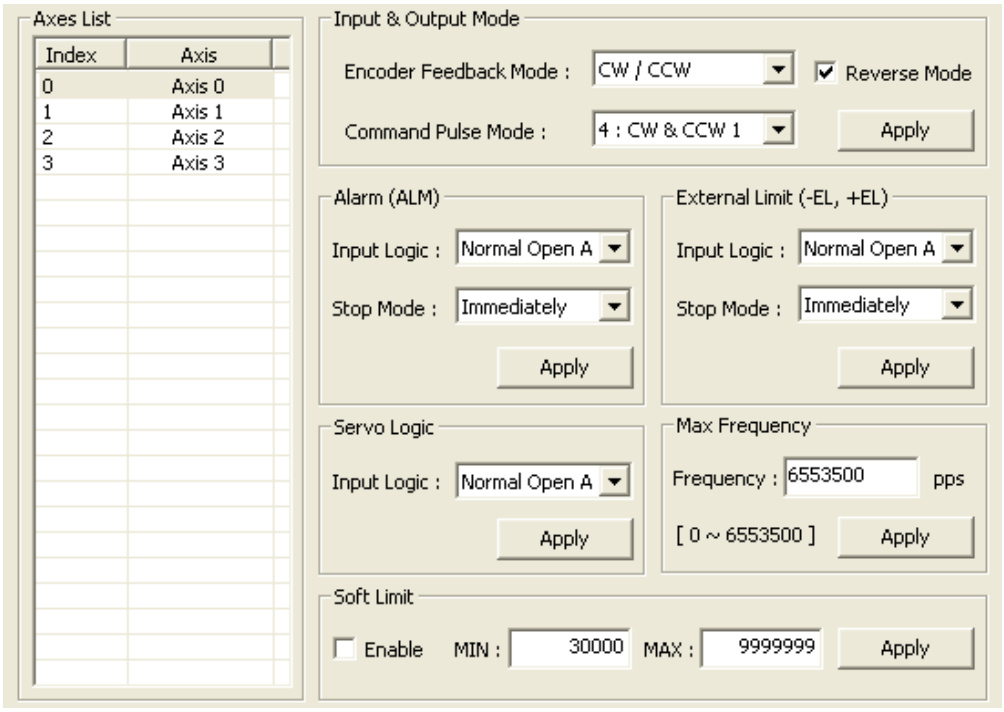


- Max Command Speed : 최대로 받을 수 있는 외부 입력 펄스 속도를 설정합니다. 이 값이 너무 작을 경우 정상적인 매뉴얼 펄사 구동이 어려울 수 있습니다.
- Input Mode : 외부 입력 펄사의 채배를 설정합니다.
- Inverse Mode : 입력 펄스의 반대로 이송 하고자 할 경우 설정합니다.
- Gain /Div Factor Setting : PA/PB 입력 펄스 대비 Command 펄스 수의 비를 설정합니다.
- Home Move : 외부 입력 펄스에 맞춰서 원점복귀를 진행합니다.
- Velocity : 외부 입력 펄스에 동기 해서 구동합니다.
- Relative : 외부 입력 펄스에 맞춰서 상대거리 이송을 합니다.
- Absolute : 외부 입력 펄스에 맞춰서 절대좌표 이송을 합니다.
- Move : 선택된 축을 Operation Mode 에 따라 PA/PB 를 시작합니다.
- E-Stop : 선택된 축을 즉시 정지 시키며, PA/PB 를 해제합니다.
- E-Stop All : 모든 축을 즉시 정지 시키며, 모든 축의 PA/PB 를 해제합니다.

V.II.viii Motion : Settings Panel

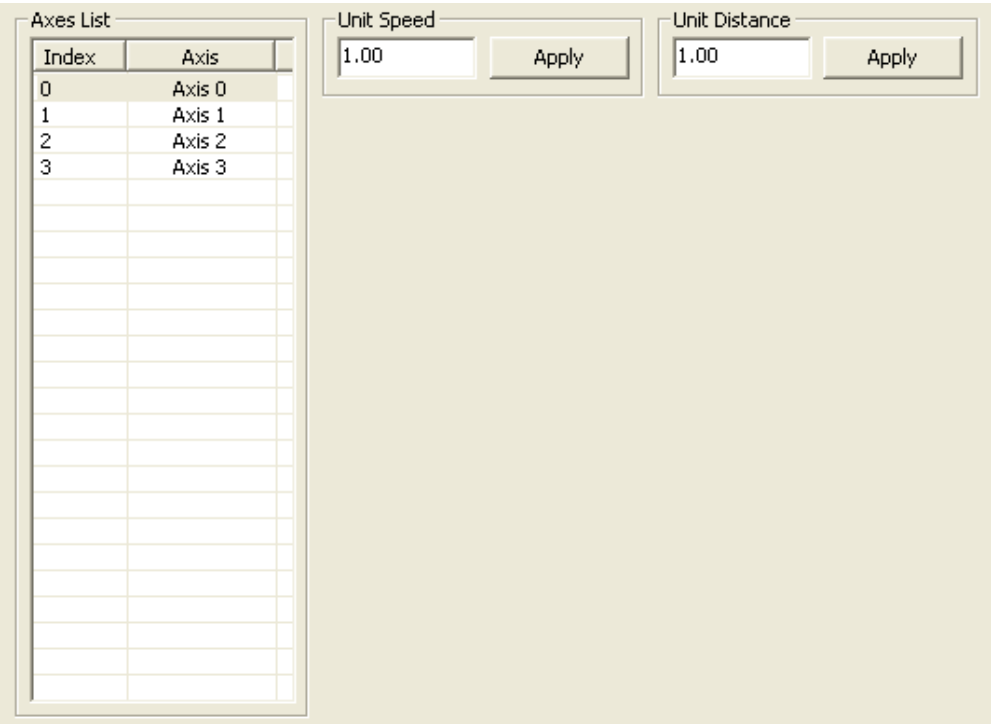
모션 구동에 필요한 기본적인 설정을 할 수 있습니다. 설정 값들은 각 축에 따라 다르며, 'Apply'버튼을 클릭하지 않았을 경우 설정이 적용되지 않습니다.

가. Settings Panel 1



- Input & Output Mode : 선택된 축의 인코더로 입력 받는 펄스의 채배와 출력 펄스 모드를 설정합니다.
- Alarm : 선택된 축의 알람 로직과 알람이 발생했을 경우에 축의 정지 모드를 설정합니다.
- External Limit : 선택된 축의 EL 신호의 로직을 설정하며, EL 신호가 감지 되었을 때 축의 정지 모드를 설정합니다.
- Servo Logic : 선택된 축의 서버 신호 로직을 설정합니다.
- Max Frequency : 선택된 축의 출력 펄스의 최대 출력 주파수 값을 설정합니다.
- Soft Limit : 선택된 축의 소프트웨어 EL 을 설정합니다.

나. Settings Panel 2



- Unit Speed : 선택된 축의 Unit Speed 를 설정합니다. 설정 값에 따라 Jog 의 속도 패턴과 Home Return 의 속도 패턴의 값이 영향을 받아 변경됩니다.
- Unit Distance : 선택된 축의 Unit Distance 를 설정합니다. 설정 값에 따라 Home Return 의 Esc Distance 값 등이 영향을 받아 변경됩니다.

V.II.ixMotion Digital I/O

모션 모듈은 한 축당 3 개의 Digital Input Channel 과 3 개의 Digital Output Channel 을 가지고 있습니다. 각 채널은 Dio 모듈과는 달리 입력과 출력을 설정 할 수 없습니다. 녹색 채널은 Digital Input 을 나타내며, 파란색 채널은 Digital Output 을 나타냅니다.

- << : 현재 표시하고 있는 채널 그룹의 앞 쪽 4 개의 채널 그룹을 표시합니다.
- >> : 현재 표시하고 있는 채널 그룹의 뒤 쪽 4 개의 채널 그룹을 표시합니다.

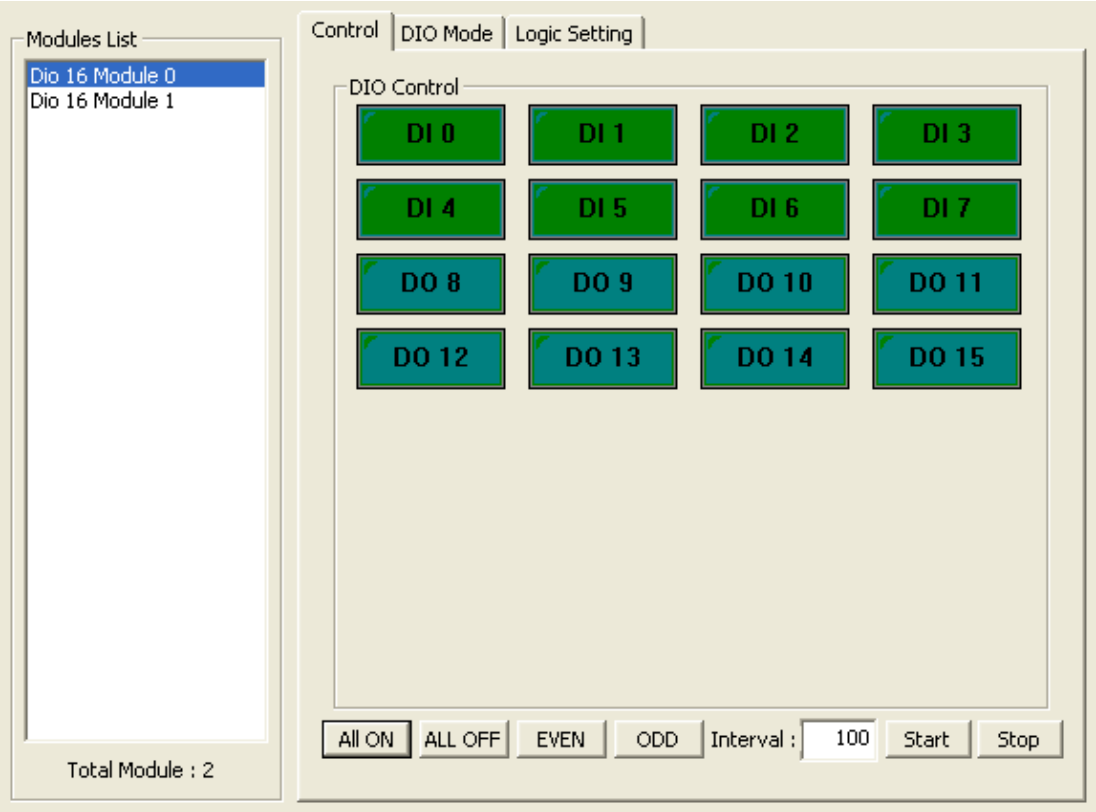
V.III MADIC 을 이용한 DIO 제어

DIO Tab 에서는 On/Off Control, Dio Mode Setting, Dio Logic Setting 의 조작이 가능합니다.

V.III.i Digital I/O : Control Panel

cEIP 모듈은 디지털 입·출력 채널의 방식을 자유롭게 설정할 수 있습니다.

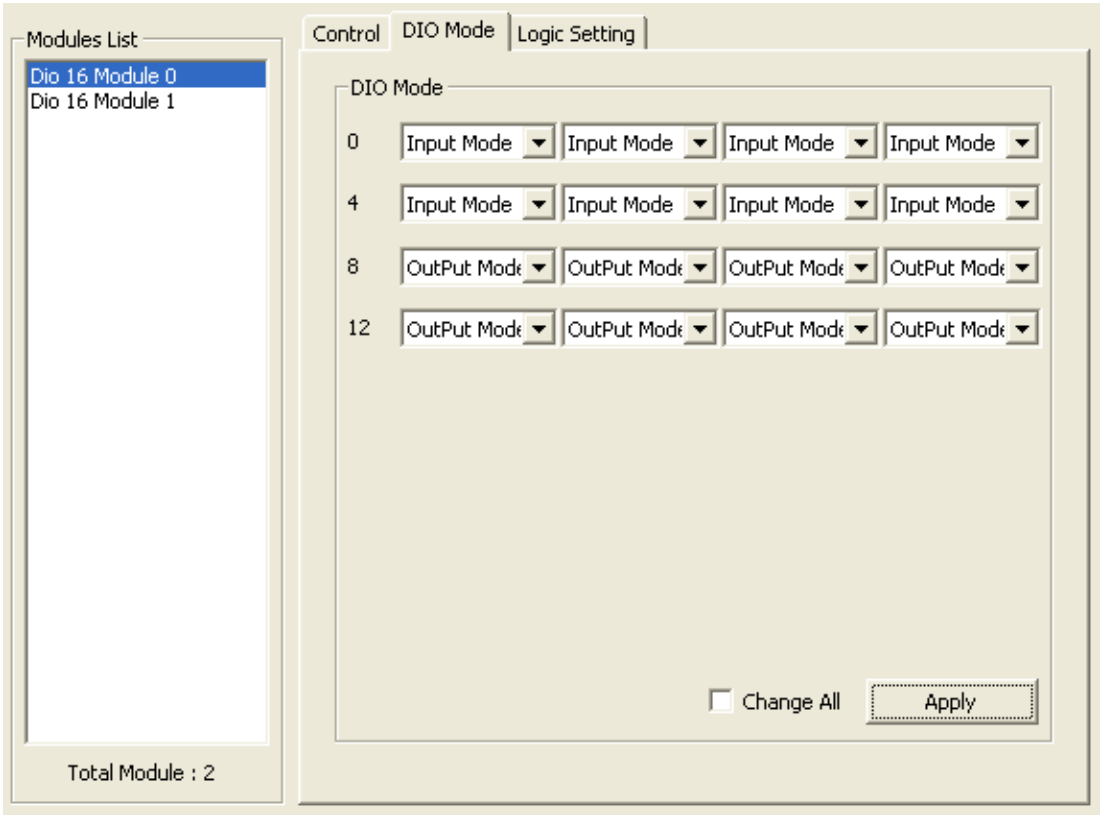
녹색은 Input Mode 를 나타내며, 파랑색은 Output Mode 를 나타냅니다. Output Mode 로 설정된 버튼은 마우스로 클릭 시 출력을 내보내게 됩니다.



- ALL ON : 모든 출력 채널을 ON 시킵니다.
- ALL OFF : 모든 출력 채널을 OFF 시킵니다.
- EVEN : 출력 채널의 짝수 행을 ON 시킵니다.
- ODD : 출력 채널의 홀수 행을 ON 시킵니다.
- Interval : 출력 채널의 Blink 모드 시 깜박이는 간격을 설정합니다. 이 값은 전체 DIO 모듈에 대해서 동일하게 적용됩니다.
- Start : Blink 모드를 시작합니다. 마지막으로 클릭한 버튼(All On, All Off, Even, Odd)에 따라서 Blink 모드를 동작 시킵니다.
- Stop : Blink 모드를 중지합니다.

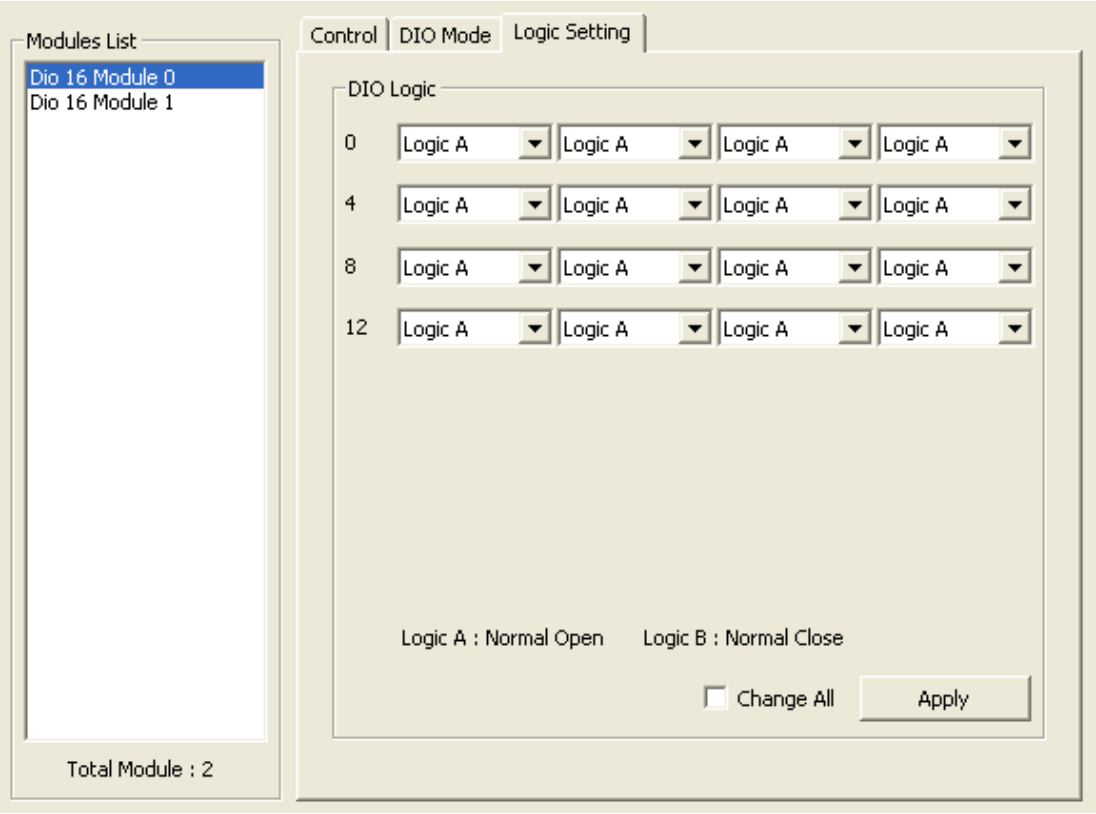
V.III.iiDigital I/O : DIO Mode Panel

DIO 모듈의 입력 및 출력 방식을 설정합니다. 왼쪽의 모듈 선택창에서 선택된 모듈의 채널들을 입력이나 출력으로 설정 할 수 있습니다. 변경할 채널을 선택하여 설정 하신 후 'Apply'버튼을 클릭하시면 설정이 적용됩니다. 변경된 상태는 DIO Control 탭에서 확인 할 수 있습니다.



V.III.iii Digital I/O : DIO Logic Setting Panel

DIO 모듈의 로직을 설정합니다. 왼쪽의 모듈 선택창에서 선택된 모듈의 채널들을 Logic A 나 Logic B 로 설정 할 수 있습니다. 변경할 채널을 선택하여 설정 하신 후 'Apply' 버튼을 클릭하시면 설정이 적용됩니다.

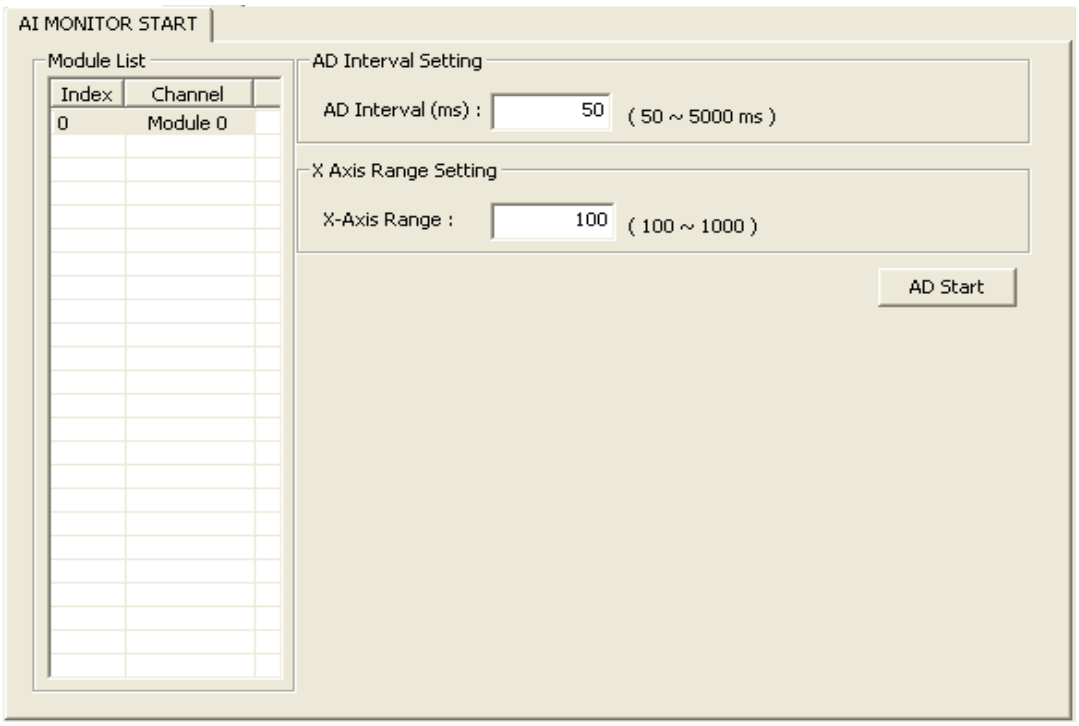


V.IV MADIC 을 이용한 AI (Analog Input) 제어

AI Tab 에서는 아날로그 입력 전압 범위를 설정 및 계측한 AD 값을 Digit, Volt, Current 값으로 반환합니다.

V.IV.iAnalog Input : AI Monitor Start

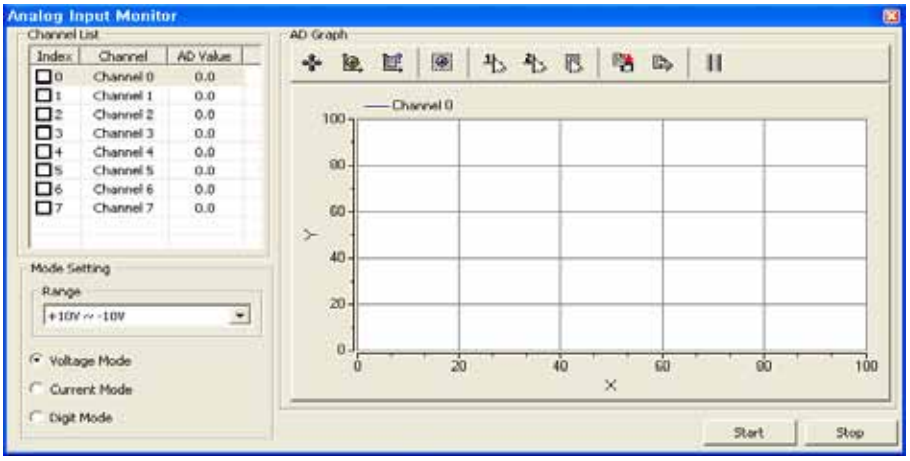
AI Monitor 다이얼로그를 시작하기 위한 기본 설정을 하는 페이지 입니다. 한번에 여러 개의 다이얼로그를 표시하여 다수의 모듈에 대한 AD 값을 감시하는 것이 가능합니다.



- Module List : AD Monitor 를 시작할 모듈을 선택합니다.
- AD Interval Setting : AD 할 시간 간격을 설정합니다. 최소 50ms 부터 최대 5000ms 까지 설정이 가능합니다.
- X Axis Range Setting : 그래프의 X 축에 표시할 아이템의 개수를 설정합니다. 최소 100 개에서 최대 1000 개의 아이템의 개수를 한축에 표시 할 수 있습니다.
- AD Start : AD 다이얼로그를 설정한 값으로 시작합니다.

V.IV.ii Analog Input : AI Monitoring

현재 Analog Input 값을 읽어서 그래프로 확인 할 수 있는 다이얼로그 입니다. 선택한 모듈에 속한 채널들 중에서 최대 4 개까지의 채널을 선택하여서 한번에 그래프에 표시하는 것이 가능합니다. 각 채널을 선택하는 리스트 컨트롤에 각 채널의 현재 AD 값을 표시하여 현재 Analog Input 값을 확인하실 수 있습니다.



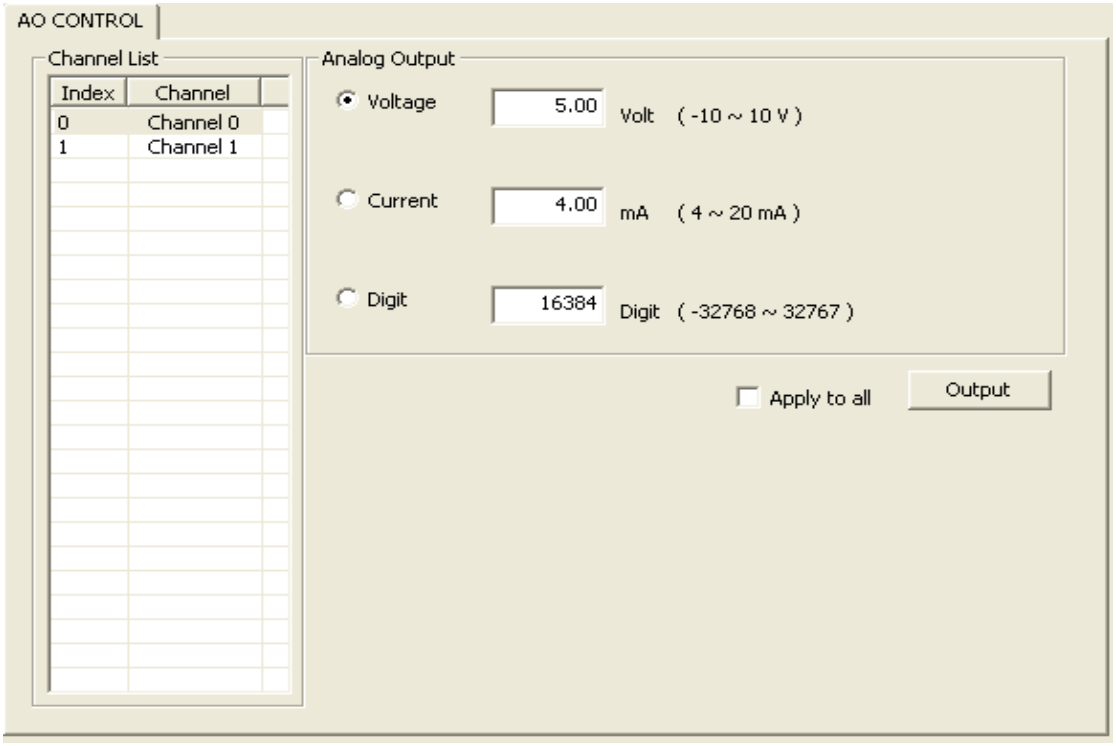
- Channel List : AD 값을 감시할 채널을 선택합니다. 채널은 한번에 4 개까지 선택 할 수 있으며, AD Value 는 선택한 채널에 대해서만 업데이트 됩니다.
- Mode Setting : AD Mode(전압 입력 범위 설정 모드)를 설정합니다. 선택된 AD 모드를 통해 AD 를 시작합니다.
- AD Graph : 현재 입력되고 있는 값을 그래프로 표시합니다. 그래프의 정확한 값은 Channel List 에 있는 AD Value 항목을 참조하시면 됩니다.
- Start : 설정값에 따라서 AD 를 시작하고, 현재 읽어들인 값을 그래프와 AD Value 에 표시합니다.
- Stop : AD 의 진행을 멈춥니다. 정지 후 다시 시작할 경우에는 X 축의 값이 0 으로 초기화 됩니다.

V.V MADIC 을 이용한 AO (Analog Output) 제어

AO Tab 에서는 아날로그 출력 채널을 통해 Digit, Volt, Current 값으로 출력 제어를 합니다.

V.V.i Analog Output : AO Control

Analog Output 을 제어할 수 있는 페이지입니다. 한번에 한 개의 채널에 대한 설정 및 출력이 가능하며, Apply All 버튼에 체크 할 경우 전체 축에 대해서 동일한 값을 설정하는 것이 가능합니다.



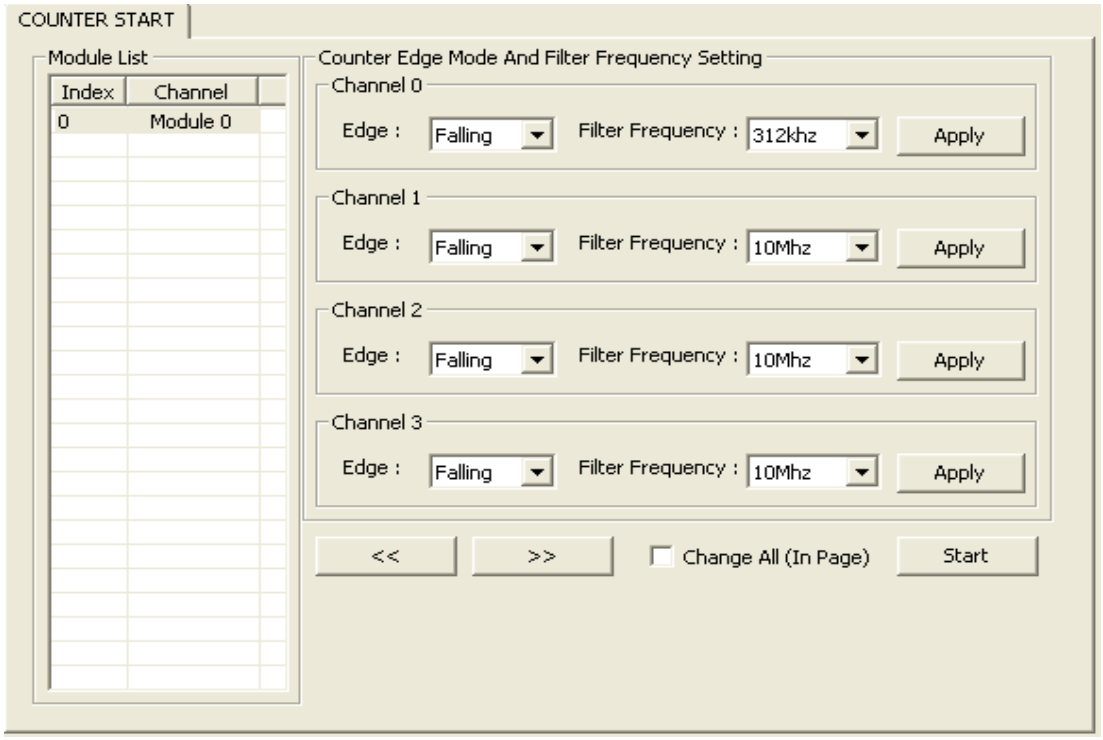
- Channel List : Analog 값을 출력할 채널을 선택합니다.
- Analog Output : Analog 출력 모드를 설정합니다.
- Apply to all : 전체 채널에 대해서 동일한값을 설정할 경우에 사용합니다. ‘Output’버튼을 클릭 할 때 값들이 적용됩니다.
- Output : 선택된 모드에 따라 설정된 값을 출력합니다.

V.VI MADIC 을 이용한 카운터(Counter) 제어

COUNTER Tab 에서는 Edge Mode, Filter Cutoff Frequency 설정 및 설정된 환경을 통해 펄스 신호를 카운트 합니다.

V.VI.iCounter : Counter Start

Counter 감시 다이얼로그의 기본 값을 설정하는 페이지 입니다. 페이지에서 설정한 값에 의하여 Counter 가 동작하게 됩니다. 아래의 '<<', '>>' 버튼을 사용하여서 설정할 채널들이 있는 페이지를 선택할 수 있습니다.



- Module List : Counter 값을 감시할 모듈을 선택합니다.
- Edge : Count 할 Edge 의 상태를 결정합니다.
- Filter Frequency : 대상 카운터의 채널별 필터 Cutoff 주파수를 설정합니다.
- << : 앞의 4 채널을 표시합니다.
- >> : 다음 4 채널을 표시합니다.
- Apply : 각 채널에 대해서 변경된 값을 적용시킵니다.
- Change All : 현재 표시되어 있는 페이지(4 개 채널)에 대해서 변경된 설정 값을 동일하게 적용합니다. 값을 변경시킨후 Apply 버튼을 4 개중 한 개를 클릭하시면 4 개 채널에 대해서 적용됩니다.
- Start : Counter Monitor 다이얼로그를 표시합니다.

V.VI.ii Counter : Counter Monitoring

Counter 의 현재 값을 표시합니다. Counter 는 Edge 모드에 따라서, Rising 이나 Falling 중 한가지 모드로만 사용하는 것이 가능합니다. 카운터의 값이 계속 증가하게 될경우 Overflow 가 발생 할 수 있으며, Overflow 가 발생하게 될 경우 'Overflow Clear'버튼이 활생화 됩니다. 활성화면 버튼을 클릭하시면, Overflow Flag 를 Clear 할 수 있습니다.

- Counter Value : 입력 받은 Rising Edge 혹은 Falling Edge 의 수를 표시합니다.
- Reset : Counter 의 값을 0 으로 초기화 합니다.
- Overflow Clear : Overflow 가 발생 했을 때만 활성화 되며, 클릭시 Overflow Flag 가 Clear 됩니다.

Appendix

C

Motion Default Parameter

어떤 응용 프로그램이나 장비에도 모든 설정에는 초기값을 보유하고 있습니다. 여기서는 보다 자세하고 정확한, 그리고 적절한 모션 환경설정을 위해 고객 여러분들께 제공하는 모션 초기화 매개변수를 소개합니다. 모션 환경 설정 이전에도 세심한 배려와 더불어 고객님께 안정적인 환경설정을 안내해 드리기 위해 준비하였습니다.

모션 시스템 초기화시에 설정되는 원격 노드 초기값(Default) 에 대해서 안내합니다. 여기서 말하는 장치 초기화는 ceSDK 의 함수를 통해 초기화된 상태 혹은 그 값(Value)을 의미하며, 고객 여러분들께서 해당 부분에 대해서 설정하지 않으셨거나, 초기화 설정 값을 알기 위한 용도로 다양하게 사용될 수 있습니다.

VI 모션 모듈 초기화 시의 초기(Default) 값

VI.I Command & Feedback

항목구분		기본(초기) 값		관련 함수
		값	의미	
Command	Output Mode	4	CW & CCW 1	cemCfgOutMode_Set
	Unit Distance	1	거리 단위를 펄스로 사용	cemCfgUnitDist_Set
	Unit Speed	1	속도 단위를 PPS 로 사용	cemCfgUnitSpeed_Set
Feedback	Input Mode	0	1X A/B Phase	cemCfgInMode_Set
	Inverse Direction	0	방향을 바꾸지 않음	cemCfgInMode_Set
In/Out Ratio		1	Feedback / Command Ratio	cemCfgInOutRatio_Set

VI.II INP, ALM , EL

항목구분		기본(초기) 값		관련 함수
		값	의미	
Inposition(INP)	Enable INP	0	INP 신호를 모션에 적용하지 않음	cemCfgMioProperty_Set (cemMPID_INP_EN)
	Input Logic	0	Normal Open(A 접점)	cemCfgMioProperty_Set (cemMPID_INP_LOGIC)
Alarm(ALM)	Input Logic	0	Normal Open(A 접점)	cemCfgMioProperty_Set (cemMPID_ALM_LOGIC)
	Stop Mode	0	즉시 정지	cemCfgMioProperty_Set (cemMPID_ALM_MODE)
External Limit (-EL, +EL)	Input Logic	0	Normal Open(A 접점)	cemCfgMioProperty_Set (cemMPID_EL_LOGIC)
	Stop Mode	0	즉시 정지	cemCfgMioProperty_Set (cemMPID_EL_MODE)

VI.III LTC, CMP, CLR, ERC

항목구분		기본(초기) 값		관련 함수
		값	의미	
Latch (LTC)	Input Logic	0	Normal Open(A 접점)	cemCfgMioProperty_Set (cemMPID_LTC_LOGIC)
	LTC2 Source	0	Deviation Counter	cemCfgMioProperty_Set (cemMPID_LTC_LTC2SRC)
Position Compare Output (CMP)	Signal Type	0	Normal Open(A 접점)	cemCfgMioProperty_Set (cemMPID_CMP_LOGIC)
	Pulse Width	0	Same width as command pulse	cemCfgMioProperty_Set (cemMPID_CMP_PWIDTH)
Clear Counter Input (CLR)	Input Signal Type	0	Falling edge	cemCfgMioProperty_Set (cemMPID_CLR_SIGTYPE)
	Target counters to clear	0	No counter selected (CLR 신호 사용안함)	cemCfgMioProperty_Set (cemMPID_CLR_CNTR)
ERC Output	Output Logic	0	Normal Open(A 접점)	cemCfgMioProperty_Set (cemMPID_ERC_LOGIC)

	On-time	6	104ms	cemCfgMioProperty_Set (cemMPID_ERC_OUT)

VI.IV DR, SD, STA, STP

항목구분		기본(초기) 값		관련 함수
		값	의미	
DR Input Signal Logic		0	Normal Open(A 접점)	cemCfgMioProperty_Set (cemMPID_DR_LOGIC)
Slow down (SD) Input	Enable SD	0	사용안함	cemCfgMioProperty_Set (cemMPID_SD_EN)
	Input logic	0	Normal Open(A 접점)	cemCfgMioProperty_Set (cemMPID_SD_LOGIC)
	SD Mode	0	SD 가 입력되면 초기속도까지 감속 후 초기속도로 이송	cemCfgMioProperty_Set (cemMPID_SD_MODE)
	SD Latch	0	SD 신호를 래치 하지 않는다.	cemCfgMioProperty_Set (cemMPID_SD_LATCH)
Start (STA) Input	Mode	0	Software 적으로 이송 명령이 내려지면 즉시 이송 시작(Hardware STA 신호 사용안함)	cemCfgMioProperty_Set (cemMPID_STA_MODE)
	Signal Type	0	Level(low active) Input	cemCfgMioProperty_Set (cemMPID_STA_TRG)
Stop (STP) Input Mode		0	Hardware 적인 STP 신호 사용안함	cemCfgMioProperty_Set (cemMPID_STP_MODE)

VI.V Software Limit

항목구분		기본(초기) 값		관련 함수
		값	의미	
Software Limit	Enable	0	Software Limit 해제	cemCfgSoftLimit_Set
	N-Limit value	-99999999	음의 방향의 최대값	
	P-Limit value	99999999	양의 방향의 최대값	

VI.VI Servo ON Input Logic

항목구분	기본(초기) 값		관련 함수
	값	의미	
Servo-ON Input Logic	0	Normal open (A 접점)	cemCfgMioProperty_Set (cemMPID_SVON_LOGIC)

VI.VII 원점복귀 환경

항목구분	기본(초기) 값		관련 함수
	값	의미	
Home Mode	0	원점복귀 모드 0 번	cemHomeConfig_Set
ORG Input Logic	0	Normal Open(A 접점)	

Ez Input Logic	0	Normal Open(A 접점)	
Ez Count	0	원점 복귀 시에 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	cemCfgMioProperty_Set (cemMPID_ERC_OUT)
Speed Mode	2	S-CURVE 모드	cemHomeSpeedPattern_Set
Work Speed	5000	원점복귀 정속도	
Acceleration	100000	원점복귀 가속도	
Deceleration	100000	원점복귀 감속도	
Reverse Speed	1000	원점복귀 역방향 이송 속도	

VI.VIII 모션 정격 속도 환경

항목구분	기본(초기) 값		관련 함수
	값	의미	
Max Speed	655350	655350 (PPS)	cemCfgSpeedRange_Set
Initial Speed	0	초기 속도	cemSxOptIniSpeed_Set
Speed Mode	2	S-CURVE 모드	cemCfgSpeedPattern_Set
Work Speed	10000	정속도	
Acceleration	100000	가속도	
Deceleration	100000	감속도	

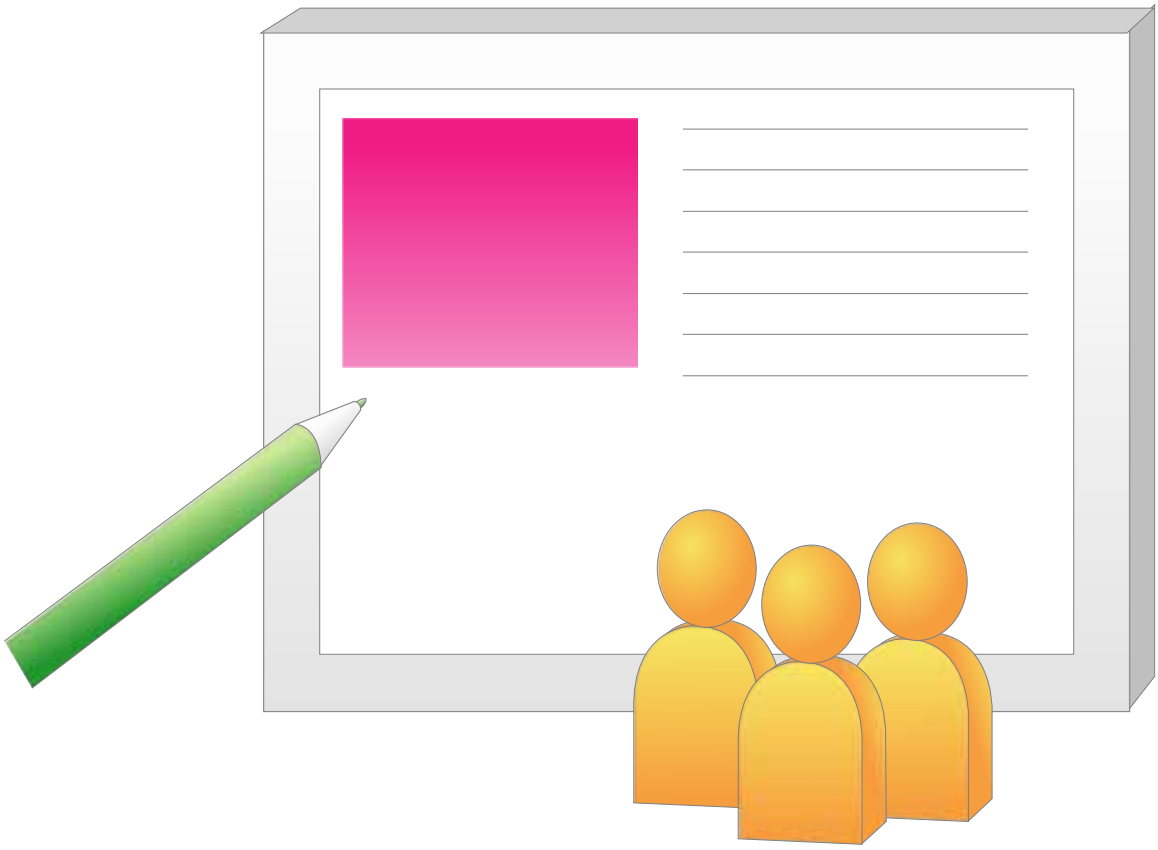
Appendix

D

List of Error Codes

에러코드는 모션 응용 프로그램뿐만 아니라, 모든 응용 프로그램에서 필요한 주요 정보 중 하나입니다. 커미조아의 ceSDK 에서는 자세하고, 일괄적인 에러코드를 제공함으로써, 에러 상태에 대한 명확한 원인과 분석이 가능할 수 있도록 돕고 있습니다. 본 장에서 열거한 에러코드는 다양한 에러 코드 상황에서, ceSDK 를 통해 모션 에러 상태를 효율적으로 자세하고 명확히 판단할 수 있는 근거를 제공합니다.

통합 라이브러리에서 제공하는 에러코드 리스트를 일람표로 정리하여 수록하였으며, 명시된 에러코드를 통해 효과적으로 에러처리를 하는 방법에 대해서 안내하고 있습니다. 일반적으로 ceSDK 사용하다가 에러가 발생하면 에러코드에 대한 의미를 에러코드 일람표를 참조하여 파악하시기 바랍니다.



I 에러 코드

I.I 에러 코드 일람표

ERROR CODE	VALUE	MEANING
ceERR_NONE	0	No error
	1	모션 함수의 수행에 오류가 발생하지 않았습니다.
	2	DIO 함수의 수행에 오류가 발생하지 않았습니다.
	3	AD/DA 함수의 수행에 오류가 발생하지 않았습니다.
	4	범용 함수의 수행에 오류가 발생하지 않았습니다.
	-1	모션 함수 수행에 알 수 없는 오류가 발생하였습니다.
	-2	DIO 함수의 수행에 알 수 없는 오류가 발생하였습니다.
	-3	AD/DA 함수의 수행에 알 수 없는 오류가 발생하였습니다.
	-4	범용 함수 수행에 알 수 없는 오류가 발생하였습니다.
ceGnERR_TIMEOUT	-101	communication timeout error.
ceGnERR_INVALID_PACKET	-102	Packet data error.
ceGnERR_CHECKSUM	-103	checksum mismatch.
ceGnERR_FLASH_ERASE_FAIL	-104	fail to erase flash-memory.
ceGnERR_UNDEFINE_COMMAND	-105	Undefined control command has been received.
cemERR_MEM_ALLOC_FAIL	-290	Memory allocation fail.
cemERR_GLOBAL_MEM_FAIL	-292	Global memory allocation fail.
cemERR_ISR_CONNEC_FAIL	-310	ISR(Interrupt Service Routine) registration fail.
cemERR_DIVIDE_BY_ZERO	-400	Cause divide by zero error.
cemERR_WORNG_NUM_DATA	-500	Number of data is too small or too big.
cemERR_VER_MISMATCH	-600	Version(of file or device) mismatch.
cmERR_FLASH_ERASE_FAIL	-601	fail to erase a flash-memory sector.
cmERR_FLASH_WRITE_FAIL	-602	fail to write a flash-memory sector.
cmERR_FLASH_COPYSECT_FAIL	-603	fail to copy whole data from source sector to target sector.
cmERR_FLASH_CANNOTMODIFY	-604	cannot modify the sector.
cmERR_INVALID_SYSCFG	-605	TSysConfig 의 dwSign 값이 SYS_CFG_SIGN 과 맞지 않음.
cmERR_MAXBYTESLIMIT_SYSCFG	-606	TSysConfig 의 dwComment 의 최대 허용바이트수 초과됨.
cemERR_INVALID_DEVICE_ID	-1010	User set invalid device id. Refer to “DeviceId” property.
cemERR_INVALID_HANDLE	-1020	Device handle is not valid. This means that loading a device has been failed or not performed. Refer to “ceGnLoad” function.
cemERR_UNSUPORTED_FUNC	-1030	User called an unsupported function for the specified product.
cemERR_INVALID_PARAMETER	-1101	Some of the function parameters are invalid.
cemERR_INVALID_CHANNEL	-1105	The channel setting parameter(s) is(are) invalid.
cemERR_INVALID_INPUT_RANGE	-1111	Invalid range value (AI, AO).
cemERR_INVALID_FREQ_RANGE	-1121	User selected invalid frequency range.
cemERR_FILE_CREATE_FAIL	-1501	File creation has been failed.
cemERR_FILE_OPEN_FAIL	-1511	File openning has been failed.
cemERR_FILE_READ_FAIL	-1522	File reading fail.
cemERR_EVENT_CREATE_FAIL	-1550	Event handle creation has been failed.
cemERR_INT_INSTANCE_FAIL	-1560	Interrupt event instance creation has been failed.
cemERR_DITHREAD_CRE	-1570	D/I state change monitor thread creation fail.
cemERR_BUFFER_SMALL	-1580	Buffer size is too small.
cemERR_HIGH_TIMER_UNSUPP	-1590	The installed hardware doesn’t support a high-resolution performance counter(when cemUtlDelayMicroSec() had failed).
cemERR_OUT_OF_RANGE	-1600	The range of some parameter is out of range that it is occurred.

ERROR CODE	VALUE	MEANING
cemERR_WRONG_MODE	-1601	Wrong range mode.
cemERR_ON_MOTION	-5001	This code is just a symbolic code. This error will never occur.
cemERR_STOP_BY_SLP	-5002	Abnormally stopped by positive soft limit.
cemERR_STOP_BY_SLN	-5003	Abnormally stopped by negative soft limit.
cemERR_STOP_BY_CMP3	-5004	Abnormally stopped by comparator3.
cemERR_STOP_BY_CMP4	-5005	Abnormally stopped by comparator4.
cemERR_STOP_BY_CMP5	-5006	Abnormally stopped by comparator5.
cemERR_STOP_BY_ELP	-5007	Abnormally stopped by (+) external limit.
cemERR_STOP_BY_ELN	-5008	Abnormally stopped by (-) external limit.
cemERR_STOP_BY_ALM	-5009	Abnormally stopped by alarm input signal.
cemERR_STOP_BY_CSTP	-5010	Abnormally stopped by CSTP input signal.
cemERR_STOP_BY_CEMG	-5011	Abnormally stopped by CEMG input signal.
cemERR_STOP_BY_SD	-5012	Abnormally stopped by SD input signal.
cemERR_STOP_BY_DERROR	-5013	Abnormally stopped by operation data error.
cemERR_STOP_BY_IP	-5014	Abnormally stopped by other axis error during interpolation.
cemERR_STOP_BY_PO	-5015	An overflow occurred in the PA/PB input buffer.
cemERR_STOP_BY_AO	-5016	Out of range position counter during interpolation.
cemERR_STOP_BY_EE	-5017	An EA/EB input error occurred (does not stop).
cemERR_STOP_BY_PE	-5018	An PA/PB input error occurred (does not stop).
cemERR_STOP_BY_SLVERR	-5019	Abnormally stopped because slave axis has been stopped (Only in Master/Slave mode).
cemERR_STOP_BY_SEMG	-5020	Abnormally stopped by "software emergency" setting.
cemERR_MOT_MAOMODE	-5110	Master output mode is not CW/CCW mode during Master/Slave operation.
cemERR_MOT_SLAVE_SET	-5120	Slave start has been failed during Master/Slave operation.
cemERR_SPEED_RANGE_OVER	-5130	Speed setting value exceeds setting range.
cemERR_INVALID_SPEED_SET	-5140	Speed setting value is not valid.
cmERR_ACC_LOW_LIMIT_OVER	-5142	Acceleration setting value is too low.
cmERR_ACC_HIGH_LIMIT_OVER	-5143	Acceleration setting value is too high.
cmERR_DEC_LOW_LIMIT_OVER	-5144	Deceleration setting value is too low.
cmERR_DEC_HIGH_LIMIT_OVER	-5145	Deceleration setting value is too high.
cemERR_INVALID_IXMAP	-5150	Invalid Interpolation Map.
cemERR_INVALID_LMMAP	-5160	Invalid List-Motion Map.
cemERR_MOT_SEQ_SKIPPED	-5170	Motion command has been skipped because the axis is already running.
cmERR_MOT_PREREG_FULL	-5180	Motion command is skipped because the pre-register is full of an axis.
cmERR_LTC_QUE_SIZE_ERROR	-5190	Specified latch-que size is too big.
cmERR_LTC_QUE_BUF_FREED	-5191	Latch-que is not allocated.
cmERR_UNKNOWN	-9999	Unknown Error.
	-11000	컨트롤 메시지 전송이 실패하였습니다.
	-20000	데몬 서비스 프로세스와 통신이 중단 되었습니다. 이 오류는 ceSDKDaemon 이 수행중이지 않거나 , 서비스 프로세스가 동작중이지 않을 때 나타납니다.
	-20001	데몬 서비스 프로세스와의 통신을 위한 SOCKET 개방에 실패하였습니다.
	-20002	데몬 서비스 프로세스와의 통신 시간이 초과하였습니다. 지정된 시간 동안 데몬 서비스 프로세스는 응답이 없었습니다.
	-20003	해당 함수를 구동하려고 하였으나, ceGnLoad 함수가 호출되지 않았습니다.
	-20004	데몬 서비스 프로세스와의 통신 중 수신이 실패하였습니다. 수신된 데이터의 길이는 0 이거나 그 이하 입니다.

ERROR CODE	VALUE	MEANING
	-20005	공유메모리를 개방하는 작업을 실패하였습니다.
	-20006	데몬 프로세스가 실행 되지 않았습니다. 데몬 프로세스 자체가 구동중이지 않을 수 있습니다.
	-20007	데몬 프로세스의 활성화가 되지 않았습니다. 데몬 프로세스 자체가 구동중이나, 준비 작업이 완료되지 않은 경우에 발생합니다. 이 상태가 지속되면, 폐사로 문의하여 주시기 바랍니다.
	-20008	올바르지 않은 원격 노드 ID 입니다.
	-20009	올바르지 않은 보간 맵 번호입니다.
	-50000	라이브러리 로드가 이미 완료되어 있어, 더 이상 로드 할 필요가 없습니다.
	-50001	디버그 모드를 설정하는 동안 오류가 발생하였습니다.
	-50002	라이브러리 로드가 되지 않아서, 동작할 수 없습니다.
	-50003	모션, DIO, AD/DA 등의 채널 개수를 얻어 오거나 관련 처리부에서 채널을 찾을 수 없거나 실패하였습니다.
	-50004	입력한 노드의 범위가 초과하였습니다. 노드 번호는 0 부터 255 이하로 사용됩니다.
	-51000	입력한 노드 번호가 유효하지 않습니다. -50004 번 에러를 참조하십시오.
	-51001	해당 노드는 노드 정보 획득 시에 등록되지 않은 노드입니다.
	-51002	올바르지 않은 전역 채널 번호입니다.
	-51003	반환되는 채널은 올바른 채널 번호가 아닙니다.
	-51004	전역 설정이 올바르지 않습니다.
	-61000	해당 모션 축은 구동 중이지 않습니다. 구동 중이지 않은 축에 대해서 SxStop 이나 SxStopEmg 를 수행하였습니다.
	-70000	다축 이송 함수의 매개변수가 올바르지 않습니다.
	-81000	보간 제어를 수행하려고 하는 맵은 이미 등록되어 있습니다.
	-81001	보간 제어를 수행하려고 하는 맵은 등록되지 않은 맵이거나, 등록되지 않은 맵을 제거하려고 하였습니다.
	-81002	보간 제어를 수행하려고 맵을 구성하였습니다만, 해당 노드에서 벗어나는 모션 축수를 선택하였습니다.
	-100000	Default 다운로드를 대상으로 하는 디폴트 설정값을 요구하는 노드가 0 개 입니다.
	-100001	업로드를 대상으로 하는 노드가 0 개입니다.
	-100002	다운로드를 하려고 하는 대상 파일을 읽을 수 없습니다.
	-100003	다운로드를 하려고 하는 대상 파일의 내부 노드 개수가 0 이거나, 그 이하입니다.
	-100004	다운로드를 하려고 하는 대상 파일을 통해 다운로드를 하였으나, 실제 다운로드된 노드가 0 이거나, 그 이하입니다.
	-500000	현재 기능이 지원되지 않는 함수입니다.

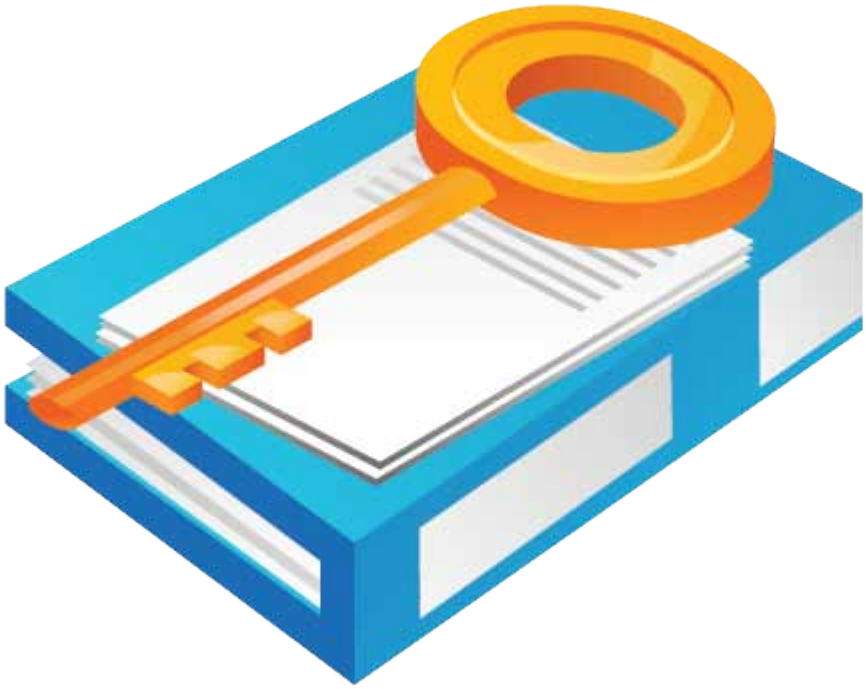
Appendix

E

Index of ceSDK Functions

필요한 함수를 가장 빠르고 쉽게 찾으십시오. ceSDK 매뉴얼에서는 고객님들께서 원하시는 함수들을 일목 요연하게 정리하였습니다. 필요한 함수는 하이퍼링크 기능으로 찾을 수 있도록 구성하였습니다.

하이퍼링크 기능을 통해 본 장에서는 빠르고 정확하게 고객님들께서 원하시는 함수를 찾으실 수 있도록 구성하였습니다. Adobe Acrobat Reader 와 같은 전자 문서 뷰어(Viewer) 를 통해 최단 시간 내에 원하시는 함수를 찾을 수 있습니다.



I Index of ceSDK Functions

I.I Quick Reference to ceSDK Functions

General Functions

TEST & MEASUREMENT & AUTOMATION / COMIZOA

ceSDK Manual

2009 년 01 월 08 일 1 판 제작
매뉴얼 자료 번호 : 1.0.5

㈜커미조아
http://www.comizoa.com
Tel) +82 - 42 - 936 - 6500~6
Fax) +82 - 42 - 936 - 6507