COMIZOA
Data Acquisition Software Development Kit

TEST & MEASUREMENT & AUTOMATION

COMIZOA DATA ACQUISITION SYSTEM

JULY 2014
P/N 0708-2014-01

API Reference Manual

TEST & MEASUREMENT & AUTOMATION / COMIZOA

DX-SDK Manual

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

Technical Support

E-mail : csteam@comizoa.com
File Server : ftp.comizoa.com
Website : http://www.comizoa.com

Address
691 Gwanpyeong-dong, Yuseong-gu, Daejeon, 305-509 Korea

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

COMIZOA DAQ System Integrated Control Library Reference

Table of Contents

Chapter

1

Introduction

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

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

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

1 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 지원을 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, 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 등록 상표입니다.

Visual C++ 는 Microsoft Corp. 의 등록 상표입니다.

Visual Basic 은 Microsoft Corp. 의 등록 상표입니다.

C++ Builder 는 Borland Software Corp. 의 등록 상표입니다.

Delphi 는 Borland Software Corp. 의 등록 상표입니다.

이외의 상표는 각 회사의 등록상표입니다.

1.1.5 주의사항

㈜커미조아의 제품 군에는 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 제품 군은 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 DX-SDK(COMI-DAQ DX Series SDK Library)에 대한 정보를 포함한, 실제 라이브러리를 통한 DAQ 제품군의 제어 사항에 대한 기반 설명을 포함하고 있습니다.

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

1.1.6 매뉴얼 아이콘 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 1] 매뉴얼 아이콘의 설명

1.2 Features

㈜ 커미조아의 DAQ 제어 라이브러리인 DX-SDK 의 장점은 다음과 같습니다.

1.2.1 독립성

DX-SDK 는 Microsoft 사의 표준 라이브러리인 DLL(Dynamic Link Library) 형태의 독립된 라이브러리 인터페이스를 제공하며, 라이브러리 자체의 독립된 장치 관리 및 "COMIZOA DAQ Environment"를 이용한 장치 관리의 편의성을 제공하고 있으며, DLL 형태의 라이브러리 장접을 통해 유지 보수와 귀사의 제품 구현에 보다 간편하게 하고 신뢰성 있는 독립형 동적연결 라이브러리를 제공합니다.

1.2.2 호환성

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

고객님께서는 언제나 DX-SDK 를 통하여 귀사의 제품에 보다 신속하고 정확한 시스템을 구현하실 수 있습니다.

1.2.3 편의성

인터페이스 함수 명명 규칙의 통일화와 의사코드 주제어를 매뉴얼과 라이브러리 매개변수(Parameter)에 부각시켜, 보다 빠른 시간 내에 숙련된 라이브러리 사용자로 만들어 드립니다. 특히, DX-SDK 의 모든 함수 명에는 의미적 명명 규칙을 내포하였으며, 이것은 분명 실무 개발 환경에서 많은 부분 이점으로 작용 할 것입니다.

1.2.4 확장성

DX-SDK 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 DX-SDK 를 통해 제품 개발을 더욱 향상시키실 수 있습니다.

1.2.5 신뢰성

㈜커미조아는 PCI 와 CompactPCI 버스 기반의 고성능 DAQ 장치를 개발하여 제공하고 있으며, 제품군들에 대한 라이브러리는 오랜 시간 산업 현장에서의 현장 경험을 바탕으로 형성된 신뢰성 있는 제품군 입니다. 또한 보다 최적화된 DAQ 시스템 구성을 지원하기 위하여 다양한 제품군을 구성하고 있으며, 각 동작에 대한 입력과 출력을 DX-SDK 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 지원 해 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 DX-SDK 로 이제 보다 향상된 고객 지원과 기술 지원을 통해 고객 여러분들의 성원에 이바지하겠습니다.

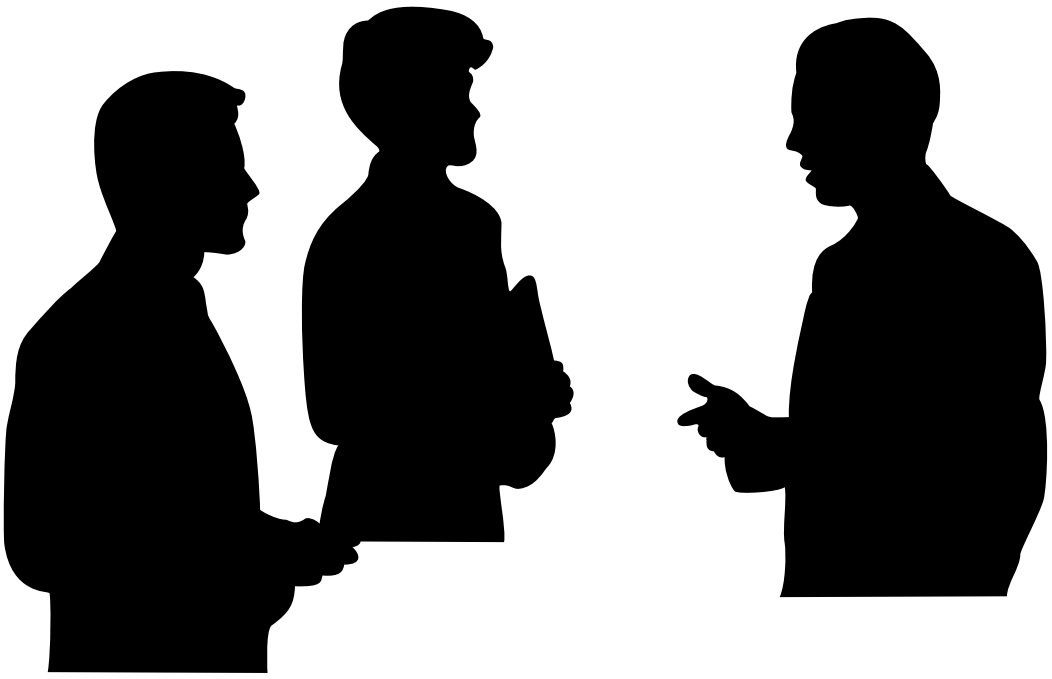
Chapter

2

Development Environment for DX-SDK

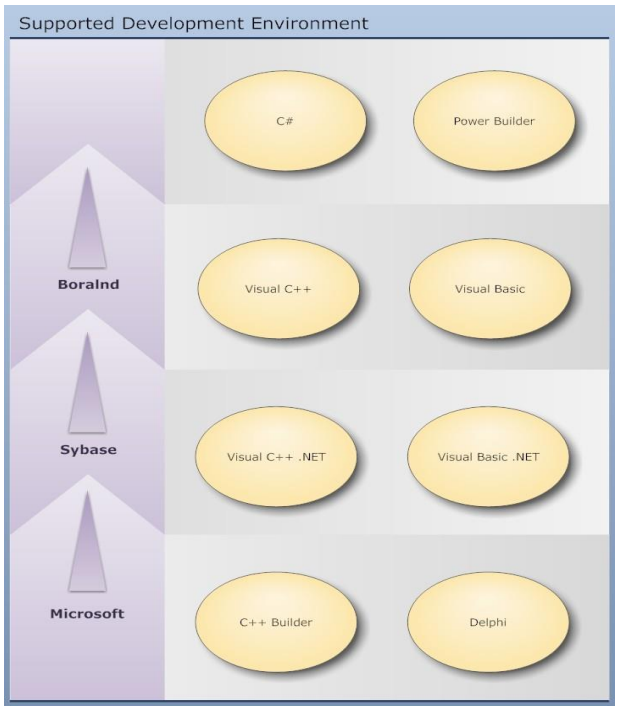
㈜커미조아는 DX-SDK 라이브러리를 통해 다양한 최신 개발환경을 지원하기 위해 노력하고 있습니다. 본 장에서 다루지 않는 개발환경을 이용하시는 고객님께서는 저희 ㈜커미조아를 통해 문의하여 주시면 신속히 대처해 드리도록 하겠으며, 제공되는 DX-SDK 인터페이스를 통해 보다 편리하고 빠르게 라이브러리를 사용할 수 있도록 지원하여 드립니다.

커미조아 DAQ 제품 군의 DX-SDK 라이브러리는 다양한 고객 여러분들의 요구에 발맞추어 사용의 편리성 과 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 DX-SDK 사용 안내

2.1 개발 환경 지원 안내



Development Environment	Programming Language	Version	Product
Microsoft Visual Studio	Visual C++	VS2003,VS2005 including 6.0 for lower version	All products including Enterprise Edition
	Visual Basic		
	C# (Sharp)		
Borland International Borland Development Studio	C++ Builder	All version	Architect, Professional, Enterprise
	Delphi	All version	
	C# (Sharp)	BDS 2006 version	
Borland International Borland Turbo Series	C++ Builder	Turbo C++	All products
	Delphi	Turbo Delphi / Turbo Delphi for .NET	All products
	C# (Sharp)	Turbo C#	All products
Sybase PowerBuilder	PowerBuilder	PowerBuilder 8, PowerBuilder 9, PowerBuilder 10.5	All products

[표 2] DX-SDK 지원 개발 환경

DX-SDK 는 위와 같은 개발 환경을 지원하며, 별도로 명시되지 않은 개발 환경에서도 윈도우의 Dynamic Link Library 형태를 사용 가능한 경우 DX-SDK 를 사용하실 수 있습니다.

2.2 COMI-DAQ DX Series SDK (DX-SDK) 소개

DX-SDK(COMI-DAQ DX Series SDK Library)는 ㈜커미조아의 DAQ(Data Acquisition) 제품을 사용하여 사용자 프로그램을 개발할 때 ㈜커미조아의 DAQ 제품이 제공하는 모든 기능을 활용할 수 있도록 소프트웨어 인터페이스를 제공하는 라이브러리 소프트웨어입니다.

DX-SDK 는 Microsoft 사의 표준 라이브러리인 DLL(Dynamic Link Library) 형태의 독립된 라이브러리 인터페이스를 제공되며 파일명은 “ComiDXDll.dll” 입니다. DLL 이 지원되는 모든 컴파일러에서 자유롭게 사용할 수 있습니다.

DX-SDK 는 DAQ 제품의 기능 활용을 극대화할 수 있도록 강력한 기능을 제공함과 동시에 개발자들이 보다 편리하고 빠르게 프로그램을 개발할 수 있도록 조직화된 인터페이스를 제공하며, 프로그램의 오류를 철저히 분석할 수 있는 개발환경까지 제공하는 최적화된 라이브러리입니다.

2.3 컴파일러 별 DX-SDK 사용 방법

개발 환경 / 항목	MS VC++	Borland C++ Builder	Borland Delphi	MS Visual Basic	MS C# (C Sharp)
DX-SDK 인터페이스 파일 명	ComiDXSDK.h ComiDXSDK.cpp	ComiDXSDK.h ComiDXSDK.cpp	ComiDXSDK.PAS	ComiDXSDK.BAS	ComiDXSDK.CS
DX-SDK 상수(常數)정의 파일	DXDllDefine.h	DXDllDefine.h			

표 3 DX-SDK 인터페이스 구성 파일 안내

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

㈜ 커미조아 고객(顧客)님께서는 "DX-SDK 인터페이스 파일" 을 통해 DX-SDK 의 함수를 명시적으로 응용프로그램에 포함시킬 수 있으며, 이에 따라 상기 표기한 개발 환경을 지원해드리고 있습니다.

저희 ㈜ 커미조아에서는 고객(顧客)님들의 개발환경에 대한 고민을 덜어드리기 위하여, 범용적인 객체지향 언어인 C++ 부터 Borland International 社의 Object Pascal 의 Delphi 제품군, 그리고 **최신 .NET 환경까지 고려한 라이브러리 인터페이스를 제공**하고 있습니다. 특히 이번 DX-SDK 에서는 개발 환경의 새 패러다임인 C Sharp(C#) 언어를 본격 지원하고 있으며, 보다 새로운 개발환경에서 저희 ㈜ 커미조아 제품 군을 경험하실 수 있도록 만전을 기하고 있습니다.

필요한 인터페이스(Interface) 파일은 실제 개발언어 및 환경에서 Header 파일 또는 프로젝트 구성파일로 반드시 사용이 됩니다. 저희 ㈜ 커미조아 DX-SDK 에서 제공하는 인터페이스파일의 경로는 다음과 같습니다.

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

표 4 이 경로는 ㈜ 커미조아에서 제공하는 COMI-AUTOMATION 설치 프로그램이 기본 경로인'C:' 드라이브에 설치된 것을 기준으로 합니다.

인터페이스 파일에 대해서 자세히 설명해 주십시오.

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

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

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

2.3.1 Visual C++ 6.x 개발자를 위한 안내

Microsoft Visual C++ 6.x 에서 DX-SDK 를 사용하시려면 다음의 절차에 따라 사용하시면 됩니다.

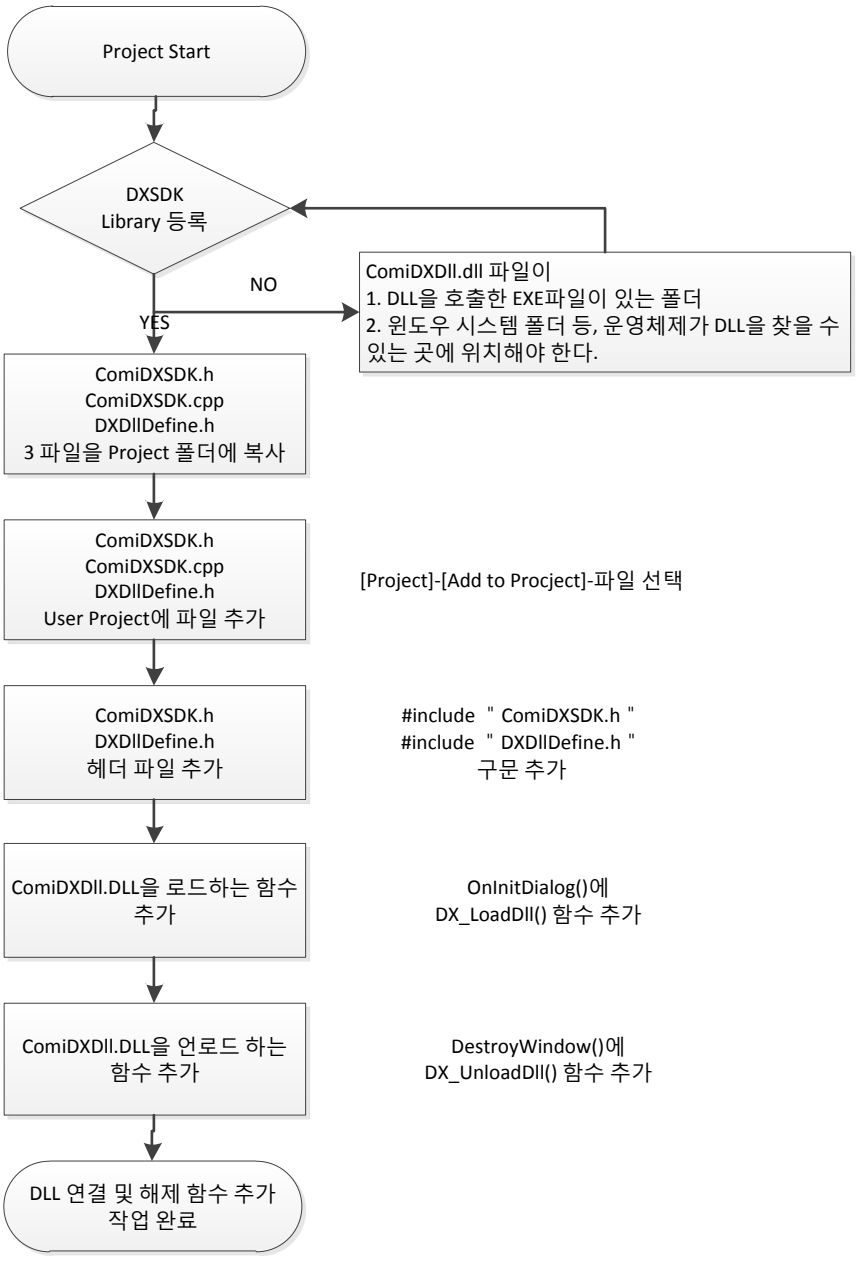


그림 1-1 Visual Studio 6.x 에서의 DX-SDK 사용 순서도

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

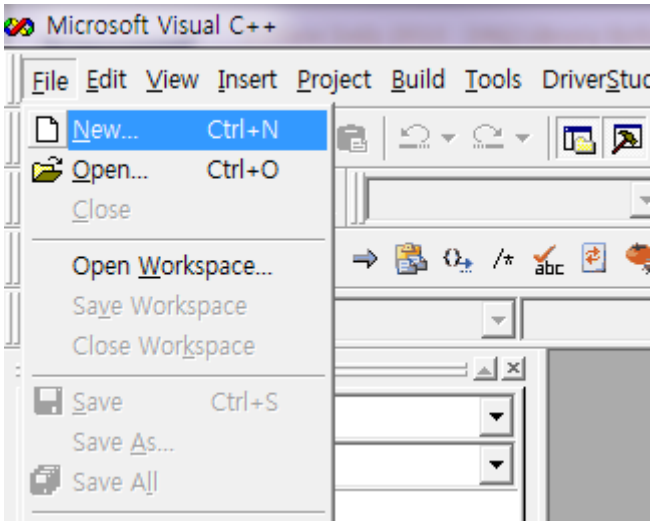


그림 1-2 Visual C++ 6.x 의 새로운 프로젝트 생성 화면

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

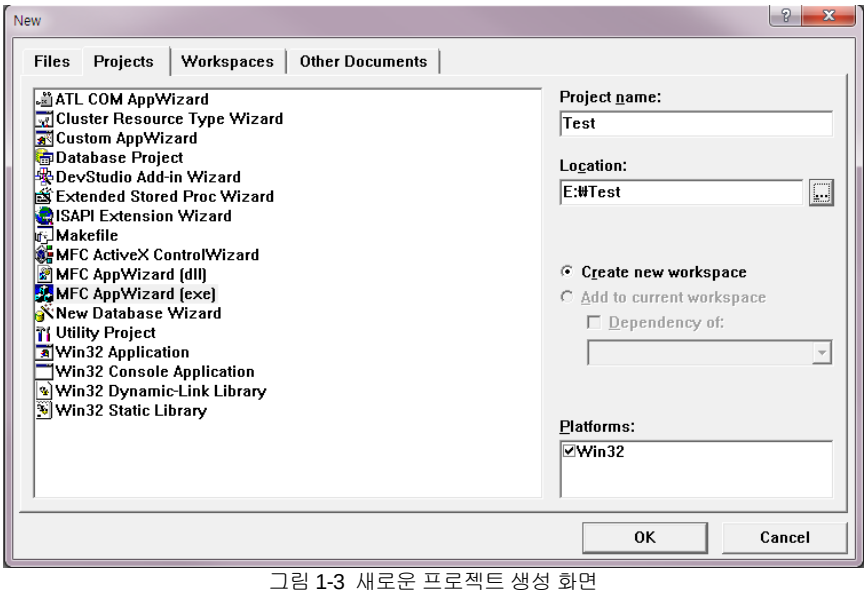


그림 1-3 새로운 프로젝트 생성 화면

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

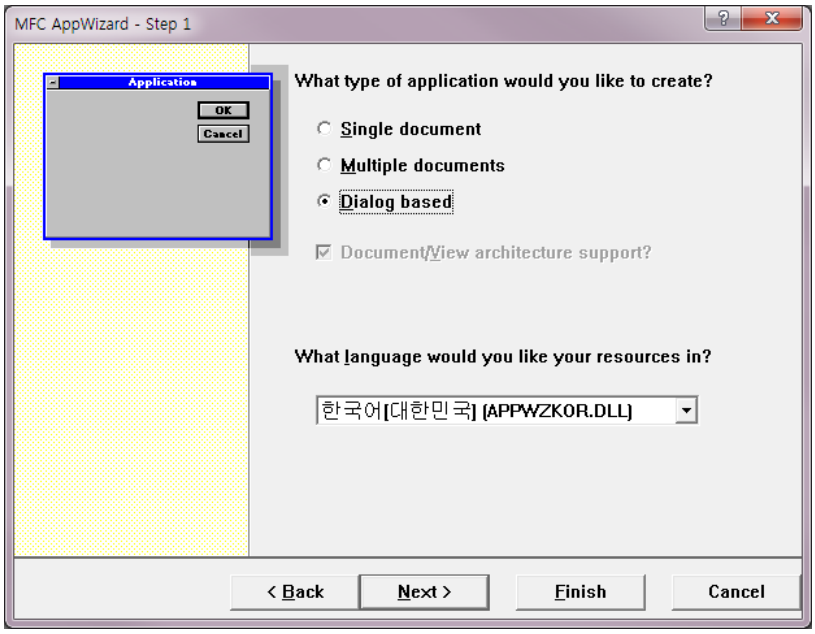


그림 1-4 MFC AppWizard 의 Application Type 선택 화면

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

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

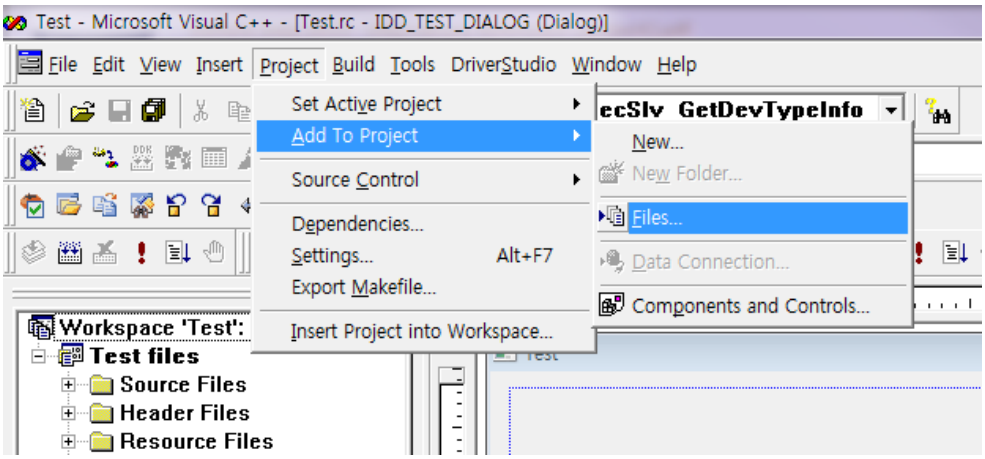


그림 1-5 프로젝트에 새로운 파일 추가 선택화면

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

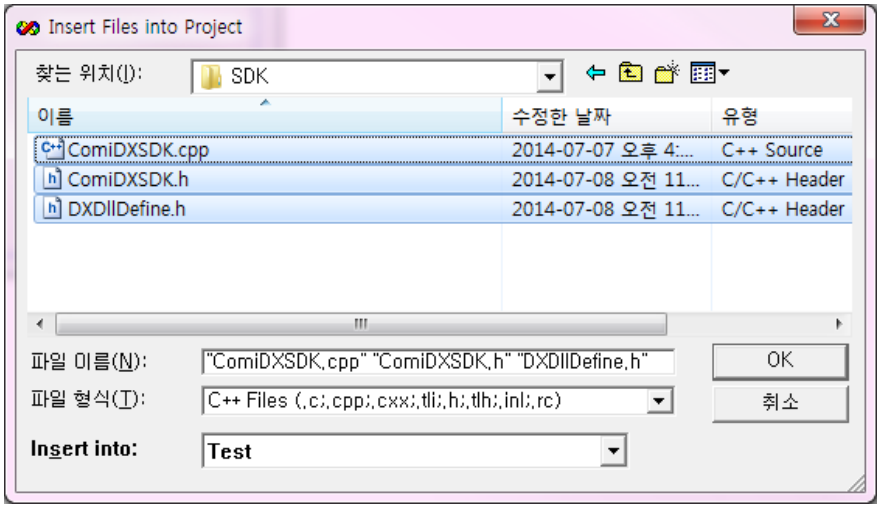


그림 1-6 프로젝트에 추가할 파일 선택 화면

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

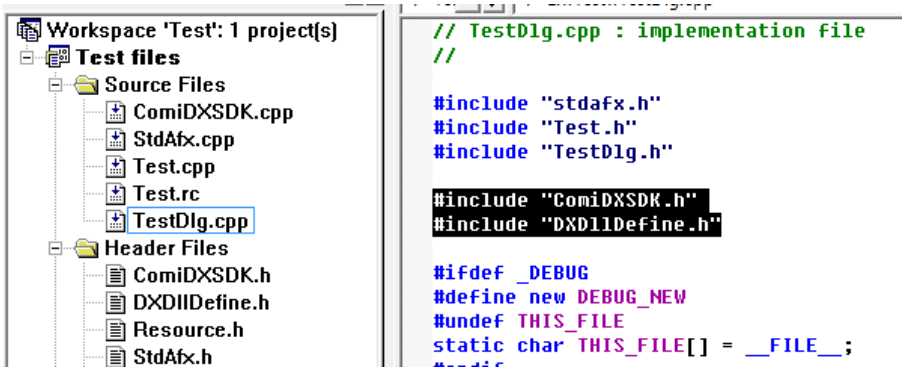


그림 1-7 사용자가 MFC AppWizard 를 통해서 생성한 소스코드에 ComiDXSDK 파일을 추가 함

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

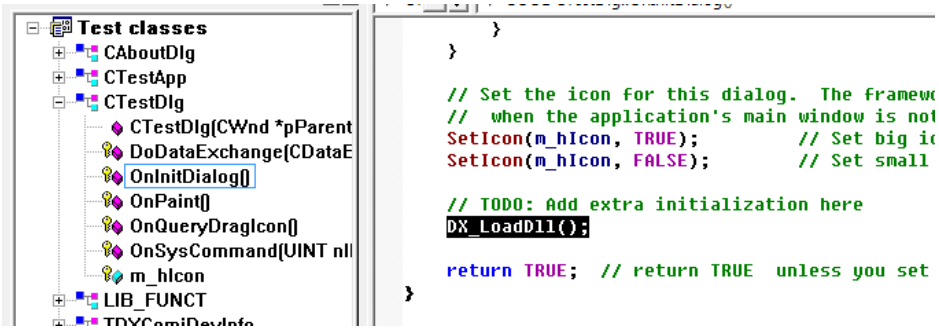


그림 1-8 DLL 로드 함수 호출 화면

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

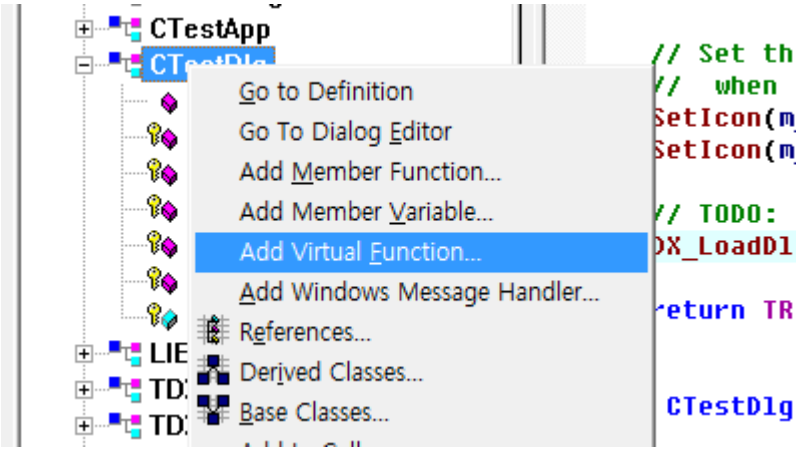


그림 1-9 가상 함수 추가

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

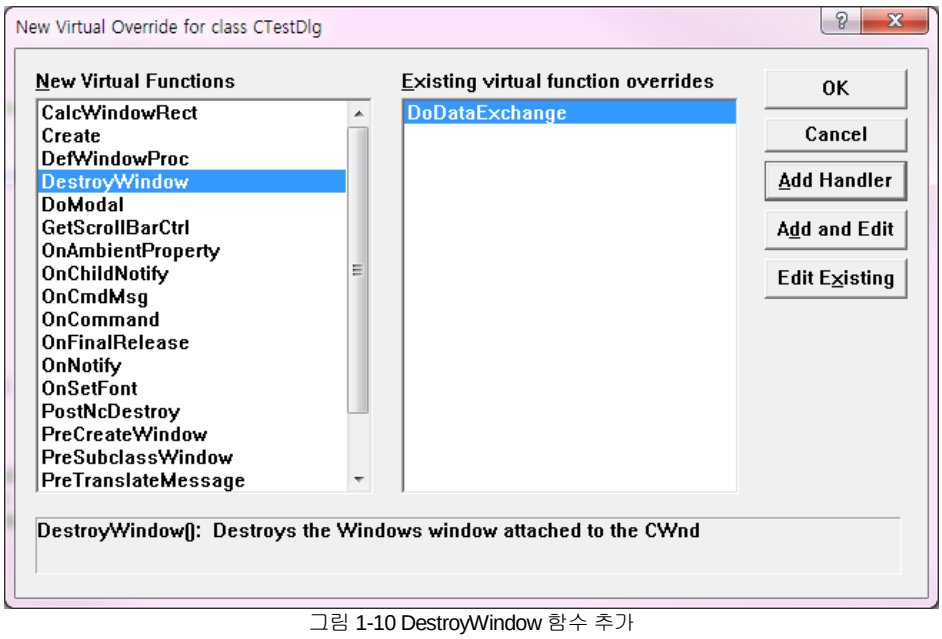


그림 1-10 DestroyWindow 함수 추가

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

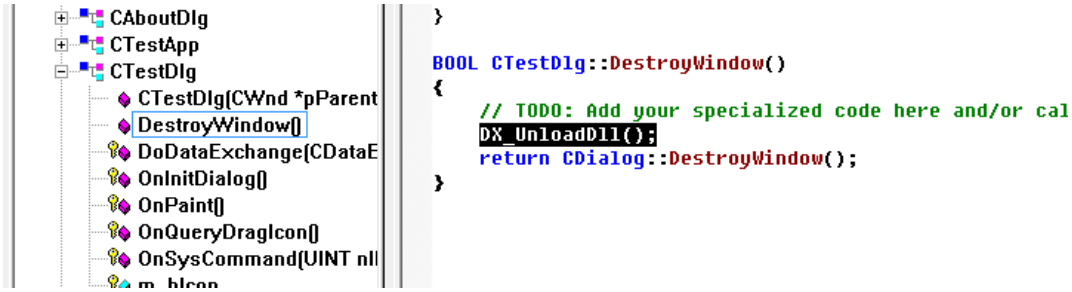


그림 1-11 DLL 로드 및 언로드 코드 추가

2.3.2 Borland C++ Builder 개발자를 위한 안내

Borland C++ Builder 는 해당 개발 환경 버전인 BCB 5, BCB 6 및 BDS 2006 버전, XESeries 에서 DX-SDK 의 인터페이스 연결 방법이 매우 유사하기 때문에 공통적인 부분으로서 안내를 해드립니다.

전체 버전(Version)의 Borland C++ Builder 에서 DX-DK 를 사용하시려면 다음의 절차를 통해 안내 받으시기 바랍니다.

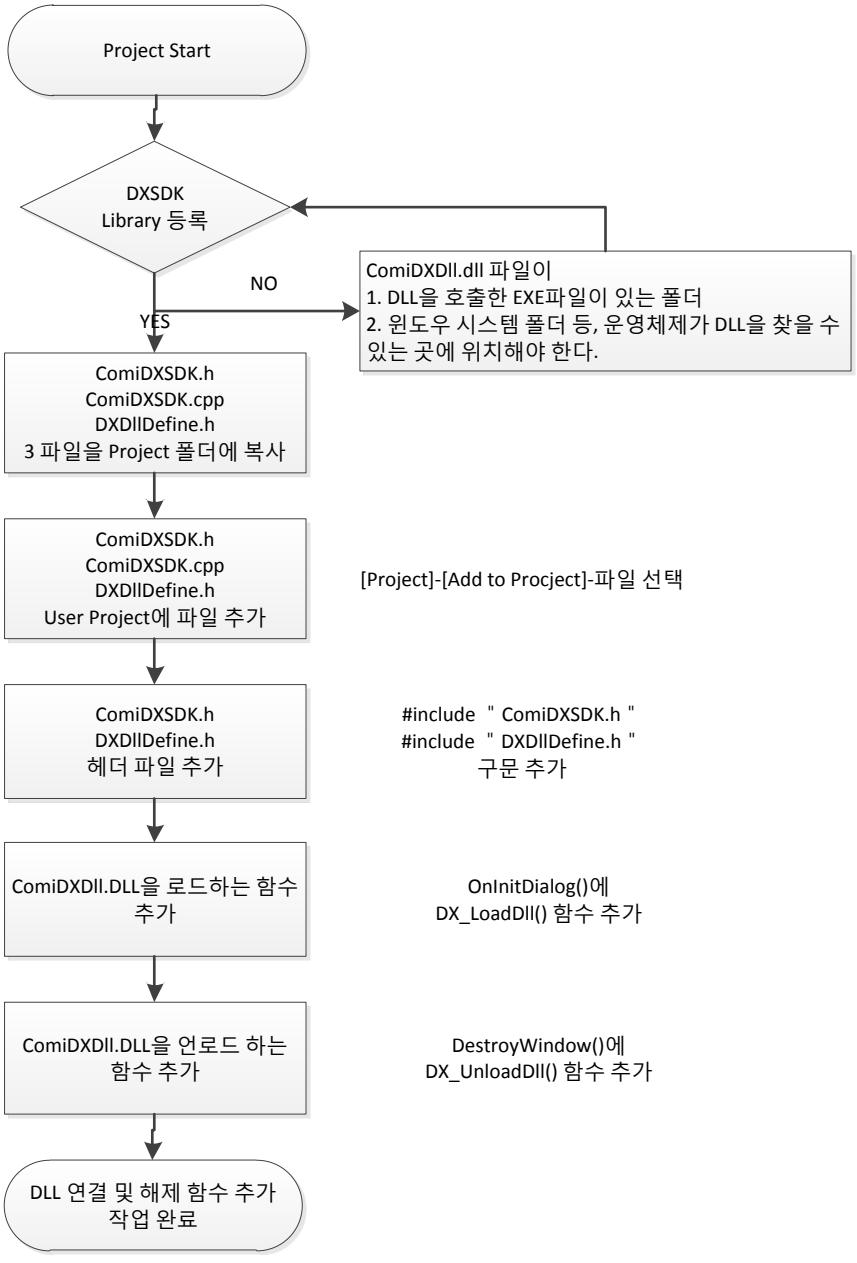


그림 2-1 Borland C++ Builder 에서 DX-SDK 사용 순서도

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 2-1 BCB 5 에서 새로운 프로젝트 생성

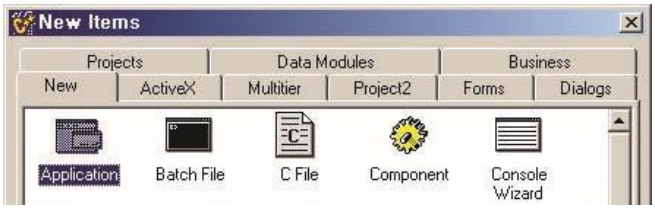


그림 2-2 BCB 5 에서 새로운 프로젝트 생성

그림 2-3 BCB 6 에서 새로운 프로젝트 생성

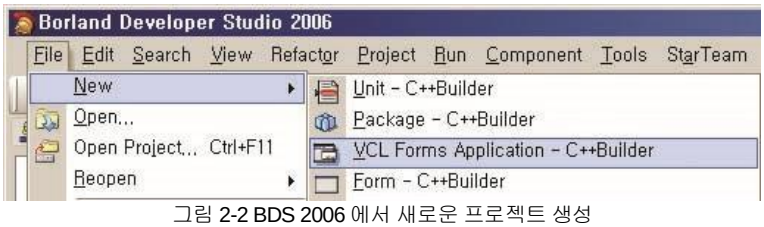


그림 2-2 BDS 2006 에서 새로운 프로젝트 생성

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

그림 2-3 DX-SDK 사용시 공통으로 사용되는 파일

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

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

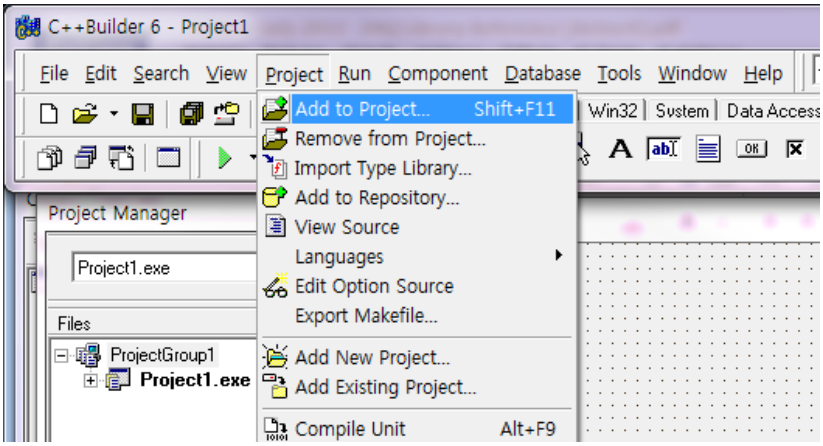


그림 2-4 C++ Builder 에서 프로젝트에 파일 추가 단계 1

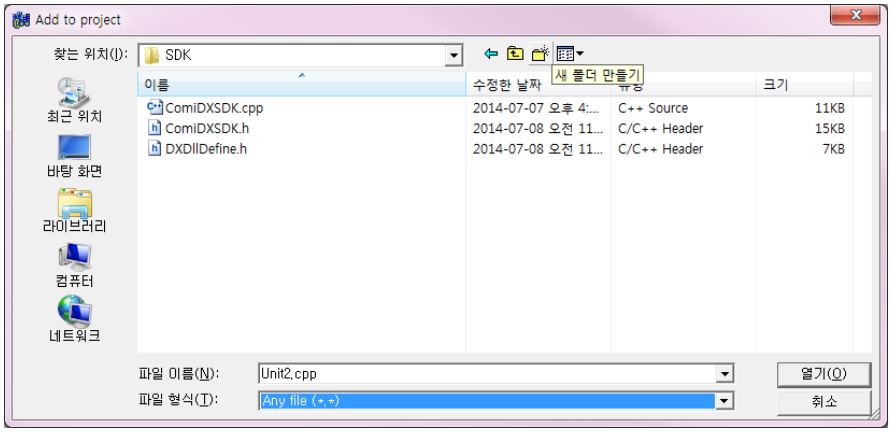


그림 2-5 C++ Builder 에서 프로젝트에 파일 추가

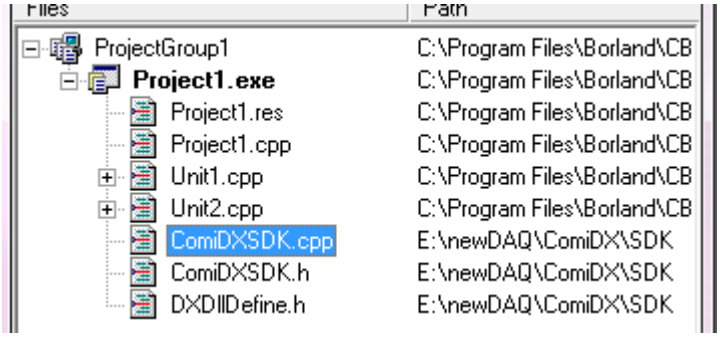


그림 2-6 Project Manager 에서 추가된 파일 확인(確認)

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

Unit1.cpp

```
//---------------------------------------------------------------------

#include <vcl.h>
#pragma hdrstop

#include "Unit1.h"

#include "ComiDXSDK.h"
#include "DXDllDefine.h"
//---------------------------------------------------------------------
#pragma package(smart_init)
#pragma resource "*.dfm"
TForm1 *Form1;
```

그림 2-7 라이브러리 사용시 필요한 헤더 파일 선언

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

Unit1.cpp

```
void __fastcall TForm1::FormCreate(TObject *Sender)
{
      DX_LoadDll();
}
//---------------------------------------------------------------------
```

그림 2-8 ComiDXSDK.dll 파일 로드

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

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

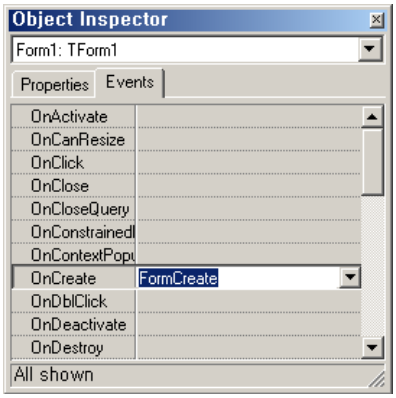


그림 2-9 OnCreate Event 추가하여 FormCreate 함수와 연결

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

Unit1.cpp

```
void __fastcall TForm1::FormCreate(TObject *Sender)
{
      DX_LoadDll();
}
//---------------------------------------------------------------------
```

그림 2-10 FormCreate 함수에 DX_LoadDll 함수 추가

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

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

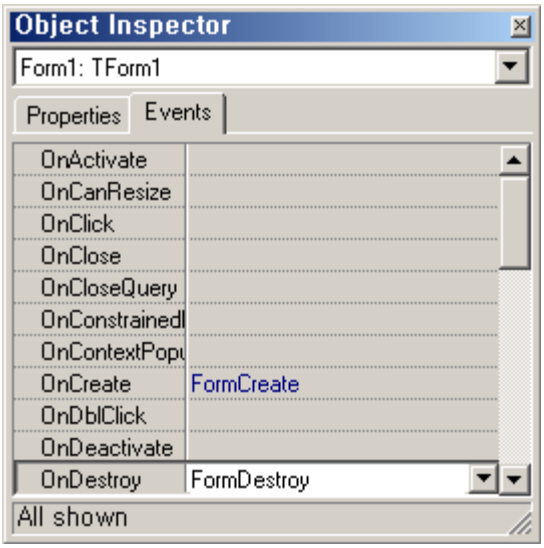


그림 2-11 응용프로그램의 종료시 DLL 이 명시적으로 UnLoad 될 수 있도록 OnDestroy Event 와 함수의 연결

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

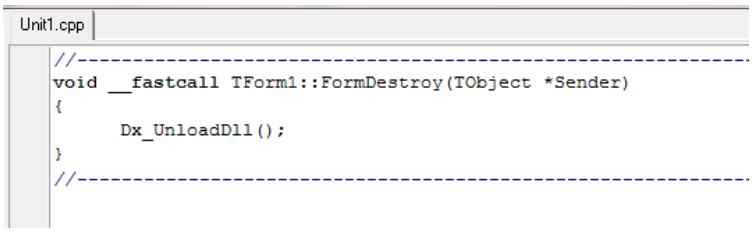


그림 2-12 FormDestroy 함수를 통하여 명시적인 UnLoadDll 함수 구현

2.3.3 Borland Delphi 개발자를 위한 안내

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

전체 버전(Version)의 Delphi 에서 DX-SDK 를 사용하시려면 다음의 절차를 통해 안내 받으시기 바랍니다.

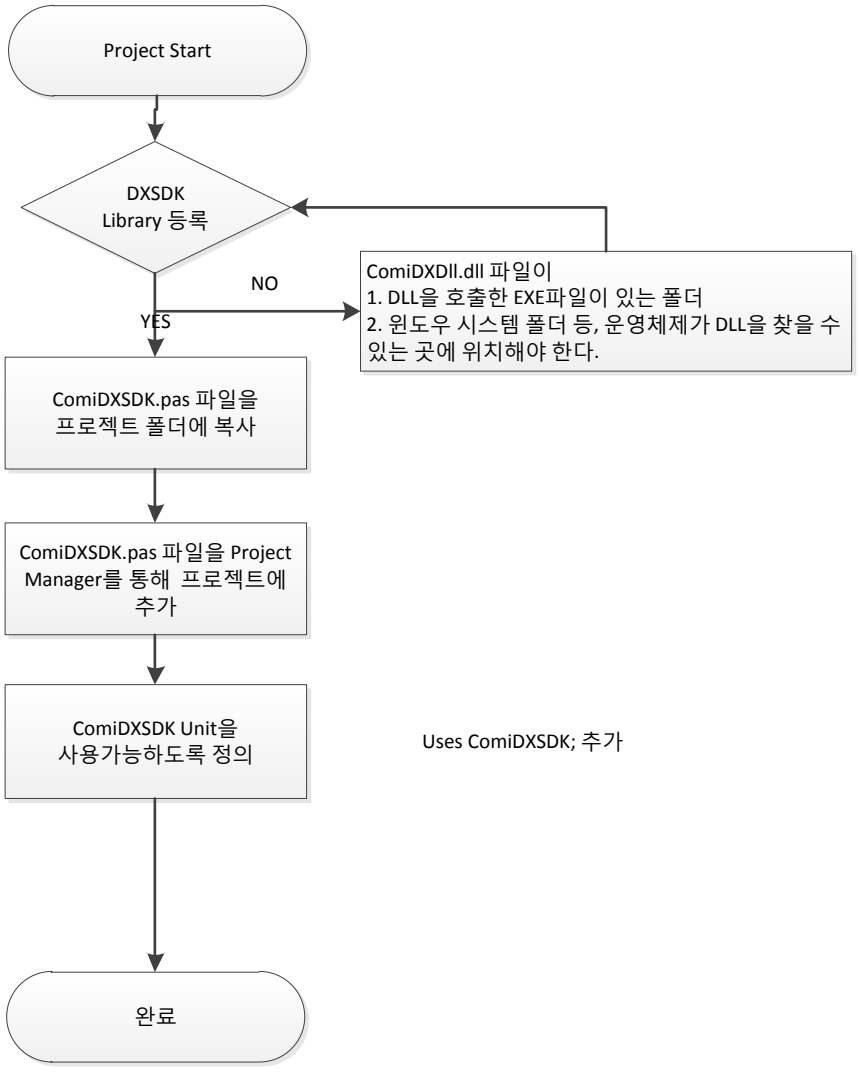


그림 2-1 Borland Delphi 에서 ComiDXSDK 사용 순서도

Delphi 모션 응용프로그램을 개발에 필요한 환경인 Borland Delphi 를 실행합니다.

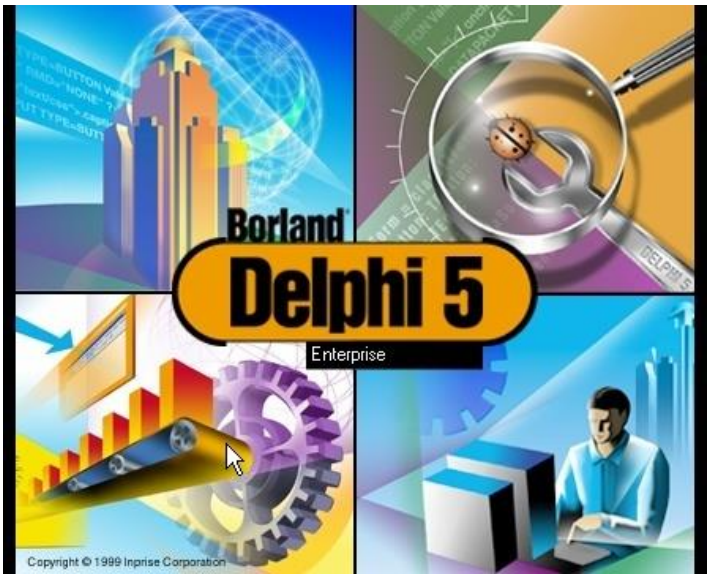


그림 2-2 Borland 社의 Delphi 5 화면

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

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

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

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

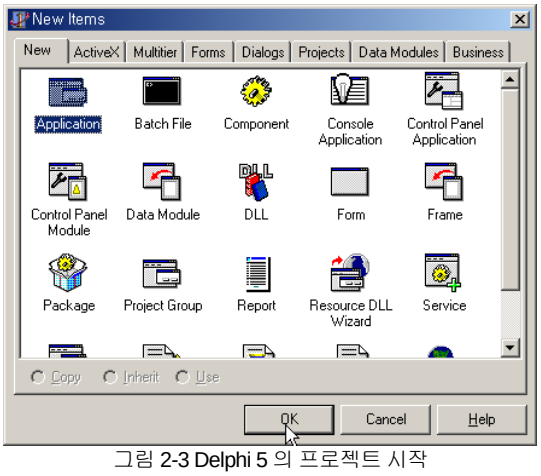


그림 2-3 Delphi 5 의 프로젝트 시작

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

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

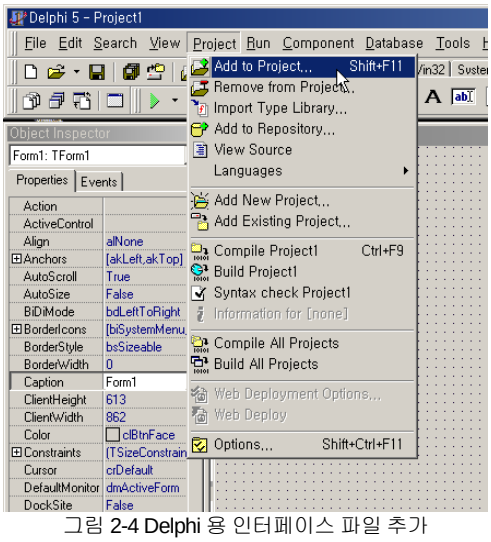


그림 2-4 Delphi 용 인터페이스 파일 추가

(주) 커미조아 DX-SDK 의 공용 인터페이스 정의 파일인 ComiDXSDK.PAS 파일을 프로젝트에 추가합니다.

그림 2-5 Delphi 용 인터페이스 파일

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

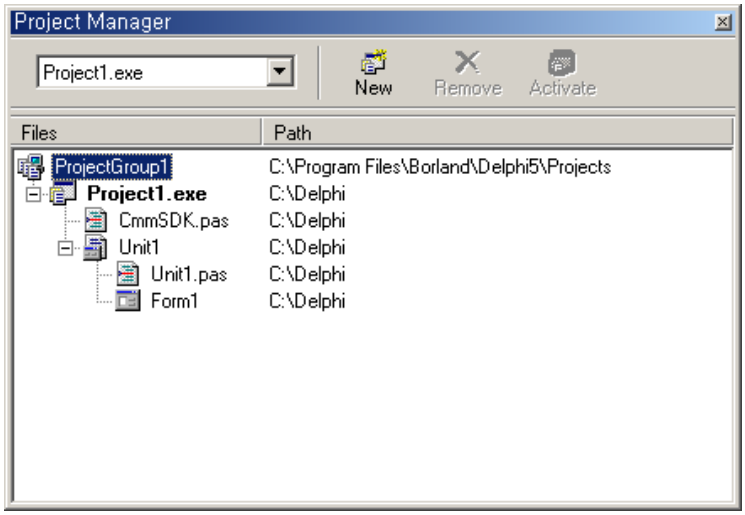


그림 2-64 Delphi 용 인터페이스 파일을 프로젝트 매니저에 추가

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

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

```
uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forr
  StdCtrls;

type
  TForm1 = class(TForm)

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

var
  Form1: TForm1;

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

{$R *.DFM}


end.
```

그림 2-75 uses 구문을 통해 DX-SDK Unit 사용

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

2.3.4 Visual Basic 개발자를 위한 안내

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

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

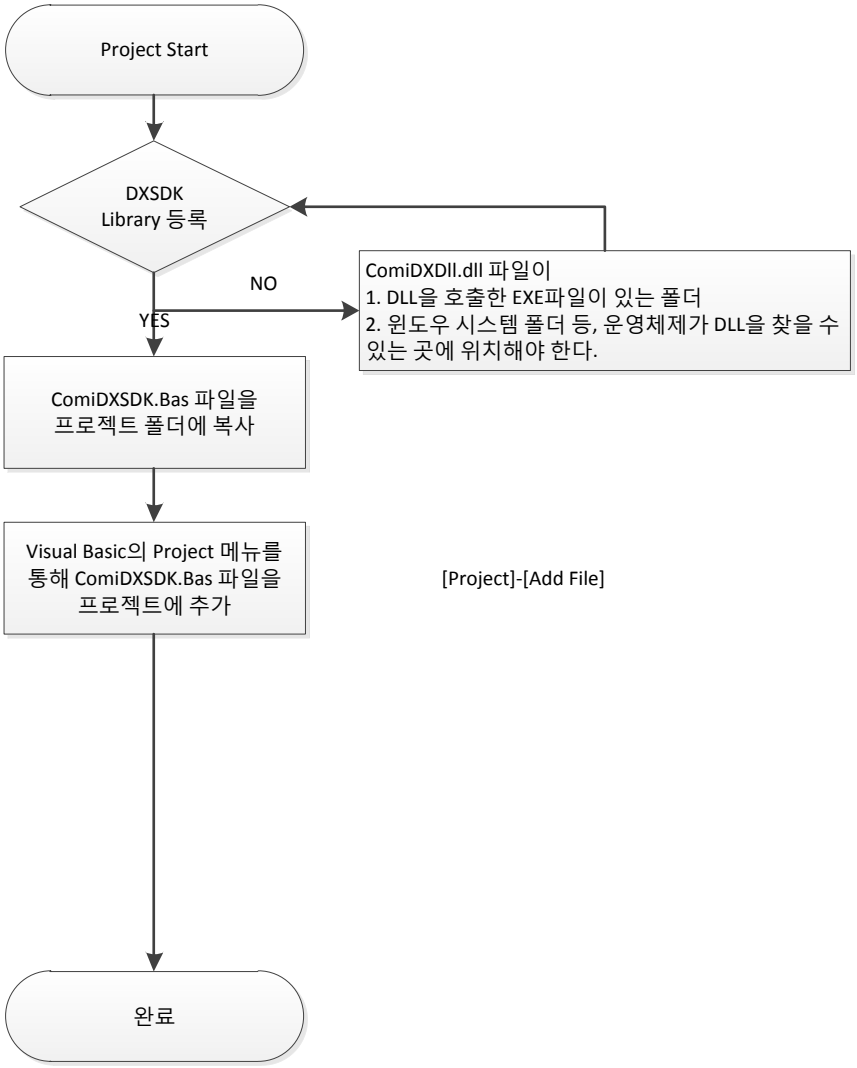


그림 4-1 Visual Basic 에서 DX-SDK 사용 순서도

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

그림 4-2 새로운 프로젝트 생성

안내	Visual Basic 에 대한 DX-SDK 지원 사항은 어떠합니까?
	DX-SDK 는 .NET 환경의 Visual Basic 까지 지원을 하고 있습니다. 특히 미리 구성된 DX-SDK 인터페이스 파일은 간단히 프로젝트에 추가만 하면, 바로 DX-SDK 를 사용하실 수 있도록 도움을 드리고 있습니다. 만약, Visual Basic 사용 고객(顧客)님께서 DX-SDK 사용에 어려움이 있으시다면, 언제든지 저희 ㈜ 커미조아 고객(顧客) 지원팀으로 연락주시기 바랍니다. 최선을 다해 지원해 드릴 것을 약속드립니다.

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

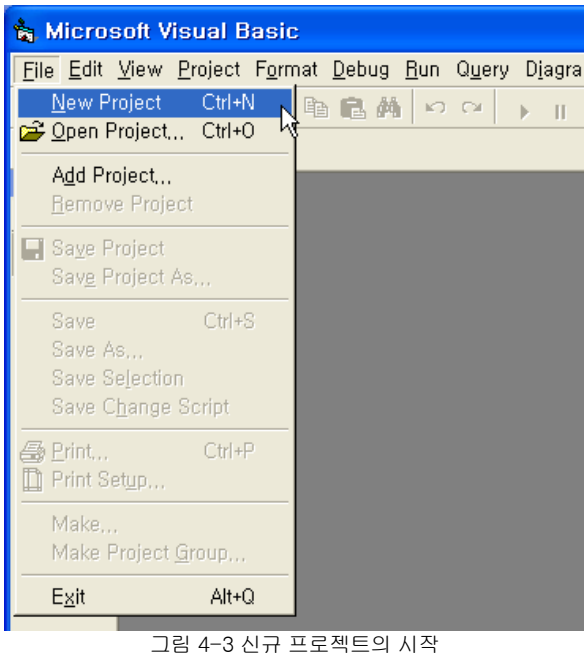


그림 4-3 신규 프로젝트의 시작

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

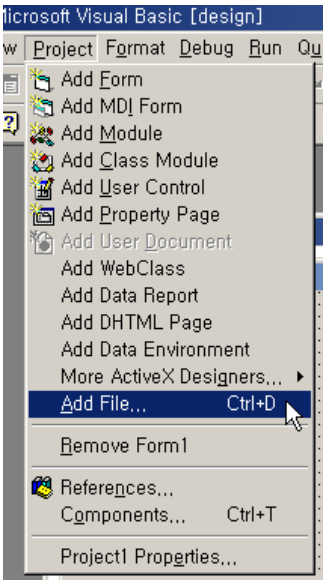


그림 4-46 프로젝트에 인터페이스 파일을 추가하기 위한 과정

그림 4-57 프로젝트에 추가 대상이 되는 ComiDXSDK.BAS 파일

ComiDXSDK.BAS 파일을 프로젝트에 추가해 주시면, 명시적인 DX-SDK 로드가 이루어지게 되며, Visual Basic 의 프로젝트에서 함께 사용하실 수 있습니다.

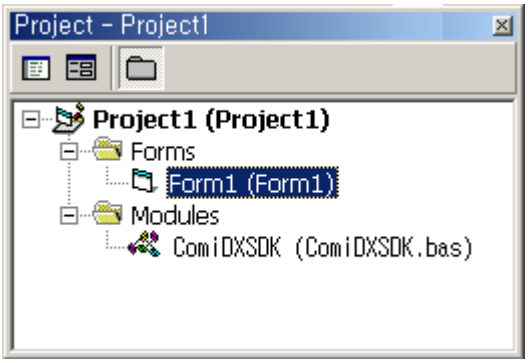


그림 4-6 프로젝트에 모듈로 추가된 ComiDXSDK.BAS 파일

추가된 인터페이스 파일을 통해 실제 응용프로그램 구현이 가능합니다.

Visual Basic 의 명시적인 DX-SDK 인터페이스에 대한 부가 안내

DX-SDK 에 포함된 Visual Basic 용 인터페이스 파일은 별도의 DLL 라이브러리(DX-SDK)의 로드(Load) 및 언로드(Unload) 가 필요 없습니다. 따라서, 고객(顧客)님의 프로젝트의 Form1 에 추가된 인터페이스 파일을 통하여, 응용프로그램 구현을 바로 시작하실 수 있습니다.

Chapter

3

DX-SDK Introduction

다년간의 보다 강력하고 편리한 라이브러리 기술개발을 통해 자신 있게 제공하여 드리는 DX-SDK 는 편리한 함수의 명명 규칙을 통해 사용자 편의성을 극대화 하였습니다. ㈜커미조아 DX-SDK 의 최신 기능과 기술은 결코 흉내 낼 수 없는 커미조아의 기술입니다. 지금 확인 하십시오.

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

3 DX-SDK 소개

3.1 함수의 명명 규칙

DX-SDK 에서 제공하는 모든 함수는 다른 API 함수와 이름이 중복되는 것을 피하기 위하여 아래의 예와 같이 "DX_"이라는 첨두어가 붙습니다.

DX_LoadDll(), DX_UnloadDll(), DX_GnDeviceLoad(), DX_GnDeviceUnload(),...

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

○. General Functions (Gn): DX_GnDeviceLoad(), DX_GnGetDevInfo (), ...

○. 디지털 입출력 함수들 (Dio): DX_DioSetUsage(), DX_DioPutOne (), ...

○. 아날로그 입력 함수들 (Ad): DX_AdSetInputType(), DX_AdGetVolt (), ...

○. 아날로그 출력 함수들 (Da): DX_DaClear(), DX_DaOut(), ...

○. 카운터 함수들 (Cnt): DX_CntSetConfig(), DX_CntStart(), ...

3.2 데이터형 표기

당사의 DX-SDK 인터페이스는 매뉴얼에서 명시한 윈도우 표준 Dynamic Link Library 를 지원하는 어떠한 개발 환경에서도 사용 가능합니다. 하지만 데이터형에 대한 이름은 개발환경에 따라서 서로 다릅니다. 데이터 형에 대한 이름은 언어나 컴파일러에 따라서 서로 다릅니다. 따라서 본 매뉴얼에서는 데이터 형 표기를 [표 3-1]과 같이 통일하여 표기 합니다. 이에 대한 각 컴파일러의 대응되는 데이터 형 표기는 [표 3-1]을 참조하여 사용하시기 바랍니다.

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

Data type	Description	C/C++	VB 6.0	Delphi	C#
VT_EMPTY	반환값이 없는 데이터 표현형	void	-	-	void
VT_HANDLE	운영체제가 특정한 정보를 유지하기 위해서, 메모리에 유지하는 정보 블록에 붙은 고유 번호	void *	Long (ByRef)	THandle	IntPtr
VT_PHANDLE	운영체제가 특정한 정보를 유지하기 위해서, 메모리에 유지하는 정보 블록에 붙은 고유 번호에 대한 주소 값(포인터) 또는 배열주소 표현형	Void**			
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	long *	Long (ByRef)	PLongInt	Int[]

	표현형				
VT_R4	4 바이트 부호 있는 실수 표현형	float	Double (ByVal)	Double	Float
VT_PR4	4 바이트 부호 있는 실수 변수의 주소 값(포인터) 또는 배열주소 표현형	float *	Double (ByRef)	PDouble	float[]
VT_R8	8 바이트 부호 있는 실수 표현형	double	Double (ByVal)	Double	double
VT_PR8	8 바이트 부호 있는 실수변수의 주소 값(포인터) 또는 배열주소 표현형	double *	Double (ByRef)	PDouble	double[]
VT_STR	선형 메모리 상의 문자열 선두 주소를 지시하는 4 바이트 주소 표현형	char *	String (ByVal)	PChar	String
VT_STRUCT	라이브러리 내에서 지정된 특정한 구조체 표현형				

표 5 언어독립적 데이터 표기 및 각 언어별 대응 데이터 형

Chapter

4

General Functions

표준 함수와 관계된 CMD-SDK 라이브러리의 기능을 통해 최고의 성능과 최적의 개발환경을 제공합니다. 뛰어난 성능을 기반으로 한 CMD-SDK 라이브러리의 즐거움을 이제 함께 하십시오.

본 장에서는 COMI-DAQ 제품의 종류와 관계없이 모든 제품이 공통적으로 사용하는 함수들을 설명합니다. 이 중 DX_GnLoadDevice() 함수는 프로그램 시작 시에 반드시 수행해 주어야 하는 함수입니다. 그리고 나머지 기타 함수들은 필요에 따라 사용하는 함수들입니다.

4 General Functions

4.1 디바이스 시작/종료

4.1.1 함수 요약

General Functions 중 디바이스 시작/종료에 관련된 관련된 함수는 다음과 같습니다.

Summary of Functions
❒ VT_I4 DX_GnLoadDevice ([out]VT_PI4 numDevs, [out]VT_PHANDLE hDevList) 장치를 로드(Load)하고, 초기화 합니다.
❒ VT_I4 DX_GnIsDevLoaded([out]VT_PBOOL IsLoaded) 장치의 로드 여부를 반환합니다.
❒ VT_I4 DX_GnGetDevInfo ([in]VT_HANDLE hDevice, [out]VT_STRUCT pDevInfo) 장치의 정보를 가져옵니다.
❒ VT_I4 DX_GnUnloadDevice() 장치를 언로드(Unload) 합니다.

4.1.2 함수 설명

NAME

DX_GnLoadDevice

- 디바이스 로드(Device Load)

INFORMATION

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

SYNOPSIS

❒ VT_I4 DX_GnLoadDevice ([out]VT_PI4 numDevs, [out]VT_PHANDLE hDevList)

DESCRIPTION

시스템에 설치된 하드웨어 장치를 로드하고 장치를 초기화 합니다.

이 함수는 DX-SDK 라이브러리의 다른 함수가 호출되기 전에 반드시 한번은 수행되어야 합니다. 일반적으로 프로그램의 시작부분에서 수행해주면 됩니다.[1]

PARAMETER

▶ ***numDevs*** : 연결된 DX Device 의 개수를 반환합니다.

▶ ***hDevList*** : 연결된 DX Device 의 핸들 리스트를 반환합니다. 이 값을 실제 연결된 장치의 수보다 크거나 같은 크기를 가져야 합니다.

RETURN VALUE

□ 함수 수행의 성공 여부.

Value	Meaning
음수	디바이스 로드 실패. 자세한 내용은 '에러 처리'편을 참고합니다.
0 (dxERR_NONE)	디바이스 로드 성공.

SEE ALSO

DX_GnGetDevInfo, DX_GnUnloadDevice

EXAMPLE

[C / C++]

```
LONG nNumDevs=0;
HANDLE hDevList[10];

If( DX_GnLoadDevice(&nNumDevs, hDevList) )
        // 에러 메시지 출력
```

NAME

DX_GnIsDevLoaded

- 디바이스 로드 여부 반환

INFORMATION

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

SYNOPSIS

❒ VT_I4 DX_GnIsDevLoaded ([out]VT_PBOOL IsLoaded)

DESCRIPTION

DX_gnLoadDevice 함수를 통하여 장치가 로드 되었는지에 대한 여부를 반환합니다.

PARAMETER

▶ ***IsLoaded*** : 장치의 로드 여부를 반환합니다.

RETURN VALUE

□ 함수 수행의 성공 여부.

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

SEE ALSO

DX_GnLoadDevice, DX_GnGetDevInfo, DX_GnUnloadDevice

EXAMPLE

[C / C++]

```
BOOL bLoaded = FALSE;
```

```
DX_GnIsDevLoaded(&bLoaded) ;
If( bLoaded )
{
    DX_GnUnloadDevice();
}
```

NAME

DX_GnGetDevInfo

- 디바이스 정보 반환

INFORMATION

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

SYNOPSIS

❒ VT_I4 DX_GnGetDevInfo ([in]VT_HANDLE hDevice, [out]VT_STRUCT pDevInfo)

DESCRIPTION

입력한 Device 의 핸들값을 바탕으로 해당 Device 의 정보 구조체를 얻어옵니다.

PARAMETER

▶ ***hDevice*** : DX_GnLoadDevice()로부터 얻어온 Device 의 핸들값 입니다.

▶ ***pDevInfo*** : 해당 구조체는 TDXComiDevInfo 타입으로 장치의 SubSysID, 이름, 채널에 대한 정보를 반환합니다.

Value	Meaning
wSubSysID	연결된 장치의 SubsysID 입니다.
nInstance	연결된 장치의 연결 순서 번호 입니다.
szDevName	연결된 장치의 이름 문자열 입니다.
bDevCaps	기능 마스크 입니다. 0 bit – A/D, 1 bit – D/A, 2bit – DIO, 3bit - CNT
NNum(기능)Chan	각 기능별 채널 수 입니다.

RETURN VALUE

□ 디바이스 로드 상태 여부.

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

SEE ALSO

DX_GnLoadDevice, DX_GnUnloadDevice

EXAMPLE

[C / C++]

```
TDXComiDevInfo DevInfo;

if(  DX_GnGetDevInfo(m_hDevice, &DevInfo) )
        // 에러 메시지 출력

printf("Device SubSysID : %x\n", DevInfo.wSubSysID);
printf("Device Name : %s\n", DevInfo.szDevName);
printf("Device AD Channel count : %d\n", DevInfo.nNumAdChan);
printf("Device DA Channel count : %d\n", DeviInfo.nNumDaChan);
```

NAME

DX_GnUnloadDevice

- 디바이스 언로드 (Device Unload)

INFORMATION

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

SYNOPSIS

❒ VT_I4 DX_GnUnloadDevice ()

DESCRIPTION

장치를 언로드(Unload)합니다. 일반적으로는 DX-SDK 인스턴스가 Destroy 될 때 자동으로 언로드를 수행하므로 사용자가 이 함수를 수행할 필요는 없습니다.

사용자가 직접 이 함수를 사용해야만 하는 경우에는 DX-SDK 인스턴스가 Destroy 되기 이전에 수행되어야만 합니다.

RETURN VALUE

□ 함수 수행의 성공 여부.

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

SEE ALSO

DX_GnLoadDevice, DX_GnGetDevInfo

EXAMPLE

[Visual C++]

```
void CDaqTestDlg::OnDestroy()
{
  // 인스턴스가 Destroy 되기 이전에 함수를 호출해야 하므로
  // CDialog::OnDestroy() 함수가 호출되기 전에 수행합니다.
  DX_GnUnloadDevice();

  CDialog::OnDestroy();
}
```

4.2 에러 처리

4.2.1 함수 요약

DX-SDK 라이브러리 사용 중에 발생할 수 있는 에러를 체크하고 처리하는 함수들을 소개합니다. 이러한 에러처리 루틴들은 정상적인 상황에서는 사용하지 않아도 무방하나 시스템의 안전성이나 프로그램의 디버깅을 용이하게 하기 위해서 사용하는 것이 바람직합니다.

DX-SDK 라이브러리에서 제공하는 에러처리 관련 함수들은 다음과 같으며 모든 제품에서 사용 가능합니다.

Summary of Functions
❒ VT_I4 DX_ErrGetLastErrorCode() 마지막에 발생한 에러 코드를 반환합니다.
❒ VT_I4 DX_ErrShowLastError ([in]HWND hParentWnd) 마지막에 발생한 에러 메시지를 메시지박스 형태로 디스플레이합니다.

4.2.2 함수 설명

NAME

DX_ErrGetLastErrorCode

- 마지막에 발생한 에러코드 확인

INFORMATION

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

SYNOPSIS

❒ VT_I4 DX_ErrGetLastErrorCode ()

DESCRIPTION

마지막에 발생된 에러코드를 반환합니다.

단, 에러가 발생한 이후에 에러처리 함수를 제외한 DX-SDK 라이브러리 함수가 호출되는 시점에서 에러코드는 자동으로 리셋됩니다.

RETURN VALUE

마지막에 발생된 에러 코드를 반환합니다.
에러 코드에 대한 내용은 “Appendix B :: 에러 코드 일람표”를 참조하시기 바랍니다.

SEE ALSO

DX_ErrShowLastError

NAME

DX_ErrShowLastError

- 마지막에 발생한 에러 메시지 디스플레이

INFORMATION

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

SYNOPSIS

❒ VT_I4 DX_ErrShowLastError ([in]HWND hParentWnd)

DESCRIPTION

마지막 에러에 대하여 에러메시지를 메시지박스 형태로 디스플레이합니다.

PARAMETER

▶ hParentWnd : 사용자 어플리케이션의 핸들을 설정합니다.

RETURN VALUE

반환 값은 무의미한 값이며 무조건 FALSE 값이 반환됩니다.

SEE ALSO

DX_ErrGetLastErrorCode

4.3 디버그 로그

4.3.1 함수 요약

DX-SDK 는 보다 편리한 개발환경을 제공하기 위해서 “디버그 로깅(Debug logging)” 기능을 제공합니다.
“디버그 로깅” 기능은 사용자의 프로그램에서 DX-SDK 라이브러리의 함수가 호출할 때마다 그 시각과 호출된 내용 그리고 그 순간의 에러 상황 등을 지정한 로그 파일 또는 TRACE 로 기록하는 기능입니다.

디버그 로그 기능과 관련된 함수들은 다음과 같습니다.

Summary of Functions
❒ VT_I4 DX_DLogSetup ([in]VT_I4 Type, [in]VT_I4 Level) 디버그 로그 기능의 환경을 설정합니다.
❒ VT_I4 DX_DLogSetFilePath([in] VT_STR szFilePath) 디버그 로그 기능을 로그파일로 기록할 경우 파일 경로를 설정합니다.
❒ VT_I4 DX_DLogAddComment ([in] VT_STR szComment) 디버그 로그 시에 사용자 정의 주석(Comment)을 추가 합니다.
❒ VT_I4 DX_DLogGetCurType ([out]VT_PI4 CurType) 디버그 로그 시에 현재 설정된 로그 타입을 반환합니다.
❒ VT_I4 DX_DLogGetCurLevel ([out]VT_PI4 CurLevel) 디버그 로그 시에 현재 설정된 로그 레벨을 반환합니다.
❒ VT_I4 DX_DLogGetCurFilePath ([out]VT_STR szFilePath) 디버그 로그 기능을 로그파일로 기록 할 경우 파일 경로를 반환합니다.

4.3.2 함수 설명

NAME

DX_DLogSetup

- 디버그 로그(Debug Log) 기능 설정

INFORMATION

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

SYNOPSIS

❒ VT_I4 DX_DLogSetup ([in]VT_I4 Type, [in]VT_I4 Level)

DESCRIPTION

디버그 로그 기능의 환경을 설정합니다.
디버그 로그 기능의 타입과 레벨을 설정합니다.

PARAMETER

▶ ***Type*** : 디버그 로그 방식을 설정합니다.

Value	Meaning
0 (dxDLOG_TRACE)	디버그 로그를 TRACE 를 통하여 로깅합니다.
1 (dxDLOG_FILE)	디버그 로그를 파일을 통하여 로깅합니다.

▶ ***Level*** : 디버그 레벨을 설정합니다.

Value	Meaning
0 (dxDLOG_DISABLE)	디버그 로그 기능을 Disable 시킵니다.
1 (dxDLOG_SET)	디버그 레벨을 LEVEL1 으로 설정합니다 => SET 함수만 로깅.
2 (dxDLOG_GET)	디버깅 레벨을 LEVEL2 로 설정합니다 => SET/GET 함수만 로깅.
3 (dxDLOG_OUT)	디버깅 레벨을 LEVEL2 로 설정합니다 => 1,2 포함 출력 함수만 로깅.
4 (dxDLOG_ALL)	디버깅 레벨을 LEVEL2 로 설정합니다 => 모든 함수 로깅.

RETURN VALUE

□ 함수 수행의 성공 여부.

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

SEE ALSO

DX_DLogSetFilePath, DX_DLogAddComment, DX_DLogGetCurType, DX_DLogGetCurLevel, DX_DLogGetCurFilePath

NAME

DX_DLogSetFilePath

- 디버그 로그 시 파일 저장 위치 설정

INFORMATION

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

SYNOPSIS

❒ VT_I4 DX_DLogSetFilePath ([in] VT_STR szFilePath)

DESCRIPTION

디버그 로그 기능이 활성화되고 파일 방식으로 선택하였을 경우 저장할 파일 위치를 설정합니다.

PARAMETER

▶ ***szFilePath*** : 파일 경로와 함께 파일 위치를 지정합니다.
Ex) C:\\Log\\Log.txt

RETURN VALUE

□ 함수 수행의 성공 여부.

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

SEE ALSO

DX_DLogSetup, DX_DLogAddComment, DX_DLogGetCurType, DX_DLogGetCurLevel, DX_DLogGetCurFilePath

NAME

DX_DLogAddComment

- 디버그 로그 시 주석(Comment) 추가

INFORMATION

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

SYNOPSIS

❒ VT_I4 DX_DLogAddComment ([in]VT_STR szComment)

DESCRIPTION

디버그 로그 기능이 활성화되었을 때 사용자 정의 주석(Comment)을 추가합니다.

PARAMETER

▶ ***szComment*** : 추가할 사용자 정의 주석 문자열.

RETURN VALUE

□ 함수 수행의 성공 여부.

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

SEE ALSO

DX_DLogSetup, DX_DLogSetFilePath, DX_DLogGetCurType, DX_DLogGetCurLevel, DX_DLogGetCurFilePath

REFERENCE

□ Visual C++이나 Visual Basic 을 사용하는 경우에는 주석문자열을 일반적으로 사용하는 문자열로 지정할 수 있습니다. 그러나 Borland C++ Builder 를 사용하는 경우에는 아래 예와 WideString 을 사용하여야 합니다.

```
WideString Comment = "This is comment.";
ComiDaq1->DlogAddComment (Comment.c_bstr());
```

NAME

DX_DLogGetCurType

- 현재 설정된 디버그 로그 타입 반환

INFORMATION

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

SYNOPSIS

❒ VT_I4 DX_DLogGetCurType([out]VT_PI4 CurType)

DESCRIPTION

디버그 로그 기능 사용 시 현재 설정되어 있는 로그 타입을 반환합니다.

PARAMETER

▶ ***CurType*** : 현재 설정되어 있는 로그 타입

Value	Meaning
0 (dxDLOG_TRACE)	디버그 로그를 TRACE 를 통하여 로깅합니다.
1 (dxDLOG_FILE)	디버그 로그를 파일을 통하여 로깅합니다.

RETURN VALUE

□ 함수 수행의 성공 여부.

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

SEE ALSO

DX_DLogSetup, DX_DLogSetFilePath, DX_DLogAddComment, DX_DLogGetCurLevel, DX_DLogGetCurFilePath

NAME

DX_DLogGetCurLevel

- 현재 설정된 디버그 로그 레벨 반환

INFORMATION

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

SYNOPSIS

❒ VT_I4 DX_DLogGetCurLevel([out]VT_PI4 CurLevel)

DESCRIPTION

디버그 로그 기능 사용 시 현재 설정되어 있는 로그 레벨을 반환합니다.

PARAMETER

▶ ***CurLevel*** : 현재 설정되어 있는 로그 레벨

Value	Meaning
0 (dxDLOG_DISABLE)	디버그 로그 기능을 Disable 시킵니다.
1 (dxDLOG_SET)	디버그 레벨을 LEVEL1 으로 설정합니다 => SET 함수만 로깅.
2 (dxDLOG_GET)	디버깅 레벨을 LEVEL2 로 설정합니다 => SET/GET 함수만 로깅.
3 (dxDLOG_OUT)	디버깅 레벨을 LEVEL2 로 설정합니다 => 1,2 포함 출력 함수만 로깅.
4 (dxDLOG_ALL)	디버깅 레벨을 LEVEL2 로 설정합니다 => 모든 함수 로깅.

RETURN VALUE

□ 함수 수행의 성공 여부.

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

SEE ALSO

DX_DLogSetup, DX_DLogSetFilePath, DX_DLogAddComment, DX_DLogGetCurType,

DX_DLogGetCurFilePath

NAME

DX_DLogGetCurFilePath

- 현재 설정된 디버그 로그 파일 경로 반환

INFORMATION

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

SYNOPSIS

❒ VT_I4 DX_DLogGetCurFilePath([in]VT_STR szFilePath)

DESCRIPTION

디버그 로그 기능 사용시 현재 설정되어 있는 파일의 경로와 파일명을 반환합니다.

PARAMETER

▶ ***szFilePath*** : 파일 경로와 함께 파일 위치.
Ex) C:\\Log\\Log.txt

RETURN VALUE

□ 함수 수행의 성공 여부.

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

SEE ALSO

DX_DLogSetup, DX_DLogSetFilePath, DX_DLogGetCurType, DX_DLogGetCurLevel, DX_DLogAddComment

REFERENCE

□ Visual C++이나 Visual Basic 을 사용하는 경우에는 주석문자열을 일반적으로 사용하는 문자열로 지정할 수 있습니다. 그러나 Borland C++ Builder 를 사용하는 경우에는 아래 예와 WideString 을 사용하여야 합니다.

```
WideString Comment = "This is comment.";
DX_DLogGetCurFilePath (Comment.c_bstr());
```

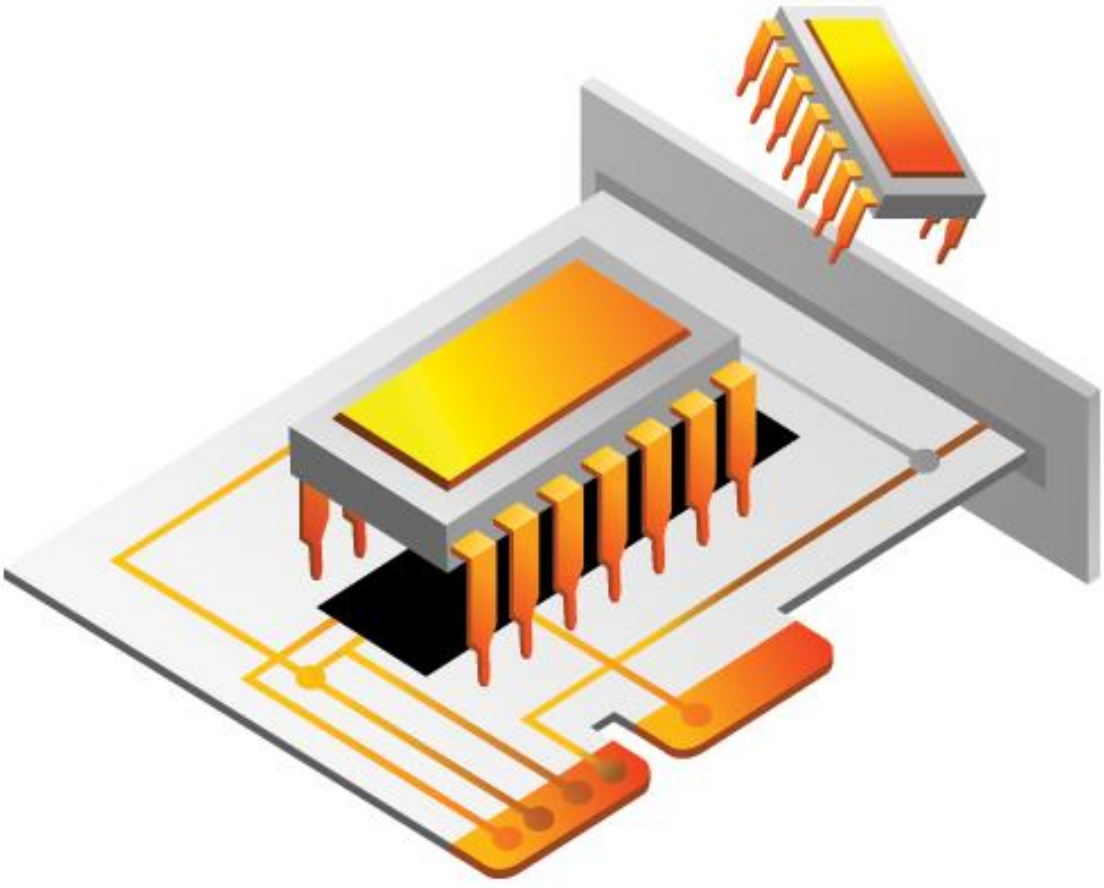
Chapter

5

Analog Input Functions

아날로그 입력 장치는 아날로그 전압이나 전류 레벨을 디지털 정보로 변환해주는 장치입니다.

아날로그 입력 장치는 아날로그 전압이나 전류레벨을 디지털 정보로 변환해주는 장치입니다. 변환된 디지털 데이터는 컴퓨터로 전달되고 컴퓨터는 이를 처리하고, 저장하거나 화면에 표시합니다. 일반적으로 아날로그 입력 장치는 온도센서, 압력센서, 유량센서 등과 같이 아날로그적인 상태를 측정하는 센서들을 계측하는데 사용됩니다.



5 아날로그 입력 함수

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

DX-SDK 에서 지원하는 A/D 방식에는 두 가지가 있습니다. 첫 번째는 Single point A/D 방식이며 두 번째는 A/D Scan 방식입니다.

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

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

5.1 일반적인 아날로그 입력

이 단원에서는 Single point A/D 함수를 포함한 일반적으로 사용되는 A/D 관련 함수들을 소개합니다. 일반적인 아날로그 입력 관련 함수 리스트 및 사용 가능 디바이스는 다음과 같습니다.

Function	Device
DX_AdSetInputType	COMI-DX10x, COMI-DX20x
DX_AdSetRange	
DX_AdGetRange	
DX_AdGetDigit	
DX_AdGetVolt	

[표 5-1] Analog Input 일반 함수 리스트 및 사용 가능 디바이스

5.1.1 함수 요약

Single Point A/D 와 관련된 함수들의 리스트는 다음과 같습니다.

Summary of Functions
❒ VT_I4 DX_AdSetInputType ([in]VT_HANDLE hDevice, [in]VT_I4 InputMode) 아날로그 입력 신호의 연결 형식을 소프트웨어로 설정합니다.
❒ VT_I4 DX_AdGetInputType ([in]VT_HANDLE hDevice, [in]VT_PI4 InputMode) 아날로그 입력 신호의 연결 형식을 반환합니다.
❒ VT_I4 DX_AdSetRange ([in]VT_HANDLE hDevice, [in]VT_I4 Channel, [in]VT_I4 Range) 각 A/D 채널의 입력 범위를 설정합니다.
❒ VT_I4 DX_AdGetRange ([in]VT_HANDLE hDevice, [in]VT_I4 Channel, [out]VT_PI4 Range) 각 A/D 채널의 입력 범위를 반환합니다.
❒ VT_I4 DX_AdGetDigit ([in]VT_HANDLE hDevice, [in]VT_I4 Channel, [out]VT_PI4 Digit) 각 A/D 채널의 입력 범위를 정수(Digit) 값으로 반환합니다.
❒ VT_I4 DX_AdGetVolt ([in]VT_HANDLE hDevice, [in]VT_I4 Channel, [out]VT_PR8 Volt) 대상 채널에 대해 A/D 변환을 수행 후 그 값을 전압(Voltage) 값으로 반환합니다.

5.1.2 함수 설명

NAME

DX_AiSetInputType / DX_AiGetInputType

- A/D 신호 연결 형식을 소프트웨어로 설정 / 반환

INFORMATION

- Analog Input
- VC++ (6, 7, 8)/VB
- BCB/Delphi
- Level 1
- 위험 요소 없음

SYNOPSIS

- ❒ VT_I4 DX_AiSetInputType ([in]VT_HANDLE hDevice, [in]VT_I4 InputMode)
- ❒ VT_I4 DX_AiGetInputType ([in]VT_HANDLE hDevice, [in]VT_PI4 InputMode)

DESCRIPTION

이 함수는 아날로그 입력 신호의 연결 형식을 소프트웨어로 설정 / 반환 합니다. 아날로그 입력 신호의 연결 형식에는 Single ended 방식과 Differential 방식의 두 가지가 있습니다. (하드웨어 매뉴얼 참조)

PARAMETER

▶ ***hDevice*** : DX_GnLoadDevice()로부터 얻어온 Device 의 핸들값 입니다.

▶ ***InputMode*** : 아날로그 입력 신호의 연결 형식입니다. 이 값은 다음 중 하나의 값이어야 합니다. 컴퓨터 부팅시의 기본값은 0(cmAI_SINGLE)입니다.

Value	Meaning
0 (dxAI_ SINGLE)	아날로그 입력 신호의 연결형식을 Single ended 로 설정합니다.
1 (dxAI_DIFF)	아날로그 입력 신호의 연결형식을 Differential 로 설정합니다.

RETURN VALUE

□ 함수 수행 결과

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

NAME

DX_AiSetRange / DX_AiGetRange

– 각 A/D 채널의 입력 범위 설정 / 반환

INFORMATION

Analog Input

VC++ (6, 7, 8)/VB

BCB/Delphi

Level 1

위험 요소 없음

SYNOPSIS

❒ VT_I4 DX_AdSetRange ([in]VT_HANDLE hDevice, [in]VT_I4 Channel, [in]VT_I4 Range)

❒ VT_I4 DX_AdGetRange ([in]VT_HANDLE hDevice, [in]VT_I4 Channel, [out]VT_PI4 Range)

DESCRIPTION

이 함수는 각 A/D 채널의 입력 범위를 설정/ 반환합니다.

PARAMETER

▶ ***hDevice*** : DX_GnLoadDevice()로부터 얻어온 Device 의 핸들값 입니다.

▶ ***Channel*** : A/D 입력 범위를 설정 할 채널 번호를 지정합니다. 채널 번호는 0 부터 시작합니다.

▶ ***Range*** : 아날로그 입력 채널의 입력 범위 입니다.

Value	Meaning
0 (dxRANGE_V0)	0v ~ 5v
1 (dxRANGE_V1)	0v ~ 10v
2 (dxRANGE_V2)	0v ~ 10.8v
3 (dxRANGE_V3)	-5v ~ 5v
4 (dxRANGE_V4)	-10v ~ 10v
5 (dxRANGE_V5)	-10.8v ~ 10.8v

RETURN VALUE

□ 함수 수행 결과

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

NAME

DX_AiGetDigit

– 대상 A/D 채널의 입력 정수(Digit) 값 반환

INFORMATION

Analog Input

VC++ (6, 7, 8)/VB

BCB/Delphi

Level 1

위험 요소 없음

SYNOPSIS

❒ VT_I4 DX_AdGetDigit ([in]VT_HANDLE hDevice, [in]VT_I4 Channel, [out]VT_PI4 Digit)

DESCRIPTION

이 함수는 주어진 채널에 대하여 A/D 변환을 수행하고 그 값을 정수(Digit) 값으로 반환합니다.

PARAMETER

▶ ***hDevice*** : DX_GnLoadDevice()로부터 얻어온 Device 의 핸들값 입니다.

▶ ***Channel*** : A/D 를 수행 할 채널 번호를 지정합니다. 채널 번호는 0 부터 시작합니다.

▶ ***Digit***: A/D 결과를 정수(Digit) 값으로 반환합니다. 이 값의 범위는 다음과 같습니다.

Device	Resolution	Range (Digit)
COMI-DX10x COMI-DX30x	16 Bit	0 ~ 65535

RETURN VALUE

□ 함수 수행 결과

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

SEE ALSO

DX_AiGetVolt

REFERENCE

	정수(Digit) 값을 전압(Voltage) 값으로 변환 시
	반환하는 정수 값을 Voltage 값으로 변환하려면 다음과 같은 식이 적용되어야 합니다. $$V = \frac{V_{max} - V_{min}}{D_{max} - D_{min}}(D - D_{min}) + V_{min}$$ 여기서 V : 정수값으로부터 환산되는 Voltage 값 D : 환산하고자 하는 대상 정수 값 Vmax : A/D 범위의 최대값 (AiSetRange 함수 참조) Vmin : A/D 범위의 최소값 (AiSetRange 함수 참조) Dmax : 정수 값의 최대 범위 값(A/D 분해능에 따라 다르며 앞의 표 참조) Dmin : 정수 값의 최소 범위 값(A/D 분해능에 따라 다르며 앞의 표 참조)

EXAMPLE

아날로그 입력 0 번 채널의 A/D 값을 정수(Digit) 값으로 받는 예제입니다.

[C / C++]

```
DX_AiSetInputType( hDevice, dxAI_SINGLE ); // Single 모드 입력 설정
DX_AiSetRange( hDevice, 0, dxRANGE_V4 ); // -10 V ~ 10 V 범위 설정

LONG nDigit = 0;
DX_AiGetDigit( hDevice, 0, &nDigit );

printf(“ 0 번 채널 A/D Digit 값 = %d\n”, nDigit);
```

NAME

DX_AiGetVolt

– 대상 A/D 채널의 입력 전압(Voltage) 값 반환

INFORMATION

- Analog Input
- VC++ (6, 7, 8)/VB
- BCB/Delphi
- Level 1
- 위험 요소 없음

SYNOPSIS

❒ VT_I4 DX_AdGetVolt ([in]VT_HANDLE hDevice, [in]VT_I4 Channel, [out]VT_PR8 Volt)

DESCRIPTION

이 함수는 주어진 채널에 대하여 A/D 변환을 수행하고 그 값을 전압(Voltage) 값으로 반환합니다.

PARAMETER

▶ ***hDevice*** : DX_GnLoadDevice()로부터 얻어온 Device 의 핸들값 입니다.

▶ ***Channel*** : A/D 를 수행 할 채널 번호를 지정합니다. 채널 번호는 0 부터 시작합니다.

▶ ***Volt*** : A/D 결과를 전압(Volt) 값으로 반환합니다. 이 값의 범위는 DX_AdSetRange() 함수로 설정한 입력 범위에 맞추어 반환됩니다.

RETURN VALUE

□ 함수 수행 결과

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

SEE ALSO

DX_AiGetDigit

EXAMPLE

아날로그 입력 0 번 채널의 A/D 값을 전압(Voltage) 값으로 받는 예제입니다.

[C / C++]

```
DX_AiSetInputType( hDevice, dxAI_SINGLE ); // Single 모드 입력 설정
DX_AiSetRange( hDevice, 0, dxRANGE_V4 ); // -10 V ~ 10 V 범위 설정

double dVolt  = 0;
DX_AiGetVolt( hDevice, 0, &dVolt );

printf(“ 0 번 채널 A/D Volt 값 = %.2f\n”, dVolt );
```

5.2 A/D Scan

이 단원에서는 A/D Scan 함수들을 소개합니다.

Unlimited A/D Scan 방식은 사용자가 직접 A/D trigger 를 하지 않고, 디바이스에 내장된 하드웨어 타이머가 일정한 주기로 A/D trigger 를 발생시키고 변환된 A/D 데이터를 버퍼에 저장하는 방식입니다. 이 때 Scan 데이터를 저장하는 버퍼는 환형 버퍼 형식으로 운용되며 사용자는 필요 시에 이 버퍼로부터 데이터를 취하게 됩니다. 이 방식은 Single point A/D 방식에 비해 속도가 빠르고 정확한 샘플링 주기를 보장할 수 있습니다. 따라서 이 방식은 고속 A/D 를 할 경우에 매우 유용하게 사용될 수 있습니다.

A/D Scan 함수와 관련된 함수들의 리스트는 다음과 같습니다.

Function	Device
DX_AdScanSetRange	COMI-DX10x, COMI-DX20x
DX_AdScanGetRange	
DX_AdScanSetChannelList	
DX_AdScanSetTriggerMode	
DX_AdScanGetTriggerMode	
DX_AdScanStart	
DX_AdScanFilterStart	
DX_AdScanStop	
DX_AdScanClear	
DX_AdScanGetFreq	
DX_AdScanChangeFreq	
DX_AdScanGetCurCount	
DX_ AdScanIsBufFull	
DX_AdScanResume	
DX_AdScanRetrChannel(I2,F4,F8)	
DX_AdScanRetrBlock(I2,F4,F8)	

DX_AdScanFilterConfig	
DX_AdScanGetMinMax	

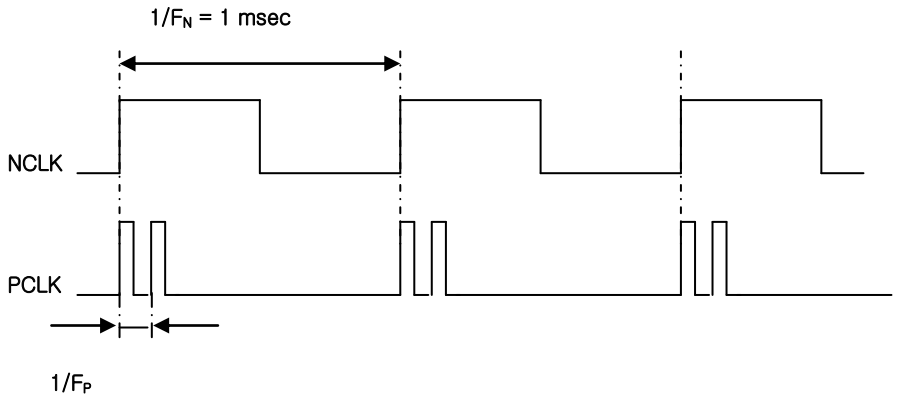
[표 5-2] Analog Input A/D Scan 함수 리스트 및 사용 가능 디바이스

Unlimited A/D Scan 과 고속 DMA A/D Scan 관련 함수들을 소개하기에 앞서, 참고해야 할 사항들을 수록합니다.

□ SCAN 이란 ?

A/D SCAN 이라 함은 지정된 여러 채널을 순차적으로 A/D 변환한다는 의미입니다. 따라서 1 회의 SCAN 은 사용자가 지정한 모든 채널에 대하여 1 회씩 A/D 변환이 완료되었을 경우를 1 회의 SCAN 으로 정의합니다. 따라서 이후에 사용되는 Scan rate(또는 Scan frequency)라는 용어는 Scan 채널로 지정된 각각의 채널에 대하여 1 초당 변환되는 A/D 횟수를 의미합니다. 이는 동일 Scan 내에서의 각 채널간 A/D 변환 주기를 결정해주는 Sampling rate(또는 Sampling frequency)와는 구분 되어야 합니다. External trigger 를 사용하는 경우를 제외하곤 Scan rate 와 Sampling rate 는 디바이스 내부의 타이머에 의하여 제어됩니다.

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



NCLK : 스캔 타이머 신호

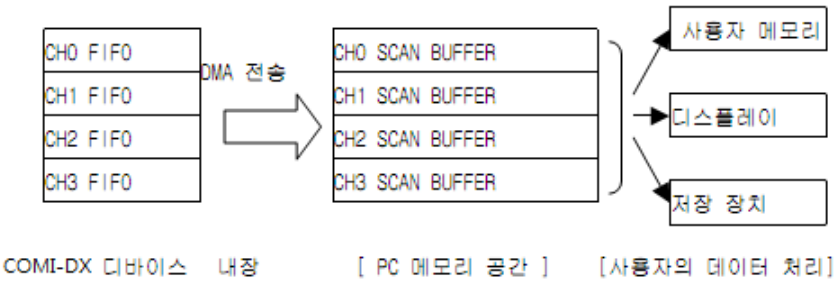
PCLK : 샘플링 타이머 신호, 이 신호가 실제 A/D Trigger 를 한다.

F N : Scan frequency

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

□ 스캔버퍼

드라이버가 자동 할당하여 A/D 된 데이터들을 임시 저장하는 PC 메모리 공간을 말합니다.



[그림 3-1] COMI-DX 시리즈 디바이스를 이용한 A/D 스캔 시에 데이터의 흐름도

□ 환형 버퍼

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

- 현재 N개의 데이터가 스캔된 경우

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

- 현재 N+2개의 데이터가 스캔된 경우

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

□ Frame Scan 과 Continuous Scan

본 라이브러리를 이용하여 고속으로 A/D Scan 을 수행할 때에 AiScanStart()함수의 IsPauseAtFull 매개변수의 값에 따라 다음의 두 가지 방식으로 수행됩니다.

Frame Scan : 일정 크기의 데이터 블록이 스캔 되면 사용자가 AiScanResume()함수를 이용하여 재개하기 전까지 자동으로 스캔을 일시 중지합니다. 이 것은 고속으로 A/D 스캔 하는 경우 데이터의 Overlap 을 방지하기 위한 방식입니다.

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

5.2.1 함수 요약

A/D Scan 과 관련된 함수들의 리스트는 다음과 같습니다.

Summary of Functions
❒ VT_I4 DX_AdScanSetRange ([in]VT_HANDLE hDevice, [in]VT_I4 numChannel, [in]VT_PI4 pChanList, [in]VT_PI4 pRangeList) A/D Scan 을 위한 출력 범위를 설정합니다.
❒ VT_I4 DX_AdScanGetRange ([in]VT_HANDLE hDevice, [in]VT_I4 numChannel, [in]VT_PI4 pChanList, [out]VT_PI4 pRangeList). A/D Scan 을 위한 출력 범위를 반환합니다.
❒ VT_I4 DX_AdScanSetChannelList ([in]VT_HANDLE hDevice, [in]VT_I4 numChannel, [in]VT_PI4 pChanList) A/D Scan 을 위한 채널 리스트를 설정합니다.
❒ VT_I4 DX_AdScanSetTriggerMode ([in]VT_HANDLE hDevice, [in]VT_I4 TrgMode, [in]VT_I4 Inverse) A/D Scan 을 위한 트리거 신호의 입력 방법을 설정합니다.
❒ VT_I4 DX_AdScanGetTriggerMode ([in]VT_HANDLE hDevice, [out]VT_PI4 TrgMode, [out]VT_PI4 Inverse) A/D Scan 을 위한 트리거 신호의 입력 방법을 반환합니다.
❒ VT_I4 DX_AdScanStart ([in]VT_HANDLE hDevice, [in]VT_I4 ScanFreq, [in]VT_I4 SampFreq, [in]VT_I4 nBufSize, [in]VT_I4 nTrsMethod, [in]VT_I4 IsPauseAtFull) A/D Scan 을 시작합니다.
❒ VT_I4 DX_AdScanFilterStart ([in]VT_HANDLE hDevice, [in]VT_I4 ScanFreq, [in]VT_I4 SampFreq, [in]VT_I4 nBufSize) A/D Scan Filter 를 시작합니다.
❒ VT_I4 DX_AdScanStop ([in]VT_HANDLE hDevice, [in]VT_I4 IsReleaseBuf) A/D Scan 을 종료 합니다.
❒ VT_I4 DX_AdScanClear ([in]VT_HANDLE hDevice) A/D Scan 기능을 초기화 합니다.
❒ VT_I4 DX_AdScanGetFreq ([in]VT_HANDLE hDevice, [out]VT_PI4 ScanFreq , [out]VT_PI4 SampFreq) A/D Scan 동작에 설정된 실제 Scan 주파수와 Sample 주파수를 반환합니다.
❒ VT_I4 DX_AdScanChangeFreq ([in]VT_HANDLE hDevice, [in]VT_I4 ScanFreq , [in]VT_I4 SampFreq) A/D Scan 진행 중에 Scan 주파수와 Sample 주파수를 변경합니다.

❒ VT_I4 DX_AdScanGetCurCount ([in]VT_HANDLE hDevice, [out]VT_PI4 dwCount)
현재까지 수행 된 A/D Scan 횟수를 반환합니다.

❒ VT_I4 DX_AdScanIsBufFull ([in]VT_HANDLE hDevice, [out]VT_PI4 IsBufFull)
A/D Scan 버퍼 상태를 반환합니다.

❒ VT_I4 DX_AdScanResume ([in]VT_HANDLE hDevice)
일시 중지된 A/D Scan 을 재 시작 합니다.

❒ VT_I4 DX_AdScanRetrChannelI2 ([in]VT_HANDLE hDevice, [in]VT_I4 ChannelOrder, [in]VT_I4 StartCount, [in]VT_I4 MaxNumData, [in]VT_PI2 DestBuf , [out]VT_PI4 RetrivedDataCount)
A/D Scan 채널 중에서 하나의 채널에 대한 데이터(short 형) 블록을 전달합니다.

❒ VT_I4 DX_AdScanRetrChannelF4 ([in]VT_HANDLE hDevice, [in]VT_I4 ChannelOrder, [in]VT_I4 StartCount, [in]VT_I4 MaxNumData, [in]VT_PR4 DestBuf, [out]VT_PI4 RetrivedDataCount)
A/D Scan 채널 중에서 하나의 채널에 대한 데이터(float 형) 블록을 전달합니다.

❒ VT_I4 DX_AdScanRetrChannelF8 ([in]VT_HANDLE hDevice, [in]VT_I4 StartCount, [in]VT_I4 MaxNumData, VT_PR8 DestBuf, [out]VT_PI4 RetrivedDataCount)
A/D Scan 채널 중에서 하나의 채널에 대한 데이터(double 형) 블록을 전달합니다.

❒ VT_I4 DX_AdScanRetrBlockI2 ([in]VT_HANDLE hDevice, [in]VT_I4 StartCount, [in]VT_I4 MaxNumData, [in]VT_PI2 DestBuf , [out]VT_PI4 RetrivedDataCount)
A/D Scan 전 채널에 대한 데이터(short 형)를 사용자가 지정하는 버퍼에 전달합니다.

❒ VT_I4 DX_AdScanRetrBlockF4 ([in]VT_HANDLE hDevice, [in]VT_I4 StartCount, [in]VT_I4 MaxNumData, [in]VT_PR4 DestBuf, [out]VT_PI4 RetrivedDataCount)
A/D Scan 전 채널에 대한 데이터(float 형)를 사용자가 지정하는 버퍼에 전달합니다.

❒ VT_I4 DX_AdScanRetrBlockF8 ([in]VT_HANDLE hDevice, [in]VT_I4 StartCount, [in]VT_I4 MaxNumData, [in]VT_PR8 DestBuf, [out]VT_PI4 RetrivedDataCount)
A/D Scan 전 채널에 대한 데이터(double 형)를 사용자가 지정하는 버퍼에 전달합니다.

❒ VT_I4 DX_AdScanFilterConfig ([in]VT_HANDLE hDevice, [in]VT_I4 FilterType, [in]VT_R8 CutOffFrequency, [in]VT_I4 AvgCount)
A/D Scan Filter 기능에 대한 상세를 설정합니다.

❒ VT_I4 DX_AdScanGetMinMax ([in]VT_HANDLE hDevice, [in]VT_I4 ChannelOrder, [out]VT_PR8 MinVolt, [out]VT_PR8 MaxVolt)
A/D Scan Filter 진행 중 최소값(Volt)과 최대값 (Volt)를 반환합니다.

5.2.2 함수 설명

NAME

DX_AdScanSetRange / DX_AdScanGetRange

- A/D Scan 출력 범위 설정 / 반환

INFORMATION

- Analog Input
- VC++ (6, 7, 8)/VB
- BCB/Delphi
- Level 1
- 위험 요소 없음

SYNOPSIS

❒ VT_I4 DX_AdScanSetRange ([in]VT_HANDLE hDevice, [in]VT_I4 numChannel, [in]VT_PI4 pChanList, [in]VT_PI4 pRangeList)

❒ VT_I4 DX_AdScanGetRange ([in]VT_HANDLE hDevice, [in]VT_I4 numChannel, [in]VT_PI4 pChanList, [out]VT_PI4 pRangeList)

DESCRIPTION

대상 디바이스의 A/D scan 출력 채널에 대하여 출력 범위를 설정/ 반환 합니다.

PARAMETER

▶ ***hDevice*** : DX_GnLoadDevice()로부터 얻어온 Device 의 핸들값 입니다.

▶ ***NumChannel*** : A/D Scan 할 채널 수 입니다.

▶ ***ChanList*** : A/D Scan 을 수행할 채널 리스트를 담고 있는 배열 또는 포인터 입니다.

▶ ***pRangeList*** : A/D Scan 을 수행할 채널 리스트에 대하여 설정/ 반환 할 출력 범위를 담고 잇는 배열 또는 포인터 입니다. 각 출력 범위의 값은 다음과 같습니다.

Value	Meaning
0 (dxRANGE_V0)	0v ~ 5v
1 (dxRANGE_V1)	0v ~ 10v
2 (dxRANGE_V2)	0v ~ 10.8v
3 (dxRANGE_V3)	-5v ~ 5v

4 (dxRANGE_V4)	-10v ~ 10v
5 (dxRANGE_V5)	-10.8v ~ 10.8v

RETURN VALUE

□ 함수 수행 결과

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

REFERENCE

□ DX_AiSetRange / DX_AiGetRange 와 호환되지 않습니다. A/D Scan 기능을 사용하기 위하여 반드시 DX_AdScanSetRange 함수를 통하여 출력 범위를 설정해주어야 하며 기본 값은 0 (0 ~ 5 v) 입니다.

NAME

DX_AdScanSetChannelList

- A/D Scan 출력 채널 리스트 설정

INFORMATION

Analog Input

VC++ (6, 7, 8)/VB

BCB/Delphi

Level 1

위험 요소 없음

SYNOPSIS

❒ VT_I4 DX_AdScanSetChannelList ([in]VT_HANDLE hDevice, [in]VT_I4 numChannel, [in]VT_PI4 pChanList)

DESCRIPTION

대상 디바이스의 A/D scan 을 수행하기 위한 출력 채널 리스트를 설정합니다.

PARAMETER

▶ ***hDevice*** : DX_GnLoadDevice()로부터 얻어온 Device 의 핸들값 입니다.

▶ ***NumChannel*** : A/D Scan 할 채널 수 입니다.

▶ ***ChanList*** : A/D Scan 을 수행할 채널 리스트를 담고 있는 배열 또는 포인터 입니다.

RETURN VALUE

□ 함수 수행 결과

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

NAME

DX_AdScanSetTriggerMode /

DX_AdScanGetTriggerMode

- A/D 신호의 종류와 운용 방법을 설정 / 반환

INFORMATION

Analog Input

VC++ (6, 7, 8)/VB

BCB/Delphi

Level 1

위험 요소 없음

SYNOPSIS

❒ VT_I4 DX_AdScanSetTriggerMode ([in]VT_HANDLE hDevice, [in]VT_I4 TrgMode, [in]VT_I4 Inverse)

❒ VT_I4 DX_AdScanGetTriggerMode ([in]VT_HANDLE hDevice, [out]VT_PI4 TrgMode, [out]VT_PI4 Inverse)

DESCRIPTION

이 함수는 Trigger 신호의 입력 방법을 설정 / 반환 합니다.

PARAMETER

▶ ***hDevice*** : DX_GnLoadDevice()로부터 얻어온 Device 의 핸들값 입니다.

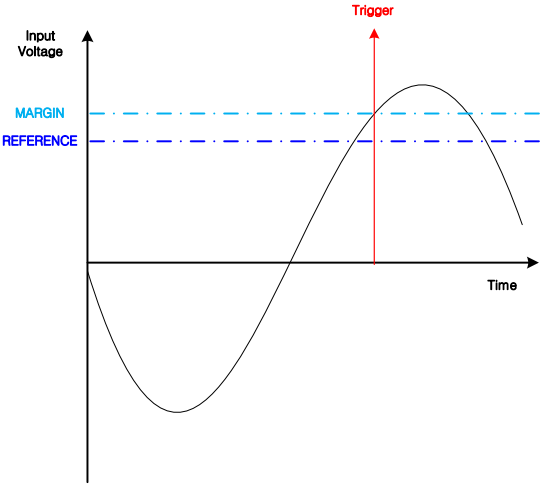
▶ ***InputSource*** : Trigger 입력 방식을 설정합니다.

Value	Meaning
0 (dxTRG_S)	내부 트리거 입력에 따라 A/D Scan 을 수행합니다.
1 (dxTRG_E)	외부 트리거 입력에 따라 A/D Scan 을 수행합니다. 이 때의 트리거는 A/D 스캔을 시작하는 트리거로서 동작하는 것이 아니라, 트리거입력이 들어올 때마다 각각의 경우에 한번씩 A/D 가 수행되도록 설계되어 있습니다.
2 (dxTRG_E_SCAN)	외부 트리거 입력에 따라 A/D 를 수행합니다. 이 때의 트리거는 A/D 스캔을 시작하는 트리거로서 동작하는 것이 아니라, 트리거입력이 들어올 때마다 설정된 A/D 출력 채널수 만큼 A/D 가 수행되도록 설계되어 있습니다.

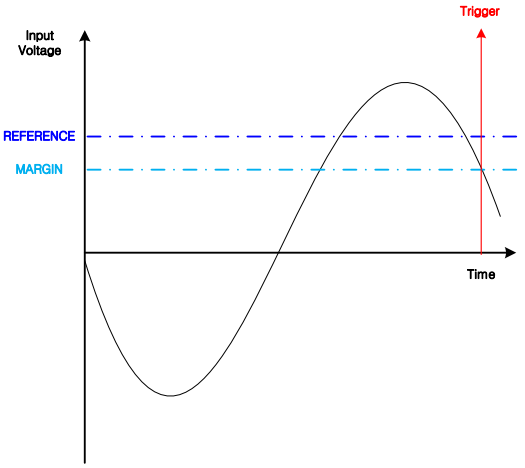
▶ ***Inverse*** : 트리거를 입력 받을 전압의 Edge 상태를 설정합니다.

Value	Meaning
0 (dxINV_FALLING)	Falling Edge.

1 (dxINV_RISING)	Rising Edge.



[그림 1] Rising Edge 설정 시



[그림 2] Falling Edge 설정 시

RETURN VALUE

□ 함수 수행 결과

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

NAME

DX_AdScanStart

- A/D Scan 기능 시작.

INFORMATION

- Analog Input
- VC++ (6, 7, 8)/VB
- BCB/Delphi
- Level 1
- 위험 요소 없음

SYNOPSIS

❒ VT_I4 DX_AdScanStart ([in]VT_HANDLE hDevice, [in]VT_I4 ScanFreq, [in]VT_I4 SampFreq, [in]VT_I4 nBufSize, [in]VT_I4 nTrsMethod, [in]VT_I4 IsPauseAtFull)

DESCRIPTION

이 함수는 A/D Scan 기능을 시작합니다.

PARAMETER

▶ ***hDevice*** : DX_GnLoadDevice()로부터 얻어온 Device 의 핸들값 입니다.

▶ ***ScanFreq*** : A/D scan frequency 를 Hz 단위로 지정합니다. 이 값은 SCAN 과 SCAN 사이의 시간차를 결정합니다.

▶ ***SampFreq*** : A/D sample frequency 를 Hz 단위로 지정합니다. 이 값은 SCAN 주기안의 Sample 간의 시간차를 결정합니다.

▶ ***nBufSize*** : 스캔 데이터를 저장할 환형버퍼의 크기를 결정하는 값으로써 각 채널의 데이터가 환형버퍼에 오버랩(Overlap)되지 않고 담길 수 있는 최대 데이터 수를 의미합니다.

환형 버퍼는 디바이스 드라이버에서 자동으로 할당하며 실제 크기 안내.

환형 버퍼는 디바이스 드라이버에서 자동으로 할당하며 실제 크기는 다음과 같습니다.

환형 버퍼의 실제 크기(bytes) = nNumChannel * nBufSize * sizeof(short)

▶ ***nTrsMethod*** : A/D 디바이스에서 스캔버퍼로 데이터를 전송하는 방식을 지정합니다. 이 값은 다음의 두 값 중의 하나이어야 합니다.

Value	Meaning
0 (dxTRS_SINGLE)	1 회의 SCAN 이 완료될 때마다 인터럽트를 발생시켜 데이터를 전송합니다. 인터럽트의 한계에 따라 Scan frequency 가 30 KHz 이상이 되면 이 방식이 적절히 작동하지 않을 수 있습니다.

1 (dxTRS_BLOCK)	A/D 디바이스는 A/D Conversion 데이터를 디바이스에 내장되어 있는 FIFO 메모리에 일단 저장하고, 2048 개의 데이터가 쌓이면 인터럽트를 발생시켜 데이터를 2048 개 단위로 사용자 버퍼에 전송합니다. 이 방식은 고속 A/D scan 시에 적합합니다.

▶ ***IsPauseAtFull*** : 이 값은 스캔 버퍼에 데이터가 꽉 찬 경우에 스캔을 일시 중지할 것인지를 결정합니다.
이 값을 1(dxTRUE)로 하면 Frame Scan, 0(dxFALSE)으로 하면 Continuous Scan 방식으로 운용됩니다. 자세한 사항은 단원 앞 부분의 “Frame Scan 과 Continuous Scan” 설명을 참조하시기 바랍니다.

RETURN VALUE

□ 함수 수행 결과

Value	Meaning
음수	수행 실패. 자세한 내용은 ‘에러 처리’ 편을 참고합니다.
0 (dxERR_NONE)	수행 성공.

SEE ALSO

DX_AdScanStop

REFERENCE

□ ScanFreq 의 설정 가능 범위는 사용하는 디바이스의 종류에 따라 달라 집니다. Analog Input Board 들은 Scan 을 수행할 채널의 개수에 따라 설정 가능한 Scan Frequency 가 달라 집니다. 예를 들어 COMI-DX101 보드를 사용해서 5 채널에 대해 AiScan 을 하면, COMI-LX101 의 Maximum Sampling Frequency 가 500kHz 이므로 채널 별로 적용 가능한 Frequency 는 100kHz (500kHz / 5 = 100kHz) 가 됩니다.

□ DX_AiSetInputType() 함수를 이용하여 Single / Diffrential 타입을 설정 해주어야 합니다.

EXAMPLE

□ 한 채널을 1kHz 로 SCAN 할 때의 함수 예제입니다.
(TRS_SINGLE 모드 사용)

[C / C++]

```
LONG nChan = 0;
LONG nRange = dxRANGE_V4;


DX_AiSetInputType ( hDevice, dxAI_SINGLE );
DX_AdScanSetTriggerMode ( hDevice, dxTRG_S, dxINV_FALLING );
```

```
DX_AdScanSetChannelList ( hDevice, 1,& nChan);
DX_AdScanSetRange ( hDevice, 1, & nChan, &nRange);

DX_AdScanStart ( hDevice, 1000, 4000, 10240, dxTRS_SINGLE, dxFALSE);
```

□ 32 채널 모두에 대하여 채널당 10kHz 로 SCAN 할 때의 함수 예제입니다.
(TRS_BLOCK 모드 사용).

[C / C++]

```
LONG nChanList[32] ={0,};
LONG nRangeList[32] = {0, };

for( int i=0; i< 32; i++ ) {
        nChanList[i] = i;
        nRangeList[i] = dxRANGE_V4;
}

DX_AiSetInputType ( hDevice, dxAI_SINGLE );
DX_AdScanSetTriggerMode ( hDevice, dxTRG_S, dxINV_FALLING );

DX_AdScanSetChannelList ( hDevice, 32, nChanList);
DX_AdScanSetRange ( hDevice, 32, nChanList, nRangeList);

DX_AdScanStart ( hDevice, 10000, 40000, 10240, dxTRS_BLOCK, dxFALSE);
```

NAME

DX_AdScanFilterStart

- A/D Scan Filter 기능 시작.

INFORMATION
Analog Input
VC++ (6, 7, 8)/VB
BCB/Delphi
Level 1
위험 요소 없음

SYNOPSIS

❒ VT_I4 DX_AdScanFilterStart ([in]VT_HANDLE hDevice, [in]VT_I4 ScanFreq, [in]VT_I4 SampFreq, [in]VT_I4 nBufSize)

DESCRIPTION

이 함수는 A/D Scan Filter 기능을 시작합니다.
A/D Scan Filter 는 기존 A/D Scan 방식과 달리 디바이스의 내부 CPU 에 의하여 A/D 입력 신호가 Filter 처리 됩니다.
고속 A/D Scan 방식을 원하는 경우에는 사용하지 마십시오.

	사용 시 주의 사항
	- 이 함수는 디바이스의 내부 CPU 를 사용 하므로 DX_AdScanFilterConfig() 함수를 통하여 Filter 환경 설정을 먼저 구성하여야 합니다.

PARAMETER

▶ ***hDevice*** : DX_GnLoadDevice()로부터 얻어온 Device 의 핸들값 입니다.

▶ ***ScanFreq*** : A/D scan frequency 를 Hz 단위로 지정합니다. 이 값은 SCAN 과 SCAN 사이의 시간차를 결정합니다.

▶ ***SampFreq*** : A/D sample frequency 를 Hz 단위로 지정합니다. 이 값은 SCAN 주기안의 Sample 간의 시간차를 결정합니다.

▶ ***nBufSize*** : 스캔 데이터를 저장할 환형버퍼의 크기를 결정하는 값으로써 각 채널의 데이터가 환형 버퍼에 오버랩(Overlap)되지 않고 담길 수 있는 최대 데이터 수를 의미합니다.

환형 버퍼는 디바이스 드라이버에서 자동으로 할당하며 실제 크기 안내.

환형 버퍼는 디바이스 드라이버에서 자동으로 할당하며 실제 크기는 다음과 같습니다.

환형 버퍼의 실제 크기(bytes) = nNumChannel * nBufSize * sizeof(short)

RETURN VALUE

□ 함수 수행 결과

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

SEE ALSO

DX_AdScanFilterConfig, DX_AdScanGetMinMax, DX_AdScanStop

EXAMPLE

□ 한 채널을 1kHz 로 SCAN 할 때의 함수 예제입니다.

(Moving Average 필터 사용)

[C / C++]

```
LONG nChan = 0;
LONG nRange = dxRANGE_V4;


DX_AiSetInputType ( hDevice, dxAI_SINGLE );
DX_AdScanSetTriggerMode ( hDevice, dxTRG_S, dxINV_FALLING );

DX_AdScanSetChannelList ( hDevice, 1,& nChan);
DX_AdScanSetRange ( hDevice, 1, & nChan, &nRange);

/*************************************************************/
/* 중요                                                      */
/* 10 개의 데이터의 평균 입력 값을 스캔 받음                 */
/*************************************************************/
DX_AdScanFilterConfig( hDevice, dxAIF_AVG, 0, 10 );

DX_AdScanFilterStart ( hDevice, 1000, 4000, 10240);
```

□ 32 채널을 10kHz 로 SCAN 할 때의 함수 예제입니다.

(LowPass Filter 필터 사용)

[C / C++]

```
LONG nChanList[32] ={0,};
LONG nRangeList[32] = {0, };

for( int i=0; i< 32; i++ ) {
        nChanList[i] = i;
        nRangeList[i] = dxRANGE_V4;
}

DX_AiSetInputType ( hDevice,  dxAI_SINGLE );
DX_AdScanSetTriggerMode ( hDevice, dxTRG_S, dxINV_FALLING );

DX_AdScanSetChannelList ( hDevice, 32, nChanList);
DX_AdScanSetRange ( hDevice, 32, nChanList, nRangeList);

/*************************************************************/
/* 중요                                                      */
/* 10KHz 이상의 값을 필터 처리                               */
/*************************************************************/
DX_AdScanFilterConfig( hDevice, dxAIF_LPF,  10000, 0 );

DX_AdScanFilterStart ( hDevice, 10000, 4000, 10240);
```

NAME

DX_AdScanStop

– A/D Scan 종료

INFORMATION

Analog Input

VC++ (6, 7, 8)/VB

BCB/Delphi

Level 1

위험 요소 없음

SYNOPSIS

❒ VT_I4 DX_AdScanStop ([in]VT_HANDLE hDevice, [in]VT_I4 IsReleaseBuf)

DESCRIPTION

이 함수는 A/D scan 을 종료합니다.

PARAMETER

▶ ***hDevice*** : DX_GnLoadDevice()로부터 얻어온 Device 의 핸들값 입니다.

▶ ***IsReleaseBuf*** : AiScanStart 함수가 수행될 때 할당되었던 스캔 버퍼를 메모리 해제시킬 것인지를 결정합니다.
만일 이 값을 0(dxFALSE)로 지정하면 후에 반드시 DX_AdScanClear() 를 사용하여 버퍼를 해제 하여야 합니다. 이 값을 1(dxTRUE)로 지정하면 DX_AdScanClear () 함수를 수행할 필요가 없습니다 .

RETURN VALUE

□ 함수 수행 결과

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

SEE ALSO

DX_AdScanStart

NAME

DX_AdScanClear

– A/D Scan 초기화

INFORMATION

Analog Input

VC++ (6, 7, 8)/VB

BCB/Delphi

Level 1

위험 요소 없음

SYNOPSIS

❒ VT_I4 DX_AdScanClear ([in]VT_HANDLE hDevice)

DESCRIPTION

대상 디바이스에 대하여 A/D scan 기능을 초기화합니다.

PARAMETER

▶ ***hDevice*** : DX_GnLoadDevice()로부터 얻어온 Device 의 핸들값 입니다.

RETURN VALUE

□ 함수 수행 결과

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

SEE ALSO

DX_AdScanStart

NAME

DX_AdScanGetFreq

- 실제 적용된 A/D Scan 주파수 반환

INFORMATION

Analog Input

VC++ (6, 7, 8)/VB

BCB/Delphi

Level 1

위험 요소 없음

SYNOPSIS

❒ VT_I4 DX_AdScanGetFreq ([in]VT_HANDLE hDevice, [out]VT_PI4 ScanFreq , [out]VT_PI4 SampFreq)

DESCRIPTION

이 함수는 실제 스캔에 적용된 스캔 주파수와 샘플 주파수를 반환합니다.

PARAMETER

▶ ***hDevice*** : DX_GnLoadDevice()로부터 얻어온 Device 의 핸들값 입니다.

▶ ***ScanFreq*** : 반환된 실제 적용된 A/D Scan 주파수 입니다. 단위는 Hz 입니다. 이 값은 DX_AdScanStart() 함수에서 설정한 주파수와 차이가 있습니다.

▶ ***SampFreq*** : 반환된 실제 적용된 A/D Sample 주파수 입니다. 단위는 Hz 입니다. 이 값은 DX_AdScanStart() 함수에서 설정한 주파수와 차이가 있습니다.

RETURN VALUE

□ 함수 수행 결과

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

NAME

DX_AdScanChangeFreq

- A/D Scan 주파수 설정

INFORMATION
Analog Input
VC++ (6, 7, 8)/VB
BCB/Delphi
Level 1
위험 요소 없음

SYNOPSIS

❒ VT_I4 DX_AdScanChangeFreq ([in]VT_HANDLE hDevice, [in]VT_I4 ScanFreq , [in]VT_I4 SampFreq)

DESCRIPTION

이 함수는 스캔이 진행되는 중에 스캔 주파수와 샘플 주파수를 변경합니다.

PARAMETER

▶ ***hDevice*** : DX_GnLoadDevice()로부터 얻어온 Device 의 핸들값 입니다.

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

▶ ***SampFreq*** : 변경할 A/D Sample 주파수를 설정합니다. 단위는 Hz 입니다.

RETURN VALUE

□ 함수 수행 결과

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

NAME

DX_AdScanGetCurCount

- 현재까지 수행된 A/D Scan 횟수 반환

INFORMATION

- Analog Input
- VC++ (6, 7, 8)/VB
- BCB/Delphi
- Level 1
- 위험 요소 없음

SYNOPSIS

❒ VT_I4 DX_AdScanGetCurCount ([in]VT_HANDLE hDevice, [out]VT_PI4 dwCount)

DESCRIPTION

이 함수는 현재까지 수행된 SCAN 횟수를 반환합니다.
사용자는 버퍼에서 데이터를 취할 때 이 함수를 참조하여 가장 최근 스캔된 데이터의 위치를 알아낼 수 있습니다.

PARAMETER

▶ ***hDevice*** : DX_GnLoadDevice()로부터 얻어온 Device 의 핸들값 입니다.

▶ ***dwCount*** : 현재까지 수행된 SCAN 횟수 입니다.

RETURN VALUE

□ 함수 수행 결과

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

NAME

DX_AdScanIsBufFull

- A/D Scan 버퍼 상태 반환

INFORMATION
Analog Input
VC++ (6, 7, 8)/VB
BCB/Delphi
Level 1
위험 요소 없음

SYNOPSIS

❒ VT_I4 DX_AdScanIsBufFull ([in]VT_HANDLE hDevice, [out]VT_PI4 IsBufFull)

DESCRIPTION

이 함수는 지정한 크기(개수)의 스캔 버퍼에 데이터가 다 찼는지를 확인하는 함수입니다.

Frame Scan 방식에서 사용합니다.

Frame Scan 방식으로 A/D 수행 시 DX_AdScanBufFull() 함수를 통해 스캔 버퍼 상태를 확인하여, 데이터가 다 차게 되면 스캔 데이터를 처리한 후 DX_AdScanResume() 함수를 이용하여 일시 중지된 스캔을 재시작 합니다. Frame Scan 방식은 DX_AdScanStart() 함수에서 IsPauseAtFull 매개변수를 통하여 설정합니다.

PARAMETER

▶ ***hDevice*** : DX_GnLoadDevice()로부터 얻어온 Device 의 핸들값 입니다.

▶ ***IsBufFull*** **:** A/D Scan 버퍼 상태를 반환합니다.

RETURN VALUE

□ 함수 수행 결과

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

SEE ALSO

DX_AdScanStart, DX_AdScanResume

EXAMPLE

[C / C++]

```
// DX_AdScanStart 예제 참조. ( IsPauseBufFull 플래그 TRUE 설정 후 )

LONG IsFull = 0;
LONG nCurCnt = 0;
DOUBLE dBuf[10240];

while( Is_Stop () ) .// Is_Stop() 은 가상의 함수임.
{
        DX_AdScanIsBufFull( hDevice, &IsFull );
        If( IsFull )
        {
                // 버퍼 Full, Ad Scan 은 자동으로 정지 된다.
                DX_AdScanRetrChannelF8( hDevice, CH_0, 1, 10240, dBuf, &nCurCnt);

                // Data Process…

                DX_AdScanResume(hDevice);
        }
}

DX_AdScanStop( hDevice, dxFALSE );
DX_AdScanClear( hDevice );
```

NAME

DX_AdScanResume

- 일시 중지된 A/D Scan 재 시작

INFORMATION

- Analog Input
- VC++ (6, 7, 8)/VB
- BCB/Delphi
- Level 1
- 위험 요소 없음

SYNOPSIS

❒ VT_I4 DX_AdScanResume ([in]VT_HANDLE hDevice)

DESCRIPTION

이 함수는 일시 중지된 A/D 스캔을 재 시작 합니다.

안내	Frame Scan 방식에서 사용합니다.
	Frame Scan 방식으로 A/D 수행 시 스캔 버퍼에 데이터가 다 차게 되면 스캔이 일시 중지 됩니다. 이 때 사용자는 필요에 따라 스캔 데이터를 처리하고 DX_AdScanResume() 함수를 이용하여 일시 정지된 스캔을 재시자 할 수 있습니다.

PARAMETER

▶ ***hDevice*** : DX_GnLoadDevice()로부터 얻어온 Device 의 핸들값 입니다.

RETURN VALUE

□ 함수 수행 결과

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

SEE ALSO

DX_AdScanStart, DX_AdIsBufFull

EXAMPLE

[C / C++]

```
// DX_AdScanStart 예제 참조. ( IsPauseBufFull 플래그 TRUE 설정 후 )

LONG IsFull = 0;
LONG nCurCnt = 0;
DOUBLE dBuf[10240];

while( Is_Stop () ) .// Is_Stop() 은 가상의 함수임.
{
        DX_AdScanIsBufFull( hDevice, &IsFull );
        If( IsFull )
        {
                // 버퍼 Full, Ad Scan 은 자동으로 정지 된다.
                DX_AdScanRetrChannelF8( hDevice, CH_0, 1, 10240, dBuf, &nCurCnt);

                // Data Process…

                DX_AdScanResume(hDevice);
        }
}

DX_AdScanStop( hDevice, dxFALSE );
DX_AdScanClear( hDevice );
```

NAME

DX_AdScanRetrChannelI2

- A/D Scan 채널 중 하나의 채널에 대한 데이터(short 형) 블록 반환

INFORMATION

Analog Input

VC++ (6, 7, 8)/VB

BCB/Delphi

Level 1

위험 요소 없음

SYNOPSIS

❒ VT_I4 DX_AdScanRetrChannelI2 ([in]VT_HANDLE hDevice, [in]VT_I4 ChannelOrder, [in]VT_I4 StartCount, [in]VT_I4 MaxNumData, [in]VT_PI2 DestBuf , [out]VT_PI4 RetrivedDataCount)

DESCRIPTION

이 함수는 A/D Scan 채널 중에서 하나의 채널에 대한 데이터 블록을 반환합니다.

데이터 블록은 사용자가 지정한 StartCount 에서부터 MaxNumData 에서 지정한 수만큼이 됩니다. 데이터는 short 형으로써, 전달되는 값은 Voltage 로 환산되기 이전의 정수형 값입니다.

PARAMETER

▶ ***hDevice*** : DX_GnLoadDevice()로부터 얻어온 Device 의 핸들값 입니다.

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

▶ ***ScanCount*** : 원하는 데이터의 Scan Count.

▶ ***MaxNumData*** : 전달 받고자 하는 데이터 블록의 크기(데이터 수)를 지정 합니다.

Value	Meaning
양수	StartCount 부터 이후에 스캔된 데이터 중 MaxNumData 에서 지정한 수만큼 데이터를 전달합니다.
음수	StartCount 부터 이전에 스캔된 데이터 중 MaxNumData 에서 지정한 수만큼 데이터를 전달합니다.

▶ ***DestBuf*** : 데이터를 전달 받을 버퍼 포인터를 지정합니다. 이 버퍼의 크기는 MaxNumData 에서 지정한 값보다 크거나 같아야 합니다.

▶ ***RetrivedDataCount*** : 실제 전달된 데이터 수. 만일 StartCount 이후에 현재까지 스캔된 데이터 수가 MaxNumData 에서 지정한 수 보다 작으면, 현재 스캔된 데이터까지만 전달하게 됩니다.

RETURN VALUE

□ 함수 수행 결과

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

SEE ALSO

DX_AdScanRetrChannelF4, DX_AdScanRetrChannelF8

EXAMPLE

DX_AdRetrChannelF8 예제 참조.

NAME

DX_AdScanRetrChannelF4

- A/D Scan 채널 중 하나의 채널에 대한 데이터(float 형) 블록 반환

INFORMATION

Analog Input

VC++ (6, 7, 8)/VB

BCB/Delphi

Level 1

위험 요소 없음

SYNOPSIS

❒ VT_I4 DX_AdScanRetrChannelF4 ([in]VT_HANDLE hDevice, [in]VT_I4 ChannelOrder, [in]VT_I4 StartCount, [in]VT_I4 MaxNumData, [in]VT_PR4 DestBuf, [out]VT_PI4 RetrivedDataCount)

DESCRIPTION

이 함수는 A/D Scan 채널 중에서 하나의 채널에 대한 데이터 블록을 반환합니다.

데이터 블록은 사용자가 지정한 StartCount 에서부터 MaxNumData 에서 지정한 수만큼이 됩니다. 데이터는 float 형으로써, 전달되는 값은 Voltage 로 환산되기 이전의 정수형 값입니다.

PARAMETER

▶ ***hDevice*** : DX_GnLoadDevice()로부터 얻어온 Device 의 핸들값 입니다.

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

▶ ***ScanCount*** : 원하는 데이터의 Scan Count.

▶ ***MaxNumData*** : 전달 받고자 하는 데이터 블록의 크기(데이터 수)를 지정 합니다.

Value	Meaning
양수	StartCount 부터 이후에 스캔된 데이터 중 MaxNumData 에서 지정한 수만큼 데이터를 전달합니다.
음수	StartCount 부터 이전에 스캔된 데이터 중 MaxNumData 에서 지정한 수만큼 데이터를 전달합니다.

▶ ***DestBuf*** : 데이터를 전달 받을 버퍼 포인터를 지정합니다. 이 버퍼의 크기는 MaxNumData 에서 지정한 값보다 크거나 같아야 합니다.

▶ ***RetrivedDataCount*** : 실제 전달된 데이터 수. 만일 StartCount 이후에 현재까지 스캔된 데이터 수가 MaxNumData 에서 지정한 수 보다 작으면, 현재 스캔된 데이터까지만 전달하게 됩니다.

RETURN VALUE

□ 함수 수행 결과

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

SEE ALSO

DX_AdScanRetrChannelI2, DX_AdScanRetrChannelF8

EXAMPLE

DX_AdRetrChannelF8 예제 참조.

NAME

DX_AdScanRetrChannelF8

- A/D Scan 채널 중 하나의 채널에 대한 데이터(double 형) 블록 반환

INFORMATION

Analog Input

VC++ (6, 7, 8)/VB

BCB/Delphi

Level 1

위험 요소 없음

SYNOPSIS

❒ VT_I4 DX_AdScanRetrChannelF8 ([in]VT_HANDLE hDevice, [in]VT_I4 StartCount, [in]VT_I4 MaxNumData, VT_PR8 DestBuf, [out]VT_PI4 RetrivedDataCount)

DESCRIPTION

이 함수는 A/D Scan 채널 중에서 하나의 채널에 대한 데이터 블록을 반환합니다.

데이터 블록은 사용자가 지정한 StartCount 에서부터 MaxNumData 에서 지정한 수만큼이 됩니다. 데이터는 double 형으로써, 전달되는 값은 Voltage 로 환산되기 이전의 정수형 값입니다.

PARAMETER

▶ ***hDevice*** : DX_GnLoadDevice()로부터 얻어온 Device 의 핸들값 입니다.

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

▶ ***ScanCount*** : 원하는 데이터의 Scan Count.

▶ ***MaxNumData*** : 전달 받고자 하는 데이터 블록의 크기(데이터 수)를 지정 합니다.

Value	Meaning
양수	StartCount 부터 이후에 스캔된 데이터 중 MaxNumData 에서 지정한 수만큼 데이터를 전달합니다.
음수	StartCount 부터 이전에 스캔된 데이터 중 MaxNumData 에서 지정한 수만큼 데이터를 전달합니다.

▶ ***DestBuf*** : 데이터를 전달 받을 버퍼 포인터를 지정합니다. 이 버퍼의 크기는 MaxNumData 에서 지정한 값보다 크거나 같아야 합니다.

▶ ***RetrivedDataCount*** : 실제 전달된 데이터 수. 만일 StartCount 이후에 현재까지 스캔된 데이터 수가 MaxNumData 에서 지정한 수 보다 작으면, 현재 스캔된 데이터까지만 전달하게 됩니다.

RETURN VALUE

□ 함수 수행 결과

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

SEE ALSO

DX_AdScanRetrChannelI2, DX_AdScanRetrChannelF4

EXAMPLE

□ Frame Scan 방식으로 A/D CH0 와 CH1 두 채널의 데이터를 취하는 예제입니다.

[C / C++]

```
#define NUM_CH                  2
#define BUF_SIZE                10240

LONG nChanList[NUM_CH] = {0, 1};
LONG nRangeList[NUM_CH] = {dxRANGE_V4, dxRANGE_V4};
LONG IsFull = 0;
LONG nCurCnt = 0;
DOUBLE dBuf[BUF_SIZE];

DX_AiSetInputType ( hDevice, dxAI_SINGLE );
DX_AdScanSetTriggerMode ( hDevice, dxTRG_S, dxINV_FALLING );

DX_AdScanSetChannelList ( hDevice, NUM_CH,&nChanList);
DX_AdScanSetRange ( hDevice, NUM_CH, & nChanList, &nRangeList);

DX_AdScanStart ( hDevice, 1000, 4000, BUF_SIZE, dxTRS_SINGLE, dxTRUE);

while( Is_Stop () ) .// Is_Stop() 은 가상의 함수임.
{
        DX_AdScanIsBufFull( hDevice, &IsFull );
        If( IsFull )
        {
                for(  int i=0; i< NUM_CH; i++ ) {
                        // 버퍼 Full, Ad Scan 은 자동으로 정지 된다.
                        DX_AdS canRetrChannelF8( hDevice, i, 1, BUF_SIZE, dBuf, &nCurCnt);

                        // Data Process…
                }

                DX_AdScanResume(hDevice);
        }
}

DX_AdScanStop(hDevice);
```

□ Continuous Scan 방식으로 A/D CH0 와 CH1 두 채널의 데이터를 취하는 예제입니다.

[C / C++]

```
#define NUM_CH                  2
#define BUF_SIZE                10240

 LONG nChanList[NUM_CH] = {0, 1};
LONG nRangeList[NUM_CH] = {dxRANGE_V4, dxRANGE_V4};
LONG IsFull = 0;
LONG nCurCnt = 0;
DOUBLE dBuf[BUF_SIZE];

DX_AiSetInputType ( hDevice, dxAI_SINGLE );
DX_AdScanSetTriggerMode ( hDevice, dxTRG_S, dxINV_FALLING );

DX_AdScanSetChannelList ( hDevice, NUM_CH,&nChanList);
DX_AdScanSetRange ( hDevice, NUM_CH, & nChanList, &nRangeList);

DX_AdScanStart ( hDevice, 1000, 4000, BUF_SIZE, dxTRS_SINGLE, dxFALSE);

 while( Is_Stop () ) .// Is_Stop() 은 가상의 함수임.
{
        DX_AdScanGetCurCount(hDevice, & CurCount);

        If( CurCount > PreCount) // 새로운 스캔 데이터가 있으면 처리
        {
                for(  int i=0; i< NUM_CH; i++ ) {

                        DX_AdS canRetrChannelF8( hDevice, i, 1, BUF_SIZE, dBuf, &nCurCnt);

                        // Data Process…
                }
                PreCount += CurCount;
        }
}

DX_AdScanStop(hDevice);
```

NAME

DX_AdScanRetrBlockI2

- A/D Scan 전 채널에 대한 데이터(short 형)를 사용자 지정 버퍼에 반환

INFORMATION

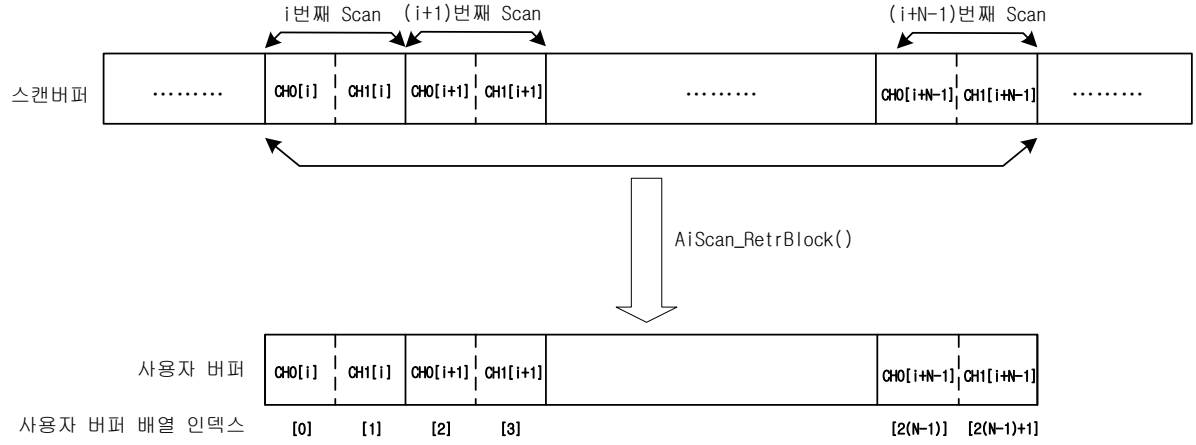
- Analog Input
- VC++ (6, 7, 8)/VB
- BCB/Delphi
- Level 1
- 위험 요소 없음

SYNOPSIS

❒ VT_I4 DX_AdScanRetrBlockI2 ([in]VT_HANDLE hDevice, [in]VT_I4 StartCount, [in]VT_I4 MaxNumData, [in]VT_PI2 DestBuf , [out]VT_PI4 RetrivedDataCount)

DESCRIPTION

이 함수는 A/D Scan 전 채널에 대한 데이터를 사용자가 지정하는 버퍼에 전달합니다. 전달되는 데이터 블록의 시작 위치는 StartCount 에 의해 결정되며, 그 크기는 MaxNumData 에 의하여 결정됩니다. 데이터 블록의 실제 크기는 'MaxNumData * 채널 수'가 됩니다.
예를 들어 CH0 와 CH1 의 두 채널에 대하여 A/D Scan 을 수행할 때 StartCount 를 i, MaxNumData 를 N 으로 하였다면 스캔 버퍼에서 사용자 버퍼로 데이터가 전달되는 것은 다음 그림과 같습니다. 데이터는 short 형으로써, Voltage 로 환산되기 이전의 정수형 값이 전달됩니다.



PARAMETER

▶ ***hDevice*** : DX_GnLoadDevice()로부터 얻어온 Device 의 핸들값 입니다.

▶ ***StartCount*** : 전달 받고자 하는 데이터 블록의 시작 Scan count.

▶ ***MaxNumData*** : 전달 받고자 하는 데이터 블록의 크기(스캔 횟수)를 지정 합니다.

Value	Meaning

양수	StartCount 부터 이후에 스캔된 데이터 중 MaxNumData 에서 지정한 수만큼 데이터를 전달합니다.
음수	StartCount 부터 이전에 스캔된 데이터 중 MaxNumData 에서 지정한 수만큼 데이터를 전달합니다.

▶ ***DestBuf*** : 데이터를 전달 받을 버퍼 포인터를 지정합니다. 이 버퍼의 크기는 MaxNumData*채널 수 보다 크거나 같아야 합니다.

사용자 버퍼의 배열 크기는 반드시 (MaxNumData * 채널 수) 보다 크거나 같아야 합니다.

▶ ***RetrivedDataCount*** : 실제 전달된 데이터 수. 만일 StartCount 이후에 현재까지 스캔된 데이터 수가 MaxNumData 에서 지정한 수 보다 작으면, 현재 스캔된 데이터까지만 전달하게 됩니다.

RETURN VALUE

□ 함수 수행 결과

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

SEE ALSO

DX_AdScanRetrBlockF4, DX_AdScanRetrBlockF8

EXAMPLE

DX_ AdRetrBlockF8 예제 참조.

NAME

DX_AdScanRetrBlockF4

- A/D Scan 전 채널에 대한 데이터(float 형)를 사용자 지정 버퍼에 반환

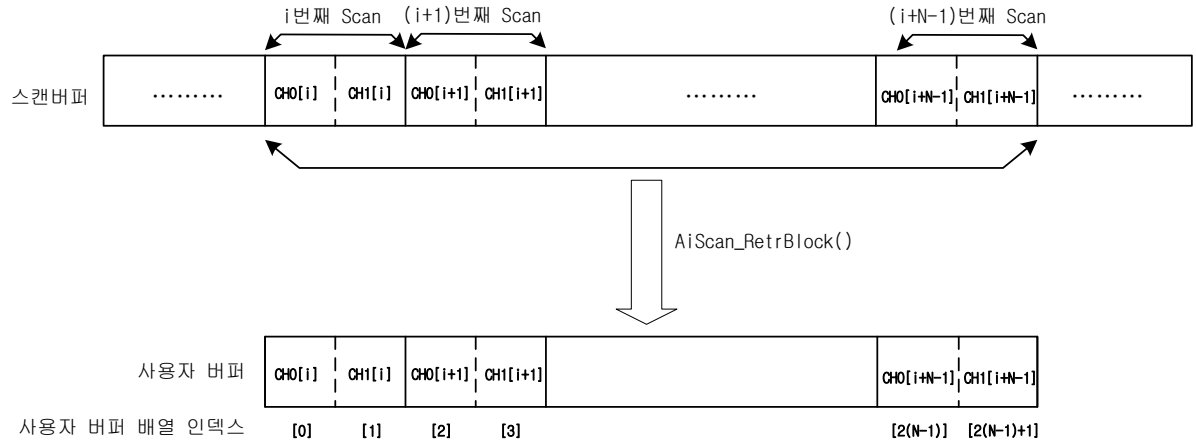
INFORMATION
Analog Input
VC++ (6, 7, 8)/VB
BCB/Delphi
Level 1
위험 요소 없음

SYNOPSIS

❒ VT_I4 DX_AdScanRetrBlockF4 ([in]VT_HANDLE hDevice, [in]VT_I4 StartCount, [in]VT_I4 MaxNumData, [in]VT_PR4 DestBuf, [out]VT_PI4 RetrivedDataCount)

DESCRIPTION

이 함수는 A/D Scan 전 채널에 대한 데이터를 사용자가 지정하는 버퍼에 전달합니다. 전달되는 데이터 블록의 시작 위치는 StartCount 에 의해 결정되며, 그 크기는 MaxNumData 에 의하여 결정됩니다. 데이터 블록의 실제 크기는 'MaxNumData * 채널 수'가 됩니다.
예를 들어 CH0 와 CH1 의 두 채널에 대하여 A/D Scan 을 수행할 때 StartCount 를 i, MaxNumData 를 N 으로 하였다면 스캔 버퍼에서 사용자 버퍼로 데이터가 전달되는 것은 다음 그림과 같습니다. 데이터는 float 형으로써, Voltage 로 환산되기 이전의 정수형 값이 전달됩니다.



PARAMETER

▶ ***hDevice*** : DX_GnLoadDevice()로부터 얻어온 Device 의 핸들값 입니다.

▶ ***StartCount*** : 전달 받고자 하는 데이터 블록의 시작 Scan count.

▶ ***MaxNumData*** : 전달 받고자 하는 데이터 블록의 크기(스캔 횟수)를 지정 합니다.

Value	Meaning

양수	StartCount 부터 이후에 스캔된 데이터 중 MaxNumData 에서 지정한 수만큼 데이터를 전달합니다.
음수	StartCount 부터 이전에 스캔된 데이터 중 MaxNumData 에서 지정한 수만큼 데이터를 전달합니다.

▶ ***DestBuf*** : 데이터를 전달 받을 버퍼 포인터를 지정합니다. 이 버퍼의 크기는 MaxNumData*채널 수 보다 크거나 같아야 합니다.

사용자 버퍼의 배열 크기는 반드시 (MaxNumData * 채널 수) 보다 크거나 같아야 합니다.

▶ ***RetrivedDataCount*** : 실제 전달된 데이터 수. 만일 StartCount 이후에 현재까지 스캔된 데이터 수가 MaxNumData 에서 지정한 수 보다 작으면, 현재 스캔된 데이터까지만 전달하게 됩니다.

RETURN VALUE

□ 함수 수행 결과

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

SEE ALSO

DX_AdScanRetrBlockI2, DX_AdScanRetrBlockF8

EXAMPLE

DX_AdRetrBlockF8 예제 참조.

NAME

DX_AdScanRetrBlockF8

- A/D Scan 전 채널에 대한 데이터(double 형)를 사용자 지정 버퍼에 반환

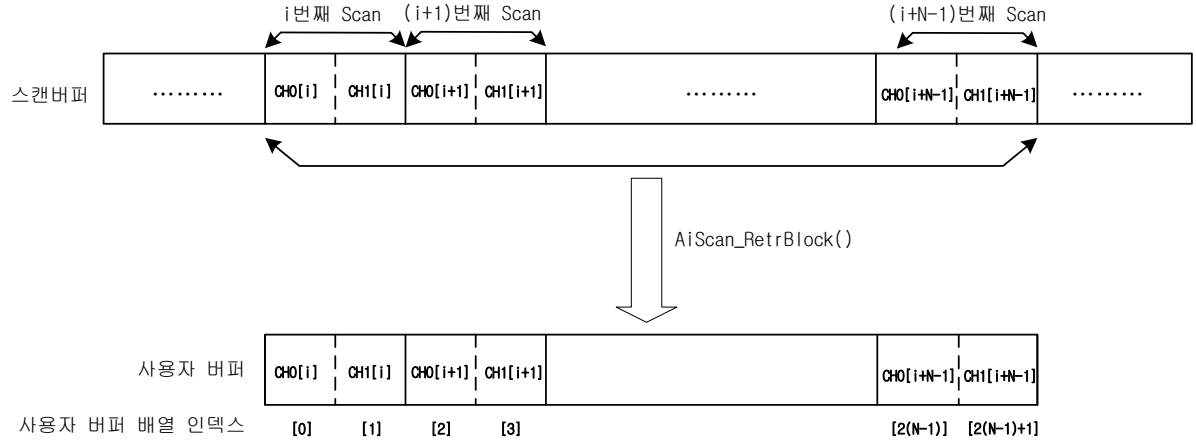
INFORMATION
Analog Input
VC++ (6, 7, 8)/VB
BCB/Delphi
Level 1
위험 요소 없음

SYNOPSIS

❒ VT_I4 DX_AdScanRetrBlockF8 ([in]VT_HANDLE hDevice, [in]VT_I4 StartCount, [in]VT_I4 MaxNumData, [in]VT_PR8 DestBuf, [out]VT_PI4 RetrivedDataCount)

DESCRIPTION

이 함수는 A/D Scan 전 채널에 대한 데이터를 사용자가 지정하는 버퍼에 전달합니다. 전달되는 데이터 블록의 시작 위치는 StartCount 에 의해 결정되며, 그 크기는 MaxNumData 에 의하여 결정됩니다. 데이터 블록의 실제 크기는 'MaxNumData * 채널 수'가 됩니다.
예를 들어 CH0 와 CH1 의 두 채널에 대하여 A/D Scan 을 수행할 때 StartCount 를 i, MaxNumData 를 N 으로 하였다면 스캔 버퍼에서 사용자 버퍼로 데이터가 전달되는 것은 다음 그림과 같습니다. 데이터는 double 형으로써, Voltage 로 환산되기 이전의 정수형 값이 전달됩니다.



PARAMETER

▶ ***hDevice*** : DX_GnLoadDevice()로부터 얻어온 Device 의 핸들값 입니다.

▶ ***StartCount*** : 전달 받고자 하는 데이터 블록의 시작 Scan count.

▶ ***MaxNumData*** : 전달 받고자 하는 데이터 블록의 크기(스캔 횟수)를 지정 합니다.

Value	Meaning

양수	StartCount 부터 이후에 스캔된 데이터 중 MaxNumData 에서 지정한 수만큼 데이터를 전달합니다.
음수	StartCount 부터 이전에 스캔된 데이터 중 MaxNumData 에서 지정한 수만큼 데이터를 전달합니다.

▶ ***DestBuf*** : 데이터를 전달 받을 버퍼 포인터를 지정합니다. 이 버퍼의 크기는 MaxNumData*채널 수 보다 크거나 같아야 합니다.

사용자 버퍼의 배열 크기는 반드시 (MaxNumData * 채널 수) 보다 크거나 같아야 합니다.

▶ ***RetrivedDataCount*** : 실제 전달된 데이터 수. 만일 StartCount 이후에 현재까지 스캔된 데이터 수가 MaxNumData 에서 지정한 수 보다 작으면, 현재 스캔된 데이터까지만 전달하게 됩니다.

RETURN VALUE

□ 함수 수행 결과

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

SEE ALSO

DX_AdScanRetrBlockI2, DX_AdScanRetrBlockF4

EXAMPLE

□ A/D CH0 와 CH1 의 두 채널을 DX_AdScanStart()을 이용하여 A/D 변환을 수행하고 그 결과를 파일로 저장하는 예제입니다.

[C / C++]

```
#define NUM_CH                  2
#define BUF_SIZE                10240

 LONG nChanList[NUM_CH] = {0, 1};
LONG nRangeList[NUM_CH] = {dxRANGE_V4, dxRANGE_V4};
 LONG IsFull = 0;
LONG nCurCnt = 0;
DOUBLE dBuf[BUF_SIZE*NUM_CH];

DX_AiSetInputType ( hDevice, dxAI_SINGLE );
DX_AdScanSetTriggerMode ( hDevice, dxTRG_S, dxINV_FALLING );

DX_AdScanSetChannelList ( hDevice, NUM_CH,&nChanList);
DX_AdScanSetRange ( hDevice, NUM_CH, & nChanList, &nRangeList);

DX_AdScanStart ( hDevice, 1000, 4000, BUF_SIZE, dxTRS_SINGLE, dxFALSE);

FILE *fp = fopen("c:\\ComiAdScan.txt", "w");
Fprintf(fp, " CH 0   CH1 \n");

 while( Is_Stop () ) .// Is_Stop() 은 가상의 함수임.
{
        DX_AdScanGetCurCount(hDevice, & CurCount);

        If( CurCount > PreCount) // 새로운 스캔 데이터가 있으면 처리
        {
                DX_AdS canRetrBlockF8( hDevice, 1, BUF_SIZE*NUM_CH, dBuf, &CurCnt);

                for( int i=0; i< CurCount; i++ )
                        fprintf(fp, "%6.2f %6.2f\n", dBuf[i*NUM_CH], dBuf[i*NUM_CH+1]);
                Sleep(100);

                // Data Process…
                PreCount += CurCount;
        }
}

fclose(fp);
DX_AdScanStop(hDevice);
```

NAME

DX_AdScanFilterConfig

– A/D Scan Filter 기능 설정

INFORMATION

Analog Input

VC++ (6, 7, 8)/VB

BCB/Delphi

Level 1

위험 요소 없음

SYNOPSIS

❒ VT_I4 DX_AdScanFilterConfig ([in]VT_HANDLE hDevice, [in]VT_I4 FilterType, [in]VT_R8 CutOffFrequency, [in]VT_I4 AvgCount)

DESCRIPTION

대상 디바이스에 대하여 A/D scan Filter 기능 상세 설정을 합니다.
DX_AdScanFilterStart() 함수를 사용하기 전에 설정을 해야 합니다..

PARAMETER

▶ ***hDevice*** : DX_GnLoadDevice()로부터 얻어온 Device 의 핸들값 입니다.

▶ ***FilterType*** **:** 필터 방식을 설정합니다.

Value	Meaning
0 (dxAIF_NONE)	필터를 사용하지 않고 A/D Scan 입력 신호값을 그대로 전달합니다.
1 (dxAIF_AVG)	Moving Average Filter
2 (dxAIF_LPF)	LowPass Filter
3 (dxAIF_TUSTIN_LPF)	Tustin LowPass Filter

DX_AdScanStart()함수와 FilterType 을 0(dxAIF_NONE) 설정 후 DX_AdScanFilterStart() 함수의 차이

두 기능 모두 그대로의 A/D Scan 입력 신호를 받아오나, Filter 함수는 디바이스내의 CPU 를 통하여 들어오기 때문에 Scan 속도가 상대적으로 느립니다.
고속 A/D 를 원할 경우에는 Filter 모드를 사용하지 마십시오.

▶ ***CutOffFrequency*** **:** 이 값은 자르고 싶은 고주파의 최소 값을 설정합니다.

.FilterType 이 dxAIF_LPF 또는 dxAIF_TUSTIN_LPF 일 경우에 사용합니다..

▶***AvgCount*** **:** 이 값은 평균값을 구하기 위한 개수를 설정합니다.

. FilterType 이 dxAIF_AVG 일 경우에 사용합니다.

RETURN VALUE

□ 함수 수행 결과

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

SEE ALSO

DX_AdScanFilterStart

EXAMPLE

DX_AdFilterStart 예제 참조.

NAME

DX_AdScanGetMinMax

– A/D Scan Filter 기능 설정

INFORMATION

Analog Input

VC++ (6, 7, 8)/VB

BCB/Delphi

Level 1

위험 요소 없음

SYNOPSIS

❒ VT_I4 DX_AdScanGetMinMax ([in]VT_HANDLE hDevice, [in]VT_I4 ChannelOrder, [out]VT_PR8 MinVolt, [out]VT_PR8 MaxVolt)

DESCRIPTION

이 함수는 A/D Scan Filter 동작 중의 최대값과 최소값을 반환합니다.
DX_AdScanFilterStart()함수로 시작한 이후부터 DX_AdScanGetMinMax()함수를 실행 하기 전까지의 최소값과 최대값을 반환합니다.

PARAMETER

▶ ***hDevice*** : DX_GnLoadDevice()로부터 얻어온 Device 의 핸들값 입니다.

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

.

▶ ***MinVolt***: 최소값(Volt) 입니다.

▶ ***MaxVolt*** : 최대값(Volt) 입니다.

RETURN VALUE

□ 함수 수행 결과

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

SEE ALSO

DX_AdScanFilterStart

EXAMPLE

□ A/D CH0 채널을 DX_AdScanFilterStart()를 이용하여 A/D 변환 하면서 최대값과 최소값을 파일에 저장하는 예제 입니다.

[C / C++]

```
#define BUF_SIZE            10240

LONG nChan = 0;
LONG nRange = dxRANGE_V4;
LONG CurCount = 0, PreCount = 0;
DOUBLE MinVolt = 0, MaxVolt = 0;


DX_AiSetInputType ( hDevice, dxAI_SINGLE );
DX_AdScanSetTriggerMode ( hDevice, dxTRG_S, dxINV_FALLING );

DX_AdScanSetChannelList ( hDevice, 1,& nChan);
DX_AdScanSetRange ( hDevice, 1, & nChan, &nRange);

/*****************************************************************/
/* 중요                                                          */
/* 10 개의 데이터의 평균 입력 값을 스캔 받음                     */
/*****************************************************************/
DX_AdScanFilterConfig( hDevice, dxAIF_AVG, 0, 10 );

DX_AdScanFilterStart ( hDevice, 1000, 4000, BUF_SIZE);

FILE *fp = fopen("c:\\ComiAdScan.txt", "w");
Fprintf(fp, " CH 0      Min     Max\n");

 while( Is_Stop () ) .// Is_Stop() 은 가상의 함수임.
{
        DX_AdScanGetCurCount(hDevice, & CurCount);

        If( CurCount > PreCount) // 새로운 스캔 데이터가 있으면 처리
        {
                DX_AdScanGetMinMax(hDevice, 0, &MinVolt, &MaxVolt);

                fprintf(fp, "\t %6.2f %6.2f\n", MinVolt, MaxVolt);

                // Data Process…
        }
}

fclose(fp);
DX_AdScanStop(hDevice);
```

Chapter

6

Analog Output Functions

아날로그 출력장치는 컴퓨터로부터 전달되는 디지털 정보(명령)를 아날로그 전압이나 전류로 변환하여 줍니다.

아날로그 출력 장치는 아날로그 입력 장치의 반대 기능을 수행하는 장치입니다. 아날로그 출력장치는 컴퓨터로부터 전달되는 디지털 정보(명령)를 아날로그 전압이나 전류로 변환하여 줍니다. 아날로그 출력 장치는 컴퓨터가 단순한 계측용도가 아닌 각종 시스템을 자동으로 제어할 수 있는 수단을 제공합니다. 아날로그 출력 장치는 아날로그 입력 장치와 함께 사용되어 수집된 정보를 PID 제어 로직 등을 통하여 시스템을 궤환제어하는 경우가 많습니다.

고급 아날로그 출력 장치는 Waveform Generation 기능을 포함합니다. 이러한 경우에는 아날로그 출력 장치가 Function Generator 의 역할을 수행할 수 있습니다.

6 아날로그 출력 함수

이 단원에서는 Analog Output 에 관한 함수를 소개합니다. CMD-SDK 에서는 두 가지 형태의 Analog Output 기능이 있습니다.

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

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

6.1 일반적인 아날로그 출력

이 단원에서는 일반적인 아날로그 출력 함수들을 소개합니다. 일반적인 D/A 는 사용자가 지정한 전압(Voltage) 값을 출력하는 기능입니다.

이와 관련된 함수 리스트 및 사용 가능 디바이스는 다음과 같습니다.

Function	Device
DX_DaClear	COMI-DX10x, COMI-DX30x
DX_DaSetRange	
DX_DaGetRange	
DX_DaOut	

[표 6-1] Analog Output 일반 함수 리스트 및 사용 가능 디바이스

6.1.1 함수 요약

일반적인 아날로그 출력과 관련된 함수들의 리스트는 다음과 같습니다.

Summary of Functions
❒ VT_I4 DX_DaClear ([in]VT_HANDLE hDevice) 모든 D/A 채널을 초기화 합니다.
❒ VT_I4 DX_DaSetRange ([in]VT_HANDLE hDevice , [in]VT_I4 Channel, [in]VT_I4 Range) 각 D/A 채널의 출력 범위를 설정합니다.
❒ VT_I4 DX_DaGetRange ([in]VT_HANDLE hDevice , [in]VT_I4 Channel, [out]VT_PI4 Range) 각 D/A 채널의 출력 범위를 반환합니다.
❒ VT_I4 DX_DaOut ([in]VT_HANDLE hDevice , [in]VT_I4 Channel, [in]VT_R8 OutVolt) 각 D/A 채널의 출력을 제어합니다.

6.1.2 함수 설명

NAME

DX_DaClear

- 대상 D/A 채널을 초기화

INFORMATION

- Analog Output
- VC++ (6, 7, 8)/VB
- BCB/Delphi
- Level 1
- 위험 요소 없음

SYNOPSIS

❒ VT_I4 DX_DaClear ([in]VT_HANDLE hDevice)

DESCRIPTION

이 함수는 대상 디바이스의 모든 아날로그 출력 채널을 초기화 합니다.

PARAMETER

▶ ***hDevice*** : DX_GnLoadDevice()로부터 얻어온 Device 의 핸들값 입니다.

RETURN VALUE

□ 함수 수행 결과

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

NAME

DX_DaSetRange / DX_DaGetRange

- 대상 D/A 채널의 출력 범위 설정 / 반환

INFORMATION

Analog Output

VC++ (6, 7, 8)/VB

BCB/Delphi

Level 1

위험 요소 없음

SYNOPSIS

- ❒ VT_I4 DX_DaSetRange ([in]VT_HANDLE hDevice , [in]VT_I4 Channel, [in]VT_I4 Range)
- ❒ VT_I4 DX_DaGetRange ([in]VT_HANDLE hDevice , [in]VT_I4 Channel, [out]VT_PI4 Range)

DESCRIPTION

이 함수는 대상 디바이스의 아날로그 출력 채널의 출력 범위를 설정/ 반환합니다.

PARAMETER

▶ ***hDevice*** : DX_GnLoadDevice()로부터 얻어온 Device 의 핸들값 입니다.

▶ ***Channel*** : 아날로그 출력 채널 번호를 지정합니다. 채널 번호는 0 부터 시작합니다.

▶ ***Range***: 아날로그 출력 채널의 출력 범위 입니다.

Value	Meaning
0 (dxRANGE_V0)	0v ~ 5v
1 (dxRANGE_V1)	0v ~ 10v
2 (dxRANGE_V2)	0v ~ 10.8v
3 (dxRANGE_V3)	-5v ~ 5v
4 (dxRANGE_V4)	-10v ~ 10v
5 (dxRANGE_V5)	-10.8v ~ 10.8v

RETURN VALUE

□ 함수 수행 결과

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

NAME

DX_DaOut

- 대상 D/A 채널을 통해 전압(Voltage) 값 출력

INFORMATION

Analog Output

VC++ (6, 7, 8)/VB

BCB/Delphi

Level 1

위험 요소 없음

SYNOPSIS

❒ VT_I4 DX_DaOut ([in]VT_HANDLE hDevice , [in]VT_I4 Channel, [in]VT_R8 OutVolt)

DESCRIPTION

이 함수는 대상 아날로그 출력 채널에 대하여 지정한 전압(Voltage) 값을 출력합니다.

PARAMETER

▶ ***hDevice*** : DX_GnLoadDevice()로부터 얻어온 Device 의 핸들값 입니다.

▶ ***Channel*** : 아날로그 출력 채널 번호를 지정합니다. 채널 번호는 0 부터 시작합니다.

▶ ***OutVolt*** : 아날로그 출력 전압(Voltage) 값.
출력 가능한 전압 범위는 DX_DaSetRange()로부터 설정한 범위로 지정됩니다..

RETURN VALUE

□ 함수 수행 결과

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

6.2 Waveform Generation

Waveform Generation 기능은 Sine Wave 또는 Square Wave 등과 같이 주기성을 가지는 신호를 아날로그 출력을 이용하여 자동으로 생성해주는 기능입니다.

Waveform 데이터는 사용자가 배열로 지정하도록 되어 있으며 한 주기의 Waveform 을 구성하는 데이터 수는 512 개 이하에서 자유롭게 설정할 수 있습니다.

Waveform Generation 기능을 지원하는 디바이스는 사용자가 지정하는 Waveform 데이터를 디바이스에 내장된 FIFO 메모리에 로드 한 후 사용자가 지정한 출력 주파수에 따라 내부 타이머를 이용하여 반복적으로 아날로그 출력을 업데이트합니다. 또한 사용자는 출력되는 Waveform 의 횟수를 제한할 수도 있습니다

이와 관련된 함수 리스트 및 사용 가능 디바이스는 다음과 같습니다.

Function	Device
DX_WfmSetUpdateFreq	COMI-DX10x, COMI-DX30x
DX_WfmGetUpdateFreq	
DX_WfmStart	
DX_WfmStop	

[표 6-2] Waveform Generation 관련 함수 리스트 및 사용 가능 디바이스

6.2.1 함수 요약

일반적인 아날로그 출력과 관련된 함수들의 리스트는 다음과 같습니다.

Summary of Functions
❒ VT_I4 DX_WfmStart ([in]VT_HANDLE hDevice, [in[VT_I4 Channel, [in]VT_PR8 DataBuffer, [in]VT_I4 NumData, [in]VT_I4 nPPS) Waveform Generation 을 시작합니다.
❒ VT_I4 DX_WfmStop ([in]VT_HANDLE hDevice, [in[VT_I4 Channel) Waveform Generation 을 종료합니다.

6.2.2 함수 설명

NAME

DX_WfmStart

- Waveform Generation 시작

INFORMATION

- Analog Output
- VC++ (6, 7, 8)/VB
- BCB/Delphi
- Level 1
- 위험 요소 없음

SYNOPSIS

❒ VT_I4 DX_WfmStart ([in]VT_HANDLE hDevice, [in[VT_I4 Channel, [in]VT_PR8 DataBuffer, [in]VT_I4 NumData, [in]VT_I4 nPPS)

DESCRIPTION

이 함수는 대상 아날로그 출력 채널에 대하여 Waveform Generation 을 시작합니다.

PARAMETER

▶ ***hDevice*** : DX_GnLoadDevice()로부터 얻어온 Device 의 핸들값 입니다.

▶ ***Channel*** : 아날로그 출력 채널 번호를 지정합니다. 채널 번호는 0 부터 시작합니다.

▶ ***DataBuffer*** : Waveform 데이터를 담은 버퍼의 주소 값(포인터)

▶ ***NumData*** : 데이터 버퍼 크기 (Byte 단위가 아닌 버퍼에 담겨진 데이터의 수).

주의	NumData 값은 512 보다는 작거나 같아야 합니다. Waveform 데이터는 사용자가 배열로 지정하도록 되어 있으며 한 주기의 Waveform 을 구성하는 데이터 수는 512 개 이하에서 자유롭게 설정할 수 있습니다.

▶ ***nPPS*** : Waveform Generation 의 업데이트 주기를 PPS 단위로 설정 합니다.

안내	PPS 설정 예.
	예를 들어 100 개의 데이터로 한 주기를 구성하였다면, 10Hz 의 신호를 만들기 위한 Waveform Generation 주기 는 1000[PPS] 입니다.

.

RETURN VALUE

□ 함수 수행 결과

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

SEE ALSO

DX_WfmStop

EXAMPLE

□ 10kHz, -5 ~ 5 Volt, Square Wave 신호를 계속 발생시키는 예제입니다.

[C / C++]

```
LONG DataBuf[2] = {-5, 5};

DX_DaSetRange( hDevice, 0, dxRANGE_V3 );

DX_WfmStart( hDevice, 0, DataBuf, 2, 1000 );
```

□ 10 KHz, -5 ~ 5 Volt, Square Wave 신호를 1000 개 발생시키는 예제입니다. // yds

[C / C++]

```
double DataBuffer[2]={-5, 5};

fActFreq = AoWfmStart(0, DataBuffer, 2, 10000 * 2, 1000);

if (ComiDaq1.GnGetErrorCode() < 0)          // Error 처리
{
        ComiDaq1.GnShowLastError();
        exit(0);
}
```

[Visual Basic]

```
Dim DataBuffer(1)
DataBuffer(0) = -5
DataBuffer(1) = 5

Call ComiDaq.AoWfmStart(0, DataBuffer(0), 2, 10000 * 2, 1000)
If (ComiDaq1.GnGetErrorCode() < 0) Then   ' Error 처리
```

```
        ComiDaq1.GnShowLastError
        End
End If
```

NAME

DX_WfmStop

– Waveform Generation 을 종료

INFORMATION

Analog Output

VC++ (6, 7, 8)/VB

BCB/Delphi

Level 1

위험 요소 없음

SYNOPSIS

❒ VT_I4 DX_WfmStop ([in]VT_HANDLE hDevice, [in[VT_I4 Channel)

DESCRIPTION

이 함수는 Waveform Generation 을 종료합니다.

PARAMETER

▶ ***hDevice*** : DX_GnLoadDevice()로부터 얻어온 Device 의 핸들값 입니다.

▶ ***Channel*** : 아날로그 출력 채널 번호를 지정합니다. 채널 번호는 0 부터 시작합니다.

RETURN VALUE

□ 함수 수행 결과

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

SEE ALSO

DX_WfmStart

Chapter 7

Digital Input/Output functions

디지털 입력 제품은 광센서, 근접센서 등과 같이 전기적 디지털 신호를 출력하는 센서들의 신호를 컴퓨터가 인식할 수 있는 디지털 데이터로 변환해주는 장치입니다.

디지털 입력 제품은 광센서, 근접센서 등과 같이 전기적 디지털 신호를 출력하는 센서들의 신호를 컴퓨터가 인식할 수 있는 디지털 데이터로 변환해주는 장치이며 주로 ON/OFF 상태를 감시하는 용도로 많이 사용됩니다. 디지털 출력 제품은 주로 스위치를 제어하는데 많이 사용되며 컴퓨터의 ON/OFF 로직을 전기적인 디지털 신호로 변환해주는 역할을 합니다.

7 디지털 입출력 함수

이 단원에서는 Digital Input 과 Output 에 관한 함수를 소개합니다. 일반적으로 Digital Input 은 스위치(Switch)의 상태를 확인하는 용도로 사용되고, Digital Output 은 스위치의 상태를 제어하는 용도로 사용됩니다.

7.1 일반적인 디지털 입출력

이 단원에서는 일반적인 디지털 입출력 기능에 관련된 함수들을 소개합니다. 일반적인 디지털 입출력은 제어 시스템에 장착된 PCI 보드에서 디지털 입출력을 직접 제어하는 것을 의미합니다.

이와 관련된 함수 리스트 및 사용 가능 디바이스는 다음과 같습니다.

함수 명	사용 가능 디바이스
DX_DioSetUsage	COMI-DX10x, COMI-DX20x, COMI-DX30x, COMI-DX50x
DX_DioGetUsage	
DX_DiGetOne	
DX_DiGetAll	
DX_DoGetOne	
DX_DoGetAll	
DX_DoPutOne	
DX_DoPutAll	

[표 7-1] Digital Input/Output 에 관련된 함수 리스트 및 사용 가능 디바이스 안내

7.1.1 함수 요약

일반적인 디지털 입출력 기능과 관련된 함수들의 리스트는 다음과 같습니다.

Summary of Functions
❒ VT_I4 DX_DioSetUsage ([in]VT_HANDLE hDevice, [in]VT_I4 Usage) 대상 디지털 입출력 채널의 입출력 모드를 설정합니다.
❒ VT_I4 DX_DioGetUsage ([in]VT_HANDLE hDevice, [out]VT_PI4 Usage) 대상 디지털 입출력 채널의 입출력 모드 설정 상태를 반환합니다.
❒ VT_I4 DX_DiGetOne ([in]VT_HANDLE hDevice, [in]VT_I4 Channel , [out]VT_PI4 State) 대상 디지털 입력 채널의 입력 상태를 반환합니다.
❒ VT_I4 DX_DiGetAll ([in]VT_HANDLE hDevice, [out]VT_PI4 States) 대상 디바이스의 최대 32 채널에 대한 디지털 입력 채널의 입력 상태를 반환합니다.
❒ VT_I4 DX_DoGetOne ([in]VT_HANDLE hDevice, [in]VT_I4 Channel, [out]VT_PI4 State) 대상 디지털 출력 채널의 입력 상태를 반환합니다.
❒ VT_I4 DX_DoGetAll([in]VT_HANDLE hDevice, [out]VT_PI4 States) 대상 디바이스의 최대 32 채널에 대한 디지털 출력 채널의 입력 상태를 반환합니다.
❒ VT_I4 DX_DoPutOne ([in]VT_HANDLE hDevice, [in]VT_I4 Channel, [in]VT_I4 State) 대상 디지털 출력 채널을 통해 출력 상태를 제어합니다.
❒ VT_I4 DX_DoPutAll ([in]VT_HANDLE hDevice, [in]VT_I4 States) 대상 디바이스의 최대 32 채널에 대한 디지털 출력 채널의 출력 상태를 제어합니다.

7.1.2 함수 설명

NAME

DX_DioSetUsage / DX_DioGetUsage

- 디지털 입출력 모드 설정 및 설정 상태 반환

INFORMATION

- Digital Input / Output
- VC++ (6, 7, 8)/VB
- BCB/Delphi
- Level 1
- 위험 요소 없음

SYNOPSIS

- ❒ VT_I4 DX_DioSetUsage ([in]VT_HANDLE hDevice, [in]VT_I4 Usage)
- ❒ VT_I4 DX_DioGetUsage ([in]VT_HANDLE hDevice, [out]VT_PI4 Usage)

DESCRIPTION

디지털 입출력 채널의 입출력 모드를 설정/ 반환 합니다.

PARAMETER

▶ ***hDevice*** : DX_GnLoadDevice()로부터 얻어온 Device 의 핸들값 입니다.

▶ ***Usage*** : 디지털 입출력 채널의 입출력 모드. 채널 수는 디바이스에 따라 다르므로 H/W 매뉴얼을 참조하여 주시기 바랍니다.

Value	Meaning
0 (dxDI_ONLY)	모든 채널을 디지털 입력 채널로 사용합니다.
1 (dxDI_DO)	모든 채널의 반은 디지털 입력, 나머지 반은 디지털 출력 채널로 사용합니다.
2 (dxDO_DI)	모든 채널의 반은 디지털 출력, 나머지 반은 디지털 입력 채널로 사용합니다.
3 (dxDO_ONLY)	모든 채널을 디지털 출력 채널로 사용합니다.

RETURN VALUE

□ 함수 수행 성공 여부.

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

0 (dxERR_NONE)	수행 성공.

REFERENCE

□ 디지털 입출력 채널의 입출력 모드 설정 시 채널 번호 안내

디지털 입출력 채널의 입출력 모드 설정에 따라 디지털 입력과 디지털 출력 채널 번호가 어떻게 되는지 혼동될 수 있습니다. 디지털 입력과 출력의 채널 번호는 언제나 각각 0 번부터 시작합니다.

예를 들어 COMI-DX101 보드는 16 채널의 디지털 입출력 채널을 제공하는데 이를 1(dxDI_DO)로 설정하였다면 0 ~ 7 번 채널은 디지털 입력 CH0 ~ CH7 로 설정되고, 8 ~ 15 번 채널은 디지털 출력 CH0 ~ CH7 로 설정됩니다. 반대로 2(dxDO_DI) 모드로 설정하였다면 0 ~ 7 번 채널은 디지털 출력 CH0 ~ CH7 로 설정되고, 8 ~ 15 번 채널은 디지털 입력 CH0 ~ CH7 로 설정됩니다.

NAME

DX_DiGetOne

- 대상 디지털 입력 채널의 입력 상태 반환

INFORMATION

Digital Input

VC++ (6, 7, 8)/VB

BCB/Delphi

Level 1

위험 요소 없음

SYNOPSIS

❒ VT_I4 DX_DiGetOne ([in]VT_HANDLE hDevice, [in]VT_I4 Channel, [out]VT_PI4 State)

DESCRIPTION

이 함수는 대상 디지털 입력 채널의 입력 상태를 반환합니다.

PARAMETER

▶ ***hDevice*** : DX_GnLoadDevice()로부터 얻어온 Device 의 핸들값 입니다.

▶ ***Channel*** : 디지털 입력 채널 번호를 지정합니다. 채널 번호는 0 부터 시작합니다.

▶ ***State*** : 대상 디지털 입력 채널의 상태를 반환합니다.

Value	Meaning
0 (dxFALSE)	OFF
1 (dxTRUE)	ON

RETURN VALUE

□ 함수 수행 결과

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

SEE ALSO

DX_DiGetAll

EXAMPLE

□ 이 예제는 DX_DoPutOne() 함수와 DX_DiGetOne() 함수를 사용하여, DO0 를 ON/OFF 교대로 출력하고 DI0 의 상태를 계속 체크합니다. DO0 와 DI0 를 서로 연결하면 DO0 의 출력을 DI0 를 통하여 ON/OFF 상태가 반복됨을 확인할 수 있습니다.

[C / C++]

```
LONG do_states=0, di_states = 0;


DX_DioSetUsage( hDevice, dxDI_DO );

While ( !kbhit() )
{
        do_states  ^= 1; /// ON/OFF 반전
        DX_DoPutOne( hDevice, 0, do_states);

        DX_DiGetOne( hDevice, 0, & di_states);
        printf("Status of D/I CH0 = %d\n", di_states);

        Sleep(500);
}
```

NAME

DX_DiGetAll

- 대상 디바이스의 최대 32 채널에 대한 디지털 입력 채널의 입력 상태 반환

INFORMATION

- Digital Input
- VC++ (6, 7, 8)/VB
- BCB/Delphi
- Level 1
- 위험 요소 없음

SYNOPSIS

❒ VT_I4 DiGetAll ([in]VT_HANDLE hDevice, [out]VT_PI4 States)

DESCRIPTION

이 함수는 대상 디바이스의 최대 32 채널에 대한 디지털 입력 채널의 입력 상태를 반환합니다.

PARAMETER

▶ ***hDevice*** : DX_GnLoadDevice()로부터 얻어온 Device 의 핸들값 입니다.

▶ ***States*** : 대상 디지털 입력 디바이스의 최대 32 개의 채널에 대한 입력 상태를 32 비트 값으로 반환합니다. 각 비트는 채널 순서와 일치하며, 각 채널의 ON/OFF 상태를 나타냅니다.

단, 디바이스에 따라 32 채널 미만의 디지털 입력 채널을 지원하는 경우에는 BIT0 부터 해당 채널 수 만큼의 비트만 사용하면 됩니다.

RETURN VALUE

□ 함수 수행 결과

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

SEE ALSO

DX_DiGetOne

EXAMPLE

□ 이 예제는 DiGetAll()과 DoPutAll() 함수를 사용하여 디지털 입력과 디지털 출력 채널의 8 채널을 동시에 제어하는 예제입니다.

[C / C++]

```
LONG do_states=0, di_states = 0;
int di_each[8], i;

ComiDaq1.DioSetUsage( dxDI_DO );

while( !kbhit() )
{
        do_states = ~do_states;          // 모든 DO 채널 ON/OFF 상태 반전
        DX_DoPutAll( do_states );        // Put D/O

        /* Get D/I and print on screen */
        di_states =DX_DiGetAll();

        /* di_states 는 전 채널의 ON/OFF 상태를 담고 있습니다. */
        /* Bit mask 를 통해 각 채널의 상태를 확인합니다. */
        for( int i = 0; i < 8; i++ )
                di_each[i] = ( di_states >> I ) & 0x1;

        printf("States of DI0 ~ DI7 = %d %d %d %d %d %d %d %d\n",
                        di_each[0], di_each[1], di_each[2], di_each[3],
                        di_each[4], di_each[5], di_each[6], di_each[7]);
        Sleep(500);
}
```

NAME

DX_DoGetOne

- 대상 디지털 출력 채널의 출력 상태 반환

INFORMATION

Digital Output

VC++ (6, 7, 8)/VB

BCB/Delphi

Level 1

위험 요소 없음

SYNOPSIS

❒ VT_I4 DX_DoGetOne ([in]VT_HANDLE hDevice, [in]VT_I4 Channel, [out]VT_PI4 State)

DESCRIPTION

이 함수는 대상 디지털 출력 채널의 출력 상태를 반환합니다.

PARAMETER

▶ ***hDevice*** : DX_GnLoadDevice()로부터 얻어온 Device 의 핸들값 입니다.

▶ ***Channel*** : 디지털 출력 채널 번호를 지정합니다. 채널 번호는 0 부터 시작합니다.

▶ ***State*** : 대상 디지털 입력 채널의 상태를 반환합니다.

Value	Meaning
0 (dxFALSE)	OFF
1 (dxTRUE)	ON

RETURN VALUE

□ 함수 수행 결과

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

SEE ALSO

DX_DoGetAll

NAME

DX_DoGetAll

- 대상 디바이스의 최대 32 채널에 대한 디지털 출력 채널의 출력 상태 반환

INFORMATION
Digital Output
VC++ (6, 7, 8)/VB
BCB/Delphi
Level 1
위험 요소 없음

SYNOPSIS

❒ VT_I4 DX_DoGetAll ([in]VT_HANDLE hDevice, [out]VT_PI4 States)

DESCRIPTION

이 함수는 대상 디바이스의 최대 32 채널에 대한 디지털 출력 채널의 출력 상태를 반환합니다.

PARAMETER

▶ ***hDevice*** : DX_GnLoadDevice()로부터 얻어온 Device 의 핸들값 입니다.

▶ ***States*** : 대상 디지털 출력 디바이스의 최대 32 개의 채널에 대한 출력 상태를 32 비트 값으로 반환합니다. 각 비트는 채널 순서와 일치하며, 각 채널의 ON/OFF 상태를 나타냅니다.

단, 디바이스에 따라 32 채널 미만의 디지털 출력 채널을 지원하는 경우에는 BIT0 부터 해당 채널 수 만큼의 비트만 사용하면 됩니다.

RETURN VALUE

□ 함수 수행 결과

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

SEE ALSO

DX_DoGetOne

NAME

DX_DoPutOne

- 대상 디지털 출력 채널의 출력 상태 제어

INFORMATION

Digital Output

VC++ (6, 7, 8)/VB

BCB/Delphi

Level 1

위험 요소 없음

SYNOPSIS

❒ VT_I4 DX_DoPutOne ([in]VT_HANDLE hDevice, [in]VT_I4 Channel, [in]VT_I4 State)

DESCRIPTION

이 함수는 대상 디지털 출력 채널의 ON/OFF 출력 상태를 제어 합니다.

PARAMETER

▶ ***hDevice*** : DX_GnLoadDevice()로부터 얻어온 Device 의 핸들값 입니다.

▶ ***Channel*** : 디지털 출력 채널 번호를 지정합니다. 채널 번호는 0 부터 시작합니다.

▶ ***State*** : 디지털 출력 채널의 출력 상태.

Value	Meaning
0 (dxFALSE)	OFF.
1 (dxTRUE)	ON.

RETURN VALUE

□ 함수 수행 결과

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

SEE ALSO

DX_DoPutAll

EXAMPLE

□ 이 예제는 DX_DoPutOne() 함수와 DX_DiGetOne() 함수를 사용하여, DO0 를 ON/OFF 교대로 출력하고 DI0 의 상태를 계속 체크합니다. DO0 와 DI0 를 서로 연결하면 DO0 의 출력을 DI0 를 통하여 ON/OFF 상태가 반복됨을 확인할 수 있습니다.

[C / C++]

```
LONG do_states=0, di_states = 0;


DX_DioSetUsage( hDevice, dxDI_DO );

While ( !kbhit() )
{
        do_states  ^= 1; /// ON/OFF 반전
        DX_DoPutOne( hDevice, 0, do_states);

        DX_DiGetOne( hDevice, 0, & di_states);
        printf("Status of D/I CH0 = %d\n", di_states);

        Sleep(500);
}
```

NAME

DX_DoPutAll

- 대상 디바이스의 최대 32 채널에 대한 디지털 출력 채널의 출력 상태 제어

INFORMATION

- Digital Output
- VC++ (6, 7, 8)/VB
- BCB/Delphi
- Level 1
- 위험 요소 없음

SYNOPSIS

❒ VT_I4 DoPutAll ([in]VT_HANDLE hDevice, [in]VT_I4 States)

DESCRIPTION

이 함수는 대상 디바이스의 최대 32 채널에 대한 디지털 출력 채널의 출력 상태를 제어합니다.

PARAMETER

▶ ***hDevice*** : DX_GnLoadDevice()로부터 얻어온 Device 의 핸들값 입니다.

▶ ***States*** : 대상 디바이스의 최대 32 채널에 대한 디지털 출력 상태를 32 비트 값으로 설정합니다. 각 비트는 채널 순서와 일치하며, 각 채널의 ON/OFF 상태를 나타냅니다.

단, 디바이스에 따라 32 채널 미만의 디지털 출력 채널을 지원하는 경우에는 BIT0 부터 해당 채널 수 만큼의 비트만 사용하면 됩니다.

RETURN VALUE

□ 함수 수행 결과

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

SEE ALSO

DX_DoPutOne

EXAMPLE

이 예제는 DiGetAll()과 DoPutAll() 함수를 사용하여 디지털 입력과 디지털 출력 채널의 8 채널을 동시에 제어하는 예제입니다.

[C / C++]

```
LONG do_states=0, di_states = 0;
int di_each[8], i;

ComiDaq1.DioSetUsage( dxDI_DO );

while( !kbhit() )
{
        do_states = ~do_states;          // 모든 DO 채널 ON/OFF 상태 반전
        DX_DoPutAll( do_states );        // Put D/O

        /* Get D/I and print on screen */
        di_states =DX_DiGetAll();

        /* di_states 는 전 채널의 ON/OFF 상태를 담고 있습니다. */
        /* Bit mask 를 통해 각 채널의 상태를 확인합니다. */
        for( int i = 0; i < 8; i++ )
                di_each[i] = ( di_states >> I ) & 0x1;

        printf("States of DI0 ~ DI7 = %d %d %d %d %d %d %d %d\n",
                        di_each[0], di_each[1], di_each[2], di_each[3],
                        di_each[4], di_each[5], di_each[6], di_each[7]);
        Sleep(500);
}
```

Chapter

8

Counter functions

카운터는 펄스의 수를 계수하는 장치입니다. 일반적으로 회전체의 속도 또는 위치를 계측하게 하기 위해 펄스 출력을 발생하는 장치가 많습니다. 또한, 일부 변위 센서나 유량계 등은 데이터의 정확성을 위하여 펄스 출력을 발생하는 경우가 많습니다. 이러한 때에 사용되는 것이 카운터입니다. 일부 카운터 장치는 정밀한 간격의 펄스 출력을 발생하기 위해 타이머로도 사용됩니다.

01 단원에서는 카운터에 관련된 함수들을 소개합니다. 최근에 많이 사용되는 카운터는 엔코더 카운터입니다. 엔코더는 A 상과 B 상 두 출력의 위상차를 통해 방향을 감지할 수 있도록 합니다. 일반 카운터가 한 방향으로만 증가하거나 감소하는데 반하여 엔코더 카운터는 방향에 따라 UP/DOWN 카운팅이 가능하여 모터의 정밀 위치 제어에 많이 사용됩니다

8 카운터 함수

이 단원에서는 카운터와 관련된 함수를 소개합니다. 카운터는 펄스신호를 카운트 할 경우에 사용되는 함수입니다. 카운터에 관련된 함수는 다음과 같습니다.

8.1 32Bit 카운터

COMI-DX series 보드에는 사용자가 사용할 수 있는 32Bit Counter(COMI-Counter)가 장착되어 있습니다. COMI-Counter 란 커미조아에서 FPGA 로 구현한 32Bit Resolution 카운터를 의미 하며, 일반적인 데이터 획득에 필요한 기능이 내장되어 있습니다. COMI-Counter 에 대한 자세한 내용은 해당 디바이스 하드웨어 매뉴얼의 부록을 참조하여 주시기 바랍니다

32Bit 카운터 기능과 관련된 함수 리스트 및 사용 가능 디바이스는 다음과 같습니다.

함수 명	사용 가능 디바이스
DX_CntSetConfig	COMI-DX10x, COMI-DX20x, COMI-DX30x, COMI-DX50x
DX_CntGetConfig	
DX_CntSetFilter	
DX_CntGetFilter	
DX_CntSetDefault	
DX_CntGetDefault	
DX_CntSetClearMode	
DX_CntGetClearMode	
DX_CntClear	
DX_CntStart	
DX_CntStop	
DX_CntIsActive	
DX_CntGetCount	

[표 8-2] 32Bit 카운터에 관련된 함수 리스트 및 사용 가능 디바이스 안내

8.1.1 함수 요약

32Bit 카운터와 관련된 함수들의 리스트는 다음과 같습니다.

Summary of Functions
❒ VT_I4 DX_CntSetConfig ([in]VT_HANDLE hDevice, [in]VT_I4 Channel, [in]VT_I4 Edge, [in]VT_I4 UpDown, [in]VT_I4 CntSrc) 펄스 주파수를 측정할 카운터 채널의 동작 방식을 설정합니다.
❒ VT_I4 DX_CntGetConfig ([in]VT_HANDLE hDevice, [in]VT_I4 Channel, [out]VT_PI4 Edge, [out]VT_PI4 UpDown, [out]VT_PI4 CntSrc) 펄스 주파수를 측정할 카운터 채널의 동작 방식의 설정값을 반환합니다.
❒ VT_I4 DX_CntSetFilter ([in]VT_HANDLE hDevice, [in]VT_I4 Channel, [in]VT_I4 Frequency) 펄스 주파수를 측정할 카운터 채널의 필터를 설정합니다.
❒ VT_I4 DX_CntGetFilter ([in]VT_HANDLE hDevice, [in]VT_I4 Channel, [out]VT_PI4 Frequency) 펄스 주파수를 측정할 카운터 채널의 필터의 설정값을 반환합니다.
❒ VT_I4 DX_CntSetDefault ([in]VT_HANDLE hDevice, [in]VT_I4 Channel, [in]VT_I4 Count) 펄스 주파수를 측정할 카운터 채널의 초기값을 설정합니다.
❒ VT_I4 DX_CntGetDefault ([in]VT_HANDLE hDevice, [in]VT_I4 Channel, [out]VT_PI4 Count) 펄스 주파수를 측정할 카운터 채널의 초기값을 반환합니다.
❒ VT_I4 DX_CntSetClearMode ([in]VT_HANDLE hDevice, [in]VT_I4 Channel, [in]VT_I4 ClrMode) 펄스 주파수를 측정할 카운터 채널의 Clear Mode 를 설정합니다.
❒ VT_I4 DX_CntSetClearMode ([in]VT_HANDLE hDevice, [in]VT_I4 Channel, [in]VT_I4 ClrMode) 펄스 주파수를 측정할 카운터 채널의 Clear Mode 를 반환합니다.
❒ VT_I4 DX_CntClear ([in]VT_HANDLE hDevice, [in]VT_I4 Channel) 펄스 주파수를 측정할 카운터 채널의 카운트값을 초기화합니다.
❒ VT_I4 DX_CntStart ([in]VT_HANDLE hDevice, [in]VT_I4 Channel) 펄스 주파수를 측정할 카운터 채널의 카운트를 시작합니다.
❒ VT_I4 DX_CntStop ([in]VT_HANDLE hDevice, [in]VT_I4 Channel) 펄스 주파수를 측정할 카운터 채널의 카운트를 종료합니다.
❒ VT_I4 DX_CntIsActive ([in]VT_HANDLE hDevice, [in]VT_I4 Channel, [out]VT_PI4 IsActive) 펄스 주파수를 측정할 카운터 채널의 카운트 동작을 확인합니다.
❒ VT_I4 DX_CntGetCount ([in]VT_HANDLE hDevice, [in]VT_I4 Channel, [out]VT_PI4 Count) 펄스 주파수를 측정할 카운터 채널의 카운트 값을 반환합니다.

8.1.2 함수 설명

NAME

DX_CntSetConfig / DX_CntConfig

- 카운터 채널의 동작 방식을 설정/반환

INFORMATION

DX_CntSetConfig/
DX_CntGetConfig

VC++ (6, 7, 8)/VB
BCB/Delphi

Level 1

위험 요소 없음

SYNOPSIS

❒ VT_I4 DX_CntSetConfig ([in]VT_HNADLE hDevice, [in]VT_I4 Channel, [in]VT_I4 Edge, [in]VT_I4 UpDown, [in]VT_I4 CntScr)

❒ VT_I4 DX_CntGetConfig ([in]VT_HNADLE hDevice, [in]VT_I4 Channel, [out]VT_PI4 Edge, [out]VT_PI4 UpDown, [out]VT_PI4 CntScr)

DESCRIPTION

지정한 채널의 펄스 주파수를 측정할 카운터의 동작방식을 엣지, 업다운 카운트 및 카운트 소스등을 설정 / 반환합니다.

PARAMETER

▶ ***hDevice*** : DX_GnLoadDevice()로부터 얻어온 Device 의 핸들값입니다.

▶ ***Channel*** : Counter 채널 번호를 지정합니다. 채널 번호는 0 부터 시작합니다.

▶ ***Edge*** : 카운트 방법 입니다.

Value	Meaning
0 (dxCNT_FALLING)	하강엣지에서 카운트.
1 (dxCNT_RISING)	상승엣지에서 카운트.

▶ ***UpDown*** : 카운트 동작 방식 입니다

Value	Meaning
0 (dxCNT_UP)	업 카운트로 카운트.
1 (dxCNT_DOWN)	다운 카운트로 카운트.

▶ ***CntScr*** : 카운트 소스를 설정/반환합니다.

Value	Meaning
0 (dxCNT_PORTA)	카운터 포트 A 를 통하여 외부 소스를 사용하여 카운트.
1 (dxCNT_20MHZ)	내부 소스를 사용하여 카운트.

RETURN VALUE

□ 함수 수행 성공 여부.

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

.

SEE ALSO

DX_CntSetFilter, DX_CntGetFilter, DX_CntStart

NAME

DX_CntSetFilter / DX_CntGetFilter

- 카운터 채널의 필터를 설정/반환

INFORMATION

DX_CntSetFilter/
DX_CntGetFilter

VC++ (6, 7, 8)/VB
BCB/Delphi

Level 1

위험 요소 없음

SYNOPSIS

❒ VT_I4 DX_CntSetFilter ([in]VT_HNADLE hDevice, [in]VT_I4 Channel, [in]VT_I4 Frequency)

❒ VT_I4 DX_CntGetFilter ([in]VT_HNADLE hDevice, [in]VT_I4 Channel, [out]VT_PI4 Frequency)

DESCRIPTION

지정한 채널의 펄스 주파수를 측정할 카운터의 필터를 설정 / 반환합니다.

PARAMETER

▶ ***hDevice*** : DX_GnLoadDevice()로부터 얻어온 Device 의 핸들값입니다.

▶ ***Channel*** : Counter 채널 번호를 지정합니다. 채널 번호는 0 부터 시작합니다.

▶ ***Frequency*** : 카운트의 로우패스필터 주파수 입니다.

Value	Meaning
-1 (dxCNT_FREQ_NONE)	필터를 Disable 시킴.
0 (dxCNT_FREQ0)	2.5MHz 이하의 주파수만 통과.
1 (dxCNT_FREQ1)	1.25MHz 이하의 주파수만 통과.
2 (dxCNT_FREQ2)	625kHz 이하의 주파수만 통과.
3 (dxCNT_FREQ3)	312.5kHz 이하의 주파수만 통과.

RETURN VALUE

□ 함수 수행 성공 여부.

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

.

SEE ALSO

DX_CntSetConfig, DX_CntGetConfig, DX_CntStart

NAME

DX_CntSetDefault/DX_CntGetDefault

- 카운터 채널의 초기값을 설정/반환

INFORMATION

DX_CntSetDefault/
DX_CntGetDefault

VC++ (6, 7, 8)/VB
BCB/Delphi

Level 1

위험 요소 없음

SYNOPSIS

❒ VT_I4 DX_CntSetDefault ([in]VT_HNADLE hDevice, [in]VT_I4 Channel, [in]VT_I4 Count)

❒ VT_I4 DX_CntGetDefault ([in]VT_HNADLE hDevice, [in]VT_I4 Channel, [out]VT_PI4 Count)

DESCRIPTION

지정한 채널의 펄스 주파수를 측정할 카운터의 초기값를 설정 / 반환합니다.

PARAMETER

▶ ***hDevice*** : DX_GnLoadDevice()로부터 얻어온 Device 의 핸들값입니다.

▶ ***Channel*** : Counter 채널 번호를 지정합니다. 채널 번호는 0 부터 시작합니다.

▶ ***Count*** : 카운트의 초기값 입니다.

DX_CntSetDefault() 함수를 실행 하였을 경우

DX_CntSetDefault()함수를 통하여 이 값을 설정 하였을 경우 DX_CntGetCount() 함수로 현재 카운트 값을 읽었을 때 설정한 Count 값이 반환됩니다.

RETURN VALUE

□ 함수 수행 성공 여부.

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

0 (dxERR_NONE)	수행 성공

.

SEE ALSO

DX_CntSetClearMode, DX_CntGetClearMode, DX_CntClear

NAME

DX_CntSetClearMode/DX_CntGetClearMode

- 선택 채널의 펄스 주파수를 측정할 카운터 채널의 초기화모드를 설정/반환

INFORMATION

DX_CntSetClearMode/
DX_CntGetClearMode

VC++ (6, 7, 8)/VB
BCB/Delphi

Level 1

위험 요소 없음

SYNOPSIS

❒ VT_I4 DX_CntSetClearMode ([in]VT_HNADLE hDevice, [in]VT_I4 Channel, [in]VT_I4 ClrMode)

❒ VT_I4 DX_CntGetClearMode ([in]VT_HNADLE hDevice, [in]VT_I4 Channel, [out]VT_PI4 ClrMode)

DESCRIPTION

지정한 채널의 펄스 주파수를 측정할 카운터의 초기화모드를 설정/반환합니다.

PARAMETER

▶ ***hDevice*** : DX_GnLoadDevice()로부터 얻어온 Device 의 핸들값입니다.

▶ ***Channel*** : Counter 채널 번호를 지정합니다. 채널 번호는 0 부터 시작합니다.

▶ ***ClrMode*** : 카운트의 초기화모드 입니다.

Value	Meaning
0 (dxCMODE_DISABLE)	카운터 초기화를 하지 않음.
1 (dxCMODE_SRC_SOFT0)	카운터를 내부의 신호를 사용하여 DX_CntSetDefault()함수로 정의된 값으로 초기화함.
2 (dxCMODE_SRC_EXT0)	카운터를 외부의 트리거 신호를 사용하여 DX_CntSetDefault()함수로 정의된 값으로 초기화함.
3 (dxCMODE_SRC_SOFT1)	카운터를 내부의 신호를 사용하여 0 값으로 초기화함.
4 (dxCMODE_SRC_EXT0)	카운터를 외부의 트리거 신호를 사용하여 0 값으로 초기화함.

RETURN VALUE

□ 함수 수행 성공 여부.

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

.

SEE ALSO

DX_CntSetDefault, DX_CntGetDefault, DX_CntClear

NAME

DX_CntClear

- 선택 채널의 펄스 주파수를 측정할 카운터 채널의 초기화

INFORMATION

DX_CntClear

VC++ (6, 7, 8)/VB

BCB/Delphi

Level 1

위험 요소 없음

SYNOPSIS

❒ VT_I4 DX_CntClear ([in]VT_HNADLE hDevice, [in]VT_I4 Channel)

DESCRIPTION

지정한 채널의 펄스 주파수를 측정할 카운터를 초기화합니다.

PARAMETER

▶ ***hDevice*** : DX_GnLoadDevice()로부터 얻어온 Device 의 핸들값입니다.

▶ ***Channel*** : Counter 채널 번호를 지정합니다. 채널 번호는 0 부터 시작합니다.

RETURN VALUE

□ 함수 수행 성공 여부.

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

.

SEE ALSO

DX_CntSetClearMode, DX_CntGetClearMode, DX_CntSetDefault, DX_CntGetDefault

NAME

DX_CntStart / DX_CntStop

- 선택 채널의 펄스 주파수를 측정할 카운터 채널의 카운트 시작/종료

INFORMATION

DX_CntStart/ DX_CntStop

VC++ (6, 7, 8)/VB

BCB/Delphi

Level 1

위험 요소 없음

SYNOPSIS

- ❒ VT_I4 DX_CntStart ([in]VT_HNADLE hDevice, [in]VT_I4 Channel)
- ❒ VT_I4 DX_CntStop ([in]VT_HNADLE hDevice, [in]VT_I4 Channel)

DESCRIPTION

지정한 채널의 펄스 주파수를 측정할 카운터를 시작/종료합니다.

PARAMETER

▶ ***hDevice*** : DX_GnLoadDevice()로부터 얻어온 Device 의 핸들값입니다.

▶ ***Channel*** : Counter 채널 번호를 지정합니다. 채널 번호는 0 부터 시작합니다.

RETURN VALUE

□ 함수 수행 성공 여부.

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

.

SEE ALSO

DX_CntSetConfig, DX_CntGetConfig, DX_CntSetFilter, DX_CntGetFilter, DX_CntIsActive

EXAMPLE

□ CH 0 카운터 동작입니다.

[C / C++]

```
LONG GetCount = 0;

// Falling Edge, Up Count, 내부 소스 클락 설정
DX_CntSetConfig(hDevice, 0, dxCNT_FALLING, dxCNT_UP, dxCNT_20MHZ);

DX_CntSetFilter(hDevice, 0, dxCNT_FREQ_NONE); // 필터 Disable
DX_CntSetClearMode(hDevice, 0, dxCMODE_SRC_SOFT1); // Clear 시 0 으로 초기화 설정

DX_CntStart(hDevice, 0);

While(!IsStop()) // IsStop()은 가상의 함수임.
{
        DX_CntGetCount(hDevice, 0, &GetCount);

        // Data Process…
}

DX_CntStop(hDevice, 0);
DX_CntClear(hDevice, 0);
```

NAME

DX_CntIsActive

- 선택 채널의 펄스 주파수를 측정할 카운터 채널의 카운트 동작 확인

INFORMATION
DX_CntIsActive
VC++ (6, 7, 8)/VB
BCB/Delphi
Level 1
위험 요소 없음

SYNOPSIS

❒ VT_I4 DX_CntIsActive ([in]VT_HNADLE hDevice, [in]VT_I4 Channel, [out]VT_PI4 IsActive)

DESCRIPTION

지정한 채널의 펄스 주파수를 측정할 카운터의 동작을 확인합니다.

PARAMETER

▶ ***hDevice*** : DX_GnLoadDevice()로부터 얻어온 Device 의 핸들값입니다.

▶ ***Channel*** : Counter 채널 번호를 지정합니다. 채널 번호는 0 부터 시작합니다.

▶ ***IsActive*** : 지정한 카운터 채널의 동작상태을 반환합니다.

Value	Meaning
0 (dxFALSE)	Frequency Checker 동작 하지 않음.
1 (dxTRUE)	Frequency Checker 동작 하고 있음.

RETURN VALUE

□ 함수 수행 성공 여부.

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

.

SEE ALSO

DX_CntStart, DX_CntStop

NAME

DX_CntGetCount

- 선택 채널의 펄스 주파수를 측정할 카운터 채널의 카운트값 반환

INFORMATION
DX_CntGetCount
VC++ (6, 7, 8)/VB
BCB/Delphi
Level 1
위험 요소 없음

SYNOPSIS

❒ VT_I4 DX_CntGetCount ([in]VT_HNADLE hDevice, [in]VT_I4 Channel, [out]VT_PI4 Count)

DESCRIPTION

지정한 채널의 펄스 주파수를 측정할 카운터의 값을 반환합니다.

PARAMETER

▶ ***hDevice*** : DX_GnLoadDevice()로부터 얻어온 Device 의 핸들값입니다.

▶ ***Channel*** : Counter 채널 번호를 지정합니다. 채널 번호는 0 부터 시작합니다.

▶ ***Count*** : 지정한 카운터 채널의 값을 반환합니다.

RETURN VALUE

□ 함수 수행 성공 여부.

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

.

SEE ALSO

DX_CntStart, DX_CntStop

8.2 엔코더(Encoder) 카운터

이 단원에서 설명하는 카운터 기능은 COMI-DX501 보드에만 적용되는 카운터 기능으로서, 주로 엔코더 센서를 계측하는데 사용되는 카운터 기능입니다. 일반적으로 엔코더 센서는 회전체의 위치나 속도를 계측하기 위해 사용되는 센서입니다.
엔코더에서 발생되는 신호는 A 상, B 상 그리고 Z 상으로 구분되는 3 개의 신호이며 이들은 모두 펄스 형태로 출력됩니다. A 상과 B 상 신호는 회전체의 미소 위치 변화(엔코더의 분해능에 따라 그 값은 다름)가 발생할 때마다 발생되는 펄스 신호로써, 이 두 신호는 일정한 위상차를 가지게 되어 회전 방향도 함께 알 수 있습니다. COMI- DX501 은 회전 방향에 따라 자동으로 UP/DOWN 카운트 됩니다. Z 상 신호는 회전체가 1 회전할 때마다 발생되는 펄스 신호입니다. COMI- DX501 은 Z 상 신호를 이용하여 A/B 상 카운터의 값을 자동으로 초기화 하도록 설정할 수 있습니다. 이 방식을 이용하면 회전체의 절대 위치를 파악하기에 용이합니다.
COMI- DX501 보드는 엔코더의 3 가지 신호를 모두 입력 받을 수 있으며, A/B 상과 Z 상을 동시에 계측할 수 있습니다. 이 단원에서 소개되는 엔코더 카운터 함수 중에서 'Z'가 함수명 끝에 붙은 함수는 Z 상 신호의 계측에 사용되는 함수이며, 나머지 함수들은 모두 A/B 상 신호의 계측에 사용되는 함수입니다.

□ COMI-DX501 디바이스에서 일반 펄스를 카운트하는 방법

COMI- DX501 에서는 엔코더 신호뿐 아니라 일반 펄스 신호도 카운트할 수 있습니다. 일반 펄스 신호는 A 상과 B 상이 따로 존재하지 않으므로 신호선을 어떻게 연결해야 할지 혼동 될 수 있습니다. 일반 펄스 신호를 카운트하기 위해서는 펄스 신호를 터미널 보드의 A 상 입력단에 연결하면 됩니다. 사용되는 라이브러리 함수는 엔코더 신호를 카운트하는 것과 동일합니다.
한편, 일반 펄스 신호를 카운트할 때에도 B 상 입력단에 0 Volt 또는 5 Volt 를 인가하여 Up/down 카운트를 할 수 있습니다. B 상 입력단에 0 Volt 가 인가되면 업카운트(Up-count) 되고, 5 Volt 가 인가되면 다운카운트(Down-count) 됩니다. 만일 B 상 단자에 아무 신호도 연결되어 있지 않으면 업카운트를 하게 되며 펄스가 입력될 때마다 카운트는 증가하게 됩니다.

엔코더 카운터 기능과 관련된 함수 리스트 및 사용 가능 디바이스는 다음과 같습니다.

함수 명	사용 가능 디바이스
DX_EcSetConfig	COMI-DX501
DX_EcGetConfig	
DX_EcFilterEnable	
DX_EcFilterDisable	
DX_EcSetMaxCount	
DX_EcGetMaxCount	
DX_EcSetDefault	
DX_EcGetDefault	
DX_EcGetCount	
DX_EcZClearEnable	
DX_EcZClearDisable	
DX_EcZSetMaxCount	
DX_EcZGetMaxCount	
DX_EcZSetDefault	
DX_EcZGetDefault	
DX_EcZGetCount	
DX_EcStart	
DX_EcStop	
DX_EcIsActive	
DX_EcClear	

[표 8-3] 엔코더 카운터에 관련된 함수 리스트 및 사용 가능 디바이스 안내

8.2.1 함수 요약

엔코더 카운터와 관련된 함수들의 리스트는 다음과 같습니다.

Summary of Functions
❒ VT_I4 DX_EcSetConfig ([in] VT_HANDLE hDevice, [in] VT_I4 Channel, [in] VT_I4 CntrMode, [in] VT_I4 CntrMethod, [in] VT_I4 InvertMode) 지정한 엔코더 카운터 채널의 동작 방식을 설정합니다.
❒ VT_I4 DX_EcGetConfig ([in] VT_HANDLE hDevice, [in] VT_I4 Channel, [out] VT_PI4 CntrMode, [out] VT_PI4 CntrMethod, [out] VT_PI4 InvertMode) 지정한 엔코더 카운터 채널의 동작 방식을 반환합니다.
❒ VT_I4 DX_EcFilterEnable ([in] VT_HANDLE hDevice, [in] VT_I4 Channel, [in] VT_I4 FilterClk) 지정한 엔코더 카운터 채널의 필터 동작을 활성화합니다.
❒ VT_I4 DX_EcFilterDisable ([in] VT_HANDLE hDevice, [in] VT_I4 Channel) 지정한 엔코더 카운터 채널의 필터 동작을 비활성화합니다.
❒ VT_I4 DX_EcSetMaxCount ([in] VT_HANDLE hDevice, [in] VT_I4 Channel, [in] VT_I4 MaxCount) 지정한 엔코더 카운터 채널의 최대 제한값을 설정합니다.
❒ VT_I4 DX_EcGetMaxCount ([in] VT_HANDLE hDevice, [in] VT_I4 Channel, [out] VT_PI4 MaxCount) 지정한 엔코더 카운터 채널의 최대 제한값을 반환합니다.
❒ VT_I4 DX_EcSetDefault ([in] VT_HANDLE hDevice, [in] VT_I4 Channel, [in] VT_I4 Count) 지정한 엔코더 카운터 채널의 기본 카운터 위치값을 설정합니다.
❒ VT_I4 DX_EcGetDefault ([in] VT_HANDLE hDevice, [in] VT_I4 Channel, [out] VT_PI4 Count) 지정한 엔코더 카운터 채널의 기본 카운터 위치값을 반환합니다.
❒ VT_I4 DX_EcGetCount ([in] VT_HANDLE hDevice, [in] VT_I4 Channel, [out] VT_PI4 Count) 지정한 엔코더 카운터 채널의 현재 카운터 위치값을 반환합니다.
❒ VT_I4 DX_EcZClearEnable ([in] VT_HANDLE hDevice, [in] VT_I4 Channel) 지정한 엔코더 Z 상 카운터 채널을 초기화 가능하도록 설정합니다.
❒ VT_I4 DX_EcZClearDisable ([in] VT_HANDLE hDevice, [in] VT_I4 Channel) 지정한 엔코더 Z 상 카운터 채널을 초기화 불가능하도록 설정합니다.
❒ VT_I4 DX_EcZSetMaxCount ([in] VT_HANDLE hDevice, [in] VT_I4 Channel, [in] VT_I4 MaxCount) 지정한 엔코더 Z 상 카운터 채널의 최대 제한값을 설정합니다.
❒ VT_I4 DX_EcZGetMaxCount ([in] VT_HANDLE hDevice, [in] VT_I4 Channel, VT_ PI4 MaxCount) 지정한 엔코더 Z 상 카운터 채널의 최대 제한값을 반환합니다.

❒ VT_I4 DX_EcZSetDefault ([in] VT_HANDLE hDevice, [in] VT_I4 Channel, [in] VT_I4 Count)
지정한 엔코더 Z 상 카운터 채널의 기본 카운터 위치값을 설정합니다.

❒ VT_I4 DX_EcZGetDefault ([in] VT_HANDLE hDevice, [in] VT_I4 Channel, [out] VT_PI4 Count)
지정한 엔코더 Z 상 카운터 채널의 기본 카운터 위치값을 반환합니다.

❒ VT_I4 DX_EcZGetCount ([in] VT_HANDLE hDevice, [in] VT_I4 Channel, VT_ PI4 Count, [out] VT_PI4 MultiTurn)
지정한 엔코더 Z 상 카운터 채널의 카운터 위치값과 MultiTurn 값을 반환합니다.

❒ VT_I4 DX_EcStart ([in] VT_HANDLE hDevice, [in] VT_I4 Channel)
지정한 엔코더 카운터 채널의 측정을 시작합니다..

❒ VT_I4 DX_EcStop ([in] VT_HANDLE hDevice, [in] VT_I4 Channel)
지정한 엔코더 카운터 채널의 측정을 정지합니다.

❒ VT_I4 DX_EcIsActive ([in] VT_HANDLE hDevice, [in] VT_I4 Channel, [out] VT_PI4 IsActive)
지정한 엔코더 카운터 채널의 활성화 상태를 반환합니다.

❒ VT_I4 DX_EcClear ([in] VT_HANDLE hDevice, [in] VT_I4 Channel)
지정한 엔코더 카운터 채널의 카운트를 초기화합니다.

8.2.2 함수 설명

NAME

DX_EcSetConfig / DX_EcGetConfig

- 엔코더 카운터 채널의 동작 방식 설정/반환

INFORMATION
Encoder Counter
VC++ (6, 7, 8)/VB
BCB/Delphi
Level 1
위험 요소 없음

SYNOPSIS

❒ VT_I4 DX_EcSetConfig ([in] VT_HANDLE hDevice, [in] VT_I4 Channel, [in] VT_I4 CntrMode, [in] VT_I4 CntrMethod, [in] VT_I4 InvertMode)

❒ VT_I4 DX_EcGetConfig ([in] VT_HANDLE hDevice, [in] VT_I4 Channel, [out] VT_PI4 CntrMode, [out] VT_PI4 CntrMethod, [out] VT_PI4 InvertMode)

DESCRIPTION

이 함수는 지정한 엔코더 카운터 채널의 동작 방식을 설정/반환 합니다.

PARAMETER

▶ ***hDevice*** : DX_GnLoadDevice()로부터 얻어온 Device 의 핸들값입니다.

▶ ***Channel***: 엔코더 카운터 채널 번호를 지정합니다. 채널 번호는 0 부터 시작합니다.

▶ ***CntrMode***: 엔코더 A/B 상 카운터 채널의 입력 모드를 의미합니다.

Value	Meaning
0 (dxEC_x4AB)	4 채배 모드. 엔코더의 기본 분해능의 4 배 분해능으로 카운트 합니다.
1 (dxEC_x2AB)	2 체배 모드. 엔코더의 기본 분해능의 2 배 분해능으로 카운트 합니다.
2 (dxEC_x1AB)	1 체배 모드. 엔코더의 기본 분해능으로 카운트 합니다.
3 (dxEC_PLSDIR)	Pulse & Direction 모드. 각 상을 엔코더 입력과 방향 신호로 간주하여 카운트합니다.
4 (dxEC_CWCCW)	CW/CCW 모드. 각 상의 입력을 CW/CCW 입력으로 간주하여 카운트합니다.

▶ CntrMethod: 엔코더 A/B 상 카운터 채널의 카운트 증감 방식을 의미합니다.

Value	Meaning
0 (dxCNT_UP)	Up Count. 엔코더 카운터 입력에 따라 카운트 증가 시 값을 증가시킵니다.
1 (dxCNT_DOWN)	Douwn Count. 엔코더 카운터 입력에 따라 카운트 증가 시 값을 감소시킵니다.

▶ InvertMode: 엔코더 A/B 상 카운터 채널의 입력 반전 모드를 의미합니다.

Value	Meaning
0 (dxEC_INV_NONE)	어떠한 엔코더 상도 반전시키지 않습니다.
1 (dxEC_INV_A)	엔코더 A 상 입력 시에만 반전하여 측정합니다.
2 (dxEC_INV_B)	엔코더 B 상 입력 시에만 반전하여 측정합니다.
3 (dxEC_INV_AB)	엔코더 A 상과 B 상 입력 시 반전하여 측정합니다.
4 (dxEC_INV_Z)	엔코더 Z 상 입력 시에만 반전하여 측정합니다.
5 (dxEC_INV_AZ)	엔코더 A 상과 Z 상 입력 시 반전하여 측정합니다.
6 (dxEC_INV_BZ)	엔코더 B 상과 Z 상 입력 시 반전하여 측정합니다.
7 (dxEC_INV_ABZ)	엔코더 A/B/Z 상 모두 입력 시 반전하여 측정합니다.

RETURN VALUE

□ 함수 수행 성공 여부.

Value	Meaning
음수	함수 수행 실패
0 (dxERR_NONE)	함수 수행 성공.

NAME

DX_EcFilterEnable / DX_EcFilterDisable

- 엔코더 카운터 채널의 필터 동작 활성화 / 비활성화

INFORMATION

- Encoder Counter
- VC++ (6, 7, 8)/VB
- BCB/Delphi
- Level 1
- 위험 요소 없음

SYNOPSIS

❒ [in] VT_I4 DX_EcFilterEnable ([in] VT_HANDLE hDevice, [in] VT_I4 Channel, [in] VT_I4 FilterClk)

❒ [in] VT_I4 DX_EcFilterDisable ([in] VT_HANDLE hDevice, [in] VT_I4 Channel)

DESCRIPTION

이 함수는 지정한 엔코더 카운터 채널의 필터 동작을 활성화/비활성화 합니다.

PARAMETER

▶ ***hDevice*** : DX_GnLoadDevice()로부터 얻어온 Device 의 핸들값입니다.

▶ ***Channel***: 엔코더 카운터 채널 번호를 지정합니다. 채널 번호는 0 부터 시작합니다.

▶ ***FilterClk***: 엔코더 A/B 상 카운터 채널의 필터 모드를 의미합니다.

Value	Meaning
0 (dxEC_10M)	10M * 3 필터 모드.
1 (dxEC_5M)	5M * 3 필터 모드.

RETURN VALUE

□ 함수 수행 성공 여부.

Value	Meaning
음수	함수 수행 실패

0 (dxERR_NONE)	함수 수행 성공.

NAME

DX_EcSetMaxCount / DX_EcGetMaxCount

- 엔코더 카운터 채널의 최대 제한값 설정 / 반환

INFORMATION

- Encoder Counter
- VC++ (6, 7, 8)/VB
- BCB/Delphi
- Level 1
- 위험 요소 없음

SYNOPSIS

❒ [in] VT_I4 DX_EcSetMaxCount ([in] VT_HANDLE hDevice, [in] VT_I4 Channel, [in] VT_I4 MaxCount)

❒ [in] VT_I4 DX_EcGetMaxCount ([in] VT_HANDLE hDevice, [in] VT_I4 Channel, [out] VT_PI4 MaxCount)

DESCRIPTION

이 함수는 지정한 엔코더 카운터 채널의 카운트 최대 제한값을 설정/반환 합니다.

PARAMETER

▶ ***hDevice*** : DX_GnLoadDevice()로부터 얻어온 Device 의 핸들값입니다.

▶ ***Channel***: 엔코더 카운터 채널 번호를 지정합니다. 채널 번호는 0 부터 시작합니다.

▶ ***MaxCount***: 엔코더 카운터 최대 제한값을 의미합니다. 이 값을 초과한 경우 더 이상 카운트되지 않습니다.

RETURN VALUE

□ 함수 수행 성공 여부.

Value	Meaning
음수	함수 수행 실패
0 (dxERR_NONE)	함수 수행 성공.

NAME

DX_EcSetDefault / DX_EcGetDefault

- 엔코더 카운터 채널의 기본값 설정 / 반환

INFORMATION

Encoder Counter

VC++ (6, 7, 8)/VB

BCB/Delphi

Level 1

위험 요소 없음

SYNOPSIS

❒ [in] VT_I4 DX_EcSetDefault ([in] VT_HANDLE hDevice, [in] VT_I4 Channel, [in] VT_I4 Count)

❒ [in] VT_I4 DX_EcGetDefault ([in] VT_HANDLE hDevice, [in] VT_I4 Channel, [out] VT_PI4 Count)

DESCRIPTION

이 함수는 지정한 엔코더 카운터 채널의 기본 카운트 값을 설정/반환 합니다.

PARAMETER

▶ ***hDevice*** : DX_GnLoadDevice()로부터 얻어온 Device 의 핸들값입니다.

▶ ***Channel***: 엔코더 카운터 채널 번호를 지정합니다. 채널 번호는 0 부터 시작합니다.

▶ ***Count***: 엔코더 카운터 기본값을 의미합니다.

RETURN VALUE

□ 함수 수행 성공 여부.

Value	Meaning
음수	함수 수행 실패
0 (dxERR_NONE)	함수 수행 성공.

NAME

DX_EcGetCount

- 엔코더 카운터 채널의 현재 카운터 값 반환

INFORMATION

- Encoder Counter
- VC++ (6, 7, 8)/VB
- BCB/Delphi
- Level 1
- 위험 요소 없음

SYNOPSIS

❒ [in] VT_I4 DX_EcGetCount ([in] VT_HANDLE hDevice, [in] VT_I4 Channel, [out] VT_PI4 Count)

DESCRIPTION

이 함수는 지정한 엔코더 카운터 채널의 현재 카운트 값을 반환합니다.

PARAMETER

▶ ***hDevice*** : DX_GnLoadDevice()로부터 얻어온 Device 의 핸들값입니다.

▶ ***Channel***: 엔코더 카운터 채널 번호를 지정합니다. 채널 번호는 0 부터 시작합니다.

▶ ***Count***: 엔코더 카운터 현재 카운트 값을 반환합니다.

RETURN VALUE

□ 함수 수행 성공 여부.

Value	Meaning
음수	함수 수행 실패
0 (dxERR_NONE)	함수 수행 성공.

NAME

DX_EcZClearEnable / DX_EcZClearDisable

- 엔코더 Z 상 카운터 채널의 초기화 활성화/비활성화

INFORMATION

- Encoder Counter
- VC++ (6, 7, 8)/VB
- BCB/Delphi
- Level 1
- 위험 요소 없음

SYNOPSIS

- ❒ [in] VT_I4 DX_EcZClearEnable ([in] VT_HANDLE hDevice, [in] VT_I4 Channel)
- ❒ [in] VT_I4 DX_EcZClearDisable ([in] VT_HANDLE hDevice, [in] VT_I4 Channel)

DESCRIPTION

이 함수는 지정한 엔코더 Z 상 카운터 채널의 초기화가 가능/불가능 하도록 설정합니다.

PARAMETER

▶ ***hDevice*** : DX_GnLoadDevice()로부터 얻어온 Device 의 핸들값입니다.

▶ ***Channel***: 엔코더 카운터 채널 번호를 지정합니다. 채널 번호는 0 부터 시작합니다.

RETURN VALUE

□ 함수 수행 성공 여부.

Value	Meaning
음수	함수 수행 실패
0 (dxERR_NONE)	함수 수행 성공.

SEE ALSO

DX_EcClear

NAME

DX_EcZSetMaxCount / DX_EcZGetMaxCount

- 엔코더 Z 상 카운터 채널의 최대 제한값 설정 / 반환

INFORMATION

- Encoder Counter
- VC++ (6, 7, 8)/VB
- BCB/Delphi
- Level 1
- 위험 요소 없음

SYNOPSIS

❒ [in] VT_I4 DX_EcZSetMaxCount ([in] VT_HANDLE hDevice, [in] VT_I4 Channel, [in] VT_I4 MaxCount)

❒ [in] VT_I4 DX_EcZGetMaxCount ([in] VT_HANDLE hDevice, [in] VT_I4 Channel, [out] VT_PI4 MaxCount)

DESCRIPTION

이 함수는 지정한 엔코더 Z 상 카운터 채널의 카운트 최대 제한값을 설정/반환 합니다.

PARAMETER

▶ ***hDevice*** : DX_GnLoadDevice()로부터 얻어온 Device 의 핸들값입니다.

▶ ***Channel***: 엔코더 카운터 채널 번호를 지정합니다. 채널 번호는 0 부터 시작합니다.

▶ ***MaxCount***: 엔코더 Z 상 카운터 최대 제한값을 의미합니다. 이 값을 초과한 경우 더 이상 카운트되지 않습니다.

RETURN VALUE

□ 함수 수행 성공 여부.

Value	Meaning
음수	함수 수행 실패
0 (dxERR_NONE)	함수 수행 성공.

NAME

DX_EcZSetDefault / DX_EcZGetDefault

- 엔코더 Z 상 카운터 채널의 기본값 설정 / 반환

INFORMATION
Encoder Counter
VC++ (6, 7, 8)/VB
BCB/Delphi
Level 1
위험 요소 없음

SYNOPSIS

❒ [in] VT_I4 DX_EcZSetDefault ([in] VT_HANDLE hDevice, [in] VT_I4 Channel, [in] VT_I4 Count)

❒ [in] VT_I4 DX_EcZGetDefault ([in] VT_HANDLE hDevice, [in] VT_I4 Channel, [out] VT_PI4 Count)

DESCRIPTION

이 함수는 지정한 엔코더 Z 상 카운터 채널의 기본 카운트 값을 설정/반환 합니다.

PARAMETER

▶ ***hDevice*** : DX_GnLoadDevice()로부터 얻어온 Device 의 핸들값입니다.

▶ ***Channel***: 엔코더 카운터 채널 번호를 지정합니다. 채널 번호는 0 부터 시작합니다.

▶ ***Count***: 엔코더 Z 상 카운터 기본값을 의미합니다.

RETURN VALUE

□ 함수 수행 성공 여부.

Value	Meaning
음수	함수 수행 실패
0 (dxERR_NONE)	함수 수행 성공.

NAME

DX_EcZGetCount

- 엔코더 Z 상 카운터 채널의 카운트값 반환

INFORMATION

Encoder Counter

VC++ (6, 7, 8)/VB

BCB/Delphi

Level 1

위험 요소 없음

SYNOPSIS

❒ [in] VT_I4 DX_EcZGetCount ([in] VT_HANDLE hDevice, [in] VT_I4 Channel, [out] VT_PI4 Count, [out] VT_PI4 MultiTurn)

DESCRIPTION

이 함수는 지정한 엔코더 Z 상 카운터 채널의 카운트 값을 반환합니다.

PARAMETER

▶ ***hDevice*** : DX_GnLoadDevice()로부터 얻어온 Device 의 핸들값입니다.

▶ ***Channel***: 엔코더 카운터 채널 번호를 지정합니다. 채널 번호는 0 부터 시작합니다.

▶ ***Count***: 엔코더 Z 상 카운터의 카운트값을 반환합니다.

▶ ***MultiTurn***: 엔코더 Z 상 카운터의 카운트 MultiTurn 을 반환합니다.

RETURN VALUE

□ 함수 수행 성공 여부.

Value	Meaning
음수	함수 수행 실패
0 (dxERR_NONE)	함수 수행 성공.

NAME

DX_EcStart / DX_EcStop

- 엔코더 카운터 측정 시작/종료

INFORMATION

Encoder Counter

VC++ (6, 7, 8)/VB

BCB/Delphi

Level 1

위험 요소 없음

SYNOPSIS

❒ [in] VT_I4 DX_EcStart ([in] VT_HANDLE hDevice, [in] VT_I4 Channel)

DESCRIPTION

이 함수는 지정한 엔코더 카운터 채널의 카운트 측정을 시작/종료 합니다.

PARAMETER

▶ ***hDevice*** : DX_GnLoadDevice()로부터 얻어온 Device 의 핸들값입니다.

▶ ***Channel***: 엔코더 카운터 채널 번호를 지정합니다. 채널 번호는 0 부터 시작합니다.

RETURN VALUE

□ 함수 수행 성공 여부.

Value	Meaning
음수	함수 수행 실패
0 (dxERR_NONE)	함수 수행 성공.

EXAMPLE

□ 엔코더 카운터 0 번 채널로부터 A/B 상 과 Z 상의 값을 주기적으로 읽는 예제 입니다

[C / C++]

```
LONG GetCount = 0, ZCount = 0;
LONG MultiTurn = 0;

DX_EcSetConfig(hDevice, 0, dxEC_x4AB, dxCNT_UP, dxEC_INV_NONE);
```

```
DX_EcStart(hDevice, 0);

While( !IsStop() ) // IsStop()은 가상의 함수임.
{
        DX_EcGetCount(hDevice, 0, &GetCount);
        DX_EcZGetCount(hDevice, 0, &ZCount, & MultiTurn);

        // Data Process…
}

DX_EcStop(hDevice, 0);
DX_EcClear(hDevice, 0);
```

NAME

DX_EcIsActive

- 엔코더 카운터 측정 상태 반환

INFORMATION

Encoder Counter

VC++ (6, 7, 8)/VB

BCB/Delphi

Level 1

위험 요소 없음

SYNOPSIS

❒ [in] VT_I4 DX_EcIsActive ([in] VT_HANDLE hDevice, [in] VT_I4 Channel, [out] VT_PI4 IsActive)

DESCRIPTION

이 함수는 지정한 엔코더 카운터 채널의 카운트 측정 여부를 반환합니다.

PARAMETER

▶ ***hDevice*** : DX_GnLoadDevice()로부터 얻어온 Device 의 핸들값입니다.

▶ ***Channel***: 엔코더 카운터 채널 번호를 지정합니다. 채널 번호는 0 부터 시작합니다.

▶ ***IsActive***: 엔코더 카운터 측정 상태를 반환 합니다.

Value	Meaning
0 (dxFALSE)	엔코더 카운터 비활성화
1 (dxTRUE)	엔코더 카운터 활성화

RETURN VALUE

□ 함수 수행 성공 여부.

Value	Meaning
음수	함수 수행 실패
0 (dxERR_NONE)	함수 수행 성공.

NAME

DX_EcClear

- 엔코더 카운터 초기화

INFORMATION

Encoder Counter

VC++ (6, 7, 8)/VB

BCB/Delphi

Level 1

위험 요소 없음

SYNOPSIS

❒ [in] VT_I4 DX_EcClear ([in] VT_HANDLE hDevice, [in] VT_I4 Channel)

DESCRIPTION

이 함수는 지정한 엔코더 카운터 채널의 카운트를 초기화합니다.

PARAMETER

▶ ***hDevice*** : DX_GnLoadDevice()로부터 얻어온 Device 의 핸들값입니다.

▶ ***Channel***: 엔코더 카운터 채널 번호를 지정합니다. 채널 번호는 0 부터 시작합니다.

RETURN VALUE

□ 함수 수행 성공 여부.

Value	Meaning
음수	함수 수행 실패
0 (dxERR_NONE)	함수 수행 성공.

8.3 Pulse Generator

펄스 발생기(Pulse Generator) 기능은 사용자 가 설정한 주파수로 펄스를 발생시키는 기능입니다.

이 기능은 발생되는 펄스의 주파수를 제어할 수 있을 뿐 아니라 펄스의 수까지 제어할 수 있습니다. 이 기능은 주로 서보모터 및 스텝모터의 제어에 사용되는 기능입니다.

Pulse Generator 기능과 관련된 함수 리스트 및 사용 가능 디바이스는 다음과 같습니다.

함수 명	사용 가능 디바이스
DX_PgStart	COMI-DX10x, COMI-DX20x, COMI-DX 30x, COMI-DX 50x
DX_PgChangeFreq	
DX_PgIsActive	
DX_PgStop	

[표 8-4] Pulse Generator 에 관련된 함수 리스트 및 사용 가능 디바이스 안내

8.3.1 함수 요약

Pulse Generator 와 관련된 함수들의 리스트는 다음과 같습니다.

Summary of Functions
❒ VT_I4 DX_PgSetConfig ([in]VT_HANDLE hDevice, [in]VT_I4 Channel, [in]VT_R8 Frequency, [in]VT_I4 Duty, [in]VT_I4 Number) Pulse Generator 의 동작을 설정합니다.
❒ VT_I4 DX_PgGetConfig ([in]VT_HANDLE hDevice, [in]VT_I4 Channel, [out]VT_PR8 Frequency, [out]VT_PI4 Duty, [out]VT_PI4 Number) Pulse Generator 의 동작을 설정을 반환합니다.
❒ VT_I4 DX_PgSetInverse ([in]VT_HANDLE hDevice, [in]VT_I4 Channel, [in]VT_I4 Inverse) Pulse Generator 의 Inverse 을 설정합니다.
❒ VT_I4 DX_PgGetInverse ([in]VT_HANDLE hDevice, [in]VT_I4 Channel, [out]VT_PI4 Inverse) Pulse Generator 의 Inverse 을 반환합니다.
❒ VT_I4 DX_PgStart ([in]VT_HANDLE hDevice, [in]VT_I4 Channel) 선택 채널을 설정한 출력 주파수 및 펄스 수에 의해 펄스를 출력합니다.
❒ VT_I4 DX_PgStop ([in]VT_HANDLE hDevice, [in]VT_I4 Channel) 펄스 출력을 중지합니다.
❒ VT_I4 DX_PgIsActive ([in]VT_HANDLE hDevice, [in]VT_I4 Channel, [out]IsActive) 현재 펄스가 출력되고 있는지 상태를 반환합니다.

8.3.2 함수 설명

NAME

DX_PgSetConfig / DX_PgGetConfig

- 선택한 채널에 Pulse Generator 의 동작 설정 / 반환

INFORMATION

- Pulse Generator
- VC++ (6, 7, 8)/VB
- BCB/Delphi
- Level 1
- 위험 요소 없음

SYNOPSIS

❒ VT_I4 DX_PgSetConfig ([in]VT_HANDLE hDevice, [in]VT_I4 Channel, [in]VT_R8 Frequency, [in]VT_I4 Duty, [in]VT_I4 Number)

❒ VT_I4 DX_PgGetConfig ([in]VT_HANDLE hDevice, [in]VT_I4 Channel, [out]VT_PR8 Frequency, [out]VT_PI4 Duty, [out]VT_PI4 Number)

DESCRIPTION

이 함수는 지정한 Pulse Generator 채널을 통해 지정한 주파수, Duty 및 펄스 수를 설정 / 반환합니다.

PARAMETER

▶ ***hDevice*** : DX_GnLoadDevice()로부터 얻어온 Device 의 핸들값입니다.

▶ ***Channel*** : Pulse Generator 채널 번호를 지정합니다. 채널 번호는 0 부터 시작합니다.

▶ ***Frequency*** : 출력할 펄스의 주파수를 Hz 단위로 설정합니다.

▶ ***Duty*** : 출력할 펄스의 듀티비를 %단위로 설정합니다.

▶ ***Number*** : 출력할 펄스의 수를 지정합니다.

Value	Meaning
0	DX_PgStop() 함수가 호출되기 전까지 계속해서 펄스를 출력합니다.
양수	설정한 수 만큼 펄스를 출력합니다.

RETURN VALUE

□ 함수 수행 성공 여부.

Value	Meaning

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

SEE ALSO

DX_PgGetConfig, DX_PgIsActive, DX_PgStop

NAME

DX_PgSetInverse / DX_PgGetInverse

- Pulse Generator 의 Inverse 을 설정 / 반환

INFORMATION

- Pulse Generator
- VC++ (6, 7, 8)/VB
- BCB/Delphi
- Level 1
- 위험 요소 없음

SYNOPSIS

❒ VT_I4 DX_PgSetInverse ([in]VT_HANDLE hDevice, [in]VT_I4 Channel, [in]VT_I4 Inverse)

❒ VT_I4 DX_PgGetInverse ([in]VT_HANDLE hDevice, [in]VT_I4 Channel, [out]VT_PI4 Inverse)

DESCRIPTION

이 함수는 지정한 Pulse Generator 채널의 Inverse 값을 설정 / 반환 합니다.

PARAMETER

▶ ***hDevice*** : DX_GnLoadDevice()로부터 얻어온 Device 의 핸들값입니다.

▶ ***Channel*** : Pulse Generator 채널 번호를 지정합니다. 채널 번호는 0 부터 시작합니다.

▶ ***Inverse*** : 출력할 펄스의 Inverse 여부 입니다.

RETURN VALUE

□ 함수 수행 성공 여부.

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

SEE ALSO

DX_PgSetConfig, DX_PgGetInverse

NAME

DX_PgStart

- 설정한 출력 주파수, Duty 및 펄스 수에 의해 펄스 출력

INFORMATION

- Pulse Generator
- VC++ (6, 7, 8)/VB
- BCB/Delphi
- Level 1
- 위험 요소 없음

SYNOPSIS

❒ VT_I4 DX_PgStart ([in]VT_HANDLE hDevice, [in]VT_I4 Channel)

DESCRIPTION

이 함수는 지정한 Pulse Generator 채널을 통해 지정한 주파수, Duty 및 펄스 수에 의해 펄스 출력합니다.

PARAMETER

▶ ***hDevice*** : DX_GnLoadDevice()로부터 얻어온 Device 의 핸들값입니다.

▶ ***Channel*** : Pulse Generator 채널 번호를 지정합니다. 채널 번호는 0 부터 시작합니다.

RETURN VALUE

□ 함수 수행 성공 여부.

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

SEE ALSO

DX_PgSetConfig, DX_PgSetInverse, DX_PgStop

NAME

DX_PgStop

- 펄스 출력 중지

INFORMATION

Pulse Generator

VC++ (6, 7, 8)/VB

BCB/Delphi

Level 1

위험 요소 없음

SYNOPSIS

❒ VT_I4 DX_PgStop ([in]VT_HANDLE hDevice, [in]VT_I4 Channel)

DESCRIPTION

이 함수는 지정한 Pulse Generator 채널의 펄스 출력을 중지합니다.

PARAMETER

▶ ***hDevice*** : DX_GnLoadDevice()로부터 얻어온 Device 의 핸들값입니다.

▶ ***Channel*** : Pulse Generator 채널 번호를 지정합니다. 채널 번호는 0 부터 시작합니다.

RETURN VALUE

□ 함수 수행 성공 여부.

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

SEE ALSO

DX_PgStart, DX_PgIsActive

NAME

DX_PgIsActive

- 펄스 출력 상태를 반환

INFORMATION

- Pulse Generator
- VC++ (6, 7, 8)/VB
- BCB/Delphi
- Level 1
- 위험 요소 없음

SYNOPSIS

❒ VT_I4 DX_PgIsActive ([in]VT_HANDLE hDevice, [in]VT_I4 Channel, [out]VT_PI4 IsActive)

DESCRIPTION

지정한 Pulse Generator 채널에 현재 펄스가 출력되고 있는지 상태를 반환합니다. 이 함수를 이용하면 출력 펄스 수를 제한한 경우에 원하는 펄스의 출력이 완료되었는지 확인할 수 있습니다.

PARAMETER

▶ ***hDevice*** : DX_GnLoadDevice()로부터 얻어온 Device 의 핸들값입니다.

▶ ***Channel*** : Pulse Generator 채널 번호를 지정합니다. 채널 번호는 0 부터 시작합니다.

▶ ***IsActive*** : Pulse Generator 의 실행 상태를 반환합니다.

Value	Meaning
0 (dxFALSE)	현재 펄스가 출력 되고 있지 않음.
1 (dxTRUE)	현재 펄스가 출력 되고 있음.

RETURN VALUE

□ 현재 Pluse Generator 동작 확인

Value	Meaning
0 (dxFALSE)	Pulse Generator 동작 하지 않음.
1 (dxTRUE)	Pulse Generator 동작 하고 있음.

.

SEE ALSO

DX_PgStart, DX_PgChangeFreq, DX_PgStop

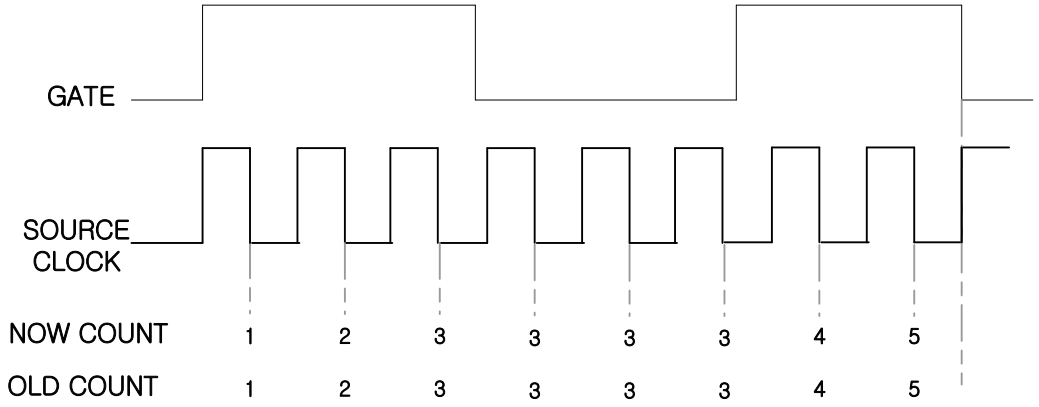
8.4 Frequency Checker

Frequency Checker 는 다음과 같이 세 가지 모드로 동작합니다.

■ MODE 0

이 모드는 일반적인 카운터 모드입니다. GATE 신호가 High 상태로 유지되는 동안에 소스 클럭(Source Clock)단에 입력되는 펄스의 수를 카운트합니다. 이 모드에서는 GATE 신호가 카운트 값을 자동으로 0 으로 초기화하지 않습니다. 그리고 GATE 신호에 아무 신호도 연결되지 않으면 GATE 신호는 자동적으로 High 상태로 유지됩니다.

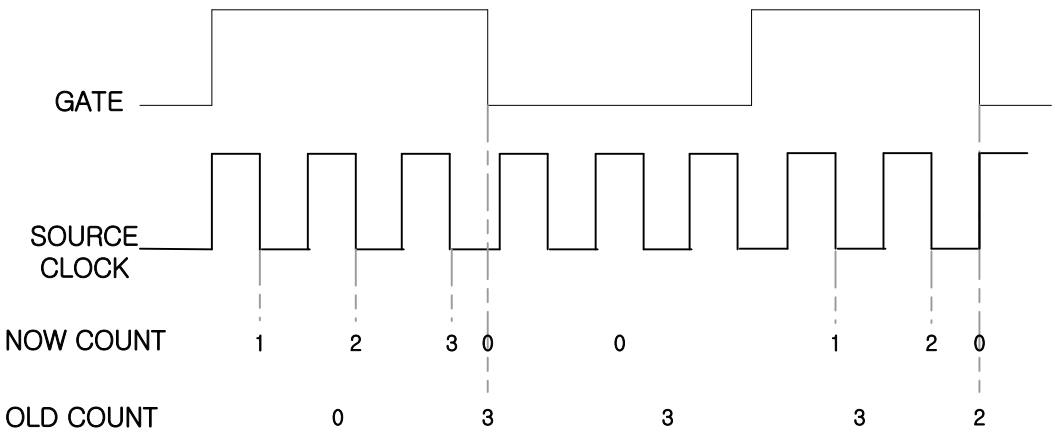
MODE 0 에서 카운터의 동작을 그림으로 표현하면 다음과 같습니다.



■ MODE 1

이 모드는 GATE 단자에 입력되는 펄스의 상태가 High 로 유지되는 시간을 계측하기 위해 주로 사용되는 모드입니다. 이 모드에서는 GATE 신호가 High 상태로 유지되는 동안에 소스 클럭(Source Clock)단에 입력되는 펄스의 수를 카운트하는 것은 MODE 0 과 동일합니다. 그러나 이 모드에서는 GATE 신호의 하강 에지(Falling Edge)에서 현재의 카운트(Now count) 값을 Latch 에 저장(Old count) 한 후 0 으로 초기화하고, 다시 카운트를 시작합니다. 따라서 Old count 값을 참조하면 GATE 신호에 입력되는 펄스 신호의 주기 또는 주파수를 알 수 있습니다.

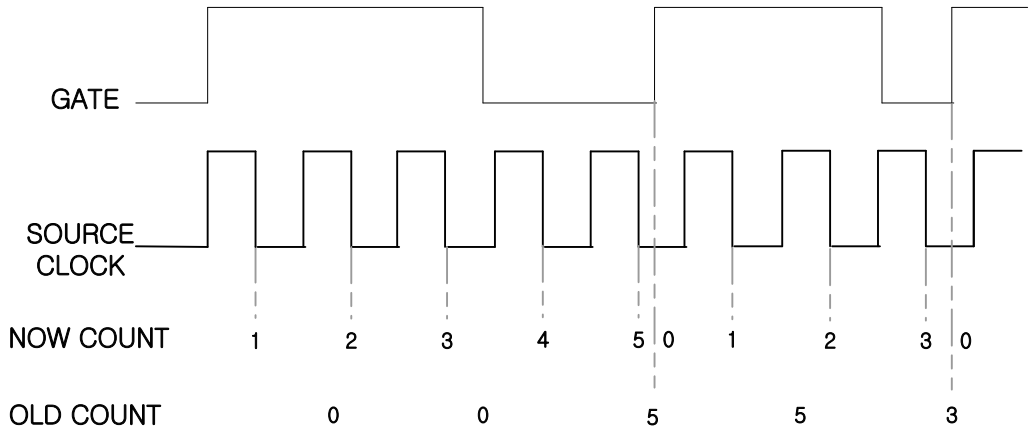
MODE 1 에서 카운터의 동작을 그림으로 표현하면 다음과 같습니다.



■ MODE 2

이 모드는 펄스의 주파수를 측정하는데 용이하게 사용될 수 있는 모드로써 GATE 신호의 상승에지(Rising Edge)가 새로운 카운트 시작을 트리거하고, 동시에 이전의 카운트 값을 Old Count 에 래치(Latch)합니다. 이 모드에서는 GATE 신호가 Low 상태일 때에도 카운트는 계속됩니다.

MODE 2 에서 카운터의 동작을 그림으로 표현하면 다음과 같습니다.



Frequency Checker 기능과 관련된 함수 리스트 및 사용 가능 디바이스는 다음과 같습니다.

함수 명	사용 가능 디바이스
DX_FcSetCounter	COMI-DX10x, COMI-DX20x, COMI-DX 30x, COMI-DX 50x
DX_FcStart	
DX_FcStop	
DX_FcGetFrequency	
DX_FcIsActive	

[표 8-5] Frequency Checker 에 관련된 함수 리스트 및 사용 가능 디바이스 안내

8.4.1 함수 요약

Frequency Checker 기능과 관련된 함수들의 리스트는 다음과 같습니다.

Summary of Functions
❒ VT_I4 DX_FcSetCounter ([in] VT_HANDLE hDevice, [in]VT_I4 CheckTime) 펄스 주파수를 측정할 주기을 설정합니다.
❒ VT_I4 DX_FcStart ([in] VT_HANDLE hDevice) 펄스 주파수를 측정을 시작합니다.
❒ VT_I4 DX_FcStop ([in] VT_HANDLE hDevice) 펄스 주파수를 측정을 정지합니다.
❒ VT_I4 DX_FcGetFrequency ([in] VT_HANDLE hDevice, [in]VT_I4 Channel, [out]VT_PR8 Frequency) 펄스 주파수를 측정하여 반환합니다.
❒ VT_I4 DX_FcIsActive ([in] VT_HANDLE hDevice, [out]VT_PI4 IsActive) 펄스 주파수를 측정중인지 확인하여 반환합니다.

8.4.2 함수 설명

NAME

DX_FcSetCounter

- 펄스 주파수를 측정할 주기율 설정

INFORMATION

Frequency Checker

VC++ (6, 7, 8)/VB

BCB/Delphi

Level 1

위험 요소 없음

SYNOPSIS

❒ VT_I4 DX_FcSetCounter ([in] VT_HANDLE hDevice, [in]VT_I4 CheckTime)

DESCRIPTION

이 함수는 펄스 주파수를 측정할 주기을 설정합니다.

PARAMETER

▶ ***hDevice*** : DX_GnLoadDevice()로부터 얻어온 Device 의 핸들값입니다.

▶ ***CheckTime*** : Frequency Checker 의 동작 주기를 설정합니다. (단위는 ms 입니다.)

RETURN VALUE

□ 함수 수행 성공 여부.

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

NAME

DX_FcStart

- Frequency Checker 시작

INFORMATION

Frequency Checker

VC++ (6, 7, 8)/VB

BCB/Delphi

Level 1

위험 요소 없음

SYNOPSIS

❒ VT_I4 DX_FcStart ([in] VT_HANDLE hDevice)

DESCRIPTION

이 함수는 펄스 주파수 측정을 시작합니다.

PARAMETER

▶ ***hDevice*** : DX_GnLoadDevice()로부터 얻어온 Device 의 핸들값입니다.

RETURN VALUE

□ 함수 수행 성공 여부.

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

.

SEE ALSO

DX_FcStop

EXAMPLE

□ CheckTime 을 10ms 로 설정한 후 0 번 채널로 입력되는 펄스의 주파수를 주기적으로 표시 하는 예제입니다.

[C / C++]

NAME

DX_FcStop

- Frequency Checker 종료

INFORMATION

Frequency Checker

VC++ (6, 7, 8)/VB

BCB/Delphi

Level 1

위험 요소 없음

SYNOPSIS

❒ VT_I4 DX_FcStop ([in] VT_HANDLE hDevice)

DESCRIPTION

이 함수는 펄스 주파수 측정을 종료합니다.

PARAMETER

▶ ***hDevice*** : DX_GnLoadDevice()로부터 얻어온 Device 의 핸들값입니다.

RETURN VALUE

□ 함수 수행 성공 여부.

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

.

SEE ALSO

DX_FcStart

NAME

DX_FcGetFrequency

- 펄스 주파수를 측정하여 반환

INFORMATION

Frequency Checker

VC++ (6, 7, 8)/VB

BCB/Delphi

Level 1

위험 요소 없음

SYNOPSIS

❒ VT_I4 DX_FcGetFrequency ([in]VT_HANDLE hDevice, [in]VT_I4 Channel, [out]VT_PR8 Frequency)

DESCRIPTION

이 함수는 선택 채널의 펄스 주파수를 측정하여 반환합니다.

PARAMETER

▶ ***hDevice*** : DX_GnLoadDevice()로부터 얻어온 Device 의 핸들값입니다.

▶ ***Channel*** : Frequency Checker 채널 번호를 지정합니다. 채널 번호는 0 부터 시작합니다.

▶ ***Frequency*** : 현재 선택된 채널의 펄스 주파수값을 반환합니다.

RETURN VALUE

□ 함수 수행 성공 여부.

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

.

NAME

DX_FcIsActive

- Frequency Checker 가 동작중인지 확인하여 반환

INFORMATION

Frequency Checker

VC++ (6, 7, 8)/VB

BCB/Delphi

Level 1

위험 요소 없음

SYNOPSIS

❒ VT_I4 DX_FcIsActive ([in]VT_HANDLE hDevice, [out]VT_PI4 IsActive)

DESCRIPTION

이 함수는 Frequency Checker 가 실행중인지 확인하여 반환합니다.

PARAMETER

▶ ***hDevice*** : DX_GnLoadDevice()로부터 얻어온 Device 의 핸들값입니다.

▶ ***IsActive*** : Frequency Checker 가 동작 중인지 확인하여 반환합니다..

Value	Meaning
0 (dxFALSE)	Frequency Checker 동작 하지 않음.
1 (dxTRUE)	Frequency Checker 동작 하고 있음.

RETURN VALUE

□ 함수 수행 성공 여부.

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

.

8.5 위치값 래치 (Position Latch)

Position Latch 는 특정 순간에 Motion 의 위치 관련 카운터값을 래치(Latch)하여 읽을 수 있도록하는 기능입니다. Position Latch 는 LTC 입력핀에 LATCH 신호가 입력되면 그 순간의 각 카운트값을 래치(Latch)합니다. 이 기능은 LTC 핀에 입력되는 하드웨어 신호에 동기되어서 각 위치값이 래치되므로 시간 지연이 거의 발생하지 않습니다.
사용자는 DX_LtcStart () 함수를 이용하여 래치를 시작한 후, 래치가 되었으면 DX_LtcReadOneLatch() 또는 DX_LtcReadAllLatch() 함수를 이용하여 래치된 위치 카운트값을 읽을 수 있습니다.

위치값 래치 기능과 관련된 함수 리스트 및 사용 가능 디바이스는 다음과 같습니다.

함수 명	사용 가능 디바이스
DX_LtcSetMode	COMI-DX501
DX_LtcGetMode	
DX_LtcStart	
DX_LtcStop	
DX_LtcIsActive	
DX_LtcClear	
DX_LtcGetDataCount	
DX_LtcReadOneLatch	
DX_LtcReadAllLatch	

[표 8-3] 위치갑 래치에 관련된 함수 리스트 및 사용 가능 디바이스 안내

8.5.1 함수 요약

Position Latch 카운터기능과 관련된 함수들의 리스트는 다음과 같습니다.

Summary of Functions
❒ VT_I4 DX_LtcSetMode ([in] VT_HANDLE hDevice, [in]VT_I4 Channel, [in]VT_I4 LtcMode) Position Latch 모드를 설정합니다.
❒ VT_I4 DX_LtcGetMode ([in] VT_HANDLE hDevice, [in]VT_I4 Channel, [out]VT_PI4 LtcMode) Position Latch 모드를 반환합니다
❒ VT_I4 DX_LtcStart ([in] VT_HANDLE hDevice, [in]VT_I4 Channel) Position Latch 동작을 시작합니다.
❒ VT_I4 DX_LtcStop ([in] VT_HANDLE hDevice, [in]VT_I4 Channel) Position Latch 동작을 종료합니다.
❒ VT_I4 DX_LtcIsActive ([in] VT_HANDLE hDevice, [in]VT_I4 Channel, [out]VT_PI4 IsActive) Position Latch 의동작 상태를 반환합니다.
❒ VT_I4 DX_LtcClear ([in] VT_HANDLE hDevice, [in]VT_I4 Channel) Position Latch 기능을 초기화 합니다.
❒ VT_I4 DX_LtcGetDataCount ([in] VT_HANDLE hDevice, [in]VT_I4 Channel, [out]VT_PI4 DataCount) Position Latch 를 통하여 램에 저장된 데이터 개수를 반환합니다.
❒ VT_I4 DX_LtcReadOneLatch ([in] VT_HANDLE hDevice, [in]VT_I4 Channel, [in]VT_I4 DataIdx, [out]VT_PI4 Data) Position Latch 를 통하여 램에 저장된 하나의 데이터를 반환합니다.
❒ VT_I4 DX_LtcReadAllLatch([in] VT_HANDLE hDevice, [in]VT_I4 Channel, [in]VT_I4 StartCnt, [in]VT_I4 BufSize, [out]VT_PI4 DataList) Position Latch 를 통하여 램에 저장된 여러 개의 데이터를 버퍼에 반환합니다.

8.5.2 함수 설명

NAME

DX_LtcSetMode / DX_LtcGetMode

- Pisiton Latch 모드 설정 / 반환

INFORMATION

- Position Latch
- VC++ (6, 7, 8)/VB
- BCB/Delphi
- Level 1
- 위험 요소 없음

SYNOPSIS

❒ VT_I4 DX_LtcSetMode ([in] VT_HANDLE hDevice, [in]VT_I4 Channel, [in]VT_I4 LtcMode)

❒ VT_I4 DX_LtcGetMode ([in] VT_HANDLE hDevice, [in]VT_I4 Channel, [out]VT_PI4 LtcMode)

DESCRIPTION

이 함수는 래치카운터의 모드를 설정 / 반환 합니다.

PARAMETER

▶ ***hDevice*** : DX_GnLoadDevice()로부터 얻어온 Device 의 핸들값입니다.

▶ ***Channel*** : Latch 카운터의 채널 번호를 지정합니다. 채널 번호는 0 부터 시작합니다.

▶ ***LtcMode*** : Latch 카운터의 동작 모드 입니다.

Value	Meaning
0 (dxLTC_NONE)	Latch 카운터의 동작을 Disable 합니다.
1 (dxLTC_R_EDGE)	Latch 카운터의 Rising edge 동작 모드 입니다..
2 (dxLTC_F_EDGE)	Latch 카운터의 Falling edge 동작 모드 입니다.
3 (dxLTC_BI_EDGE)	Latch 카운터의 Rising edge 와 Falling edge 동작 모드 입니다.

RETURN VALUE

□ 함수 수행 성공 여부.

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

NAME

DX_LtcStart / DX_LtcStop

- Pisiton Latch 동작 시작 / 종료

INFORMATION

- Position Latch
- VC++ (6, 7, 8)/VB
- BCB/Delphi
- Level 1
- 위험 요소 없음

SYNOPSIS

- ❒ VT_I4 DX_LtcStart ([in] VT_HANDLE hDevice, [in]VT_I4 Channel)
- ❒ VT_I4 DX_LtcStop ([in] VT_HANDLE hDevice, [in]VT_I4 Channel)

DESCRIPTION

이 함수는 래치카운터를 시작 / 종료 합니다.

PARAMETER

▶ ***hDevice*** : DX_GnLoadDevice()로부터 얻어온 Device 의 핸들값입니다.

▶ ***Channel*** : Latch 카운터의 채널 번호를 지정합니다. 채널 번호는 0 부터 시작합니다.

RETURN VALUE

□ 함수 수행 성공 여부.

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

NAME

DX_LtcIsActive

- Pisiton Latch 동작 상태 반환

INFORMATION

Position Latch

VC++ (6, 7, 8)/VB

BCB/Delphi

Level 1

위험 요소 없음

SYNOPSIS

❒ VT_I4 DX_LtcIsActive ([in] VT_HANDLE hDevice, [in]VT_I4 Channel, [out]VT_PI4 IsActive)

DESCRIPTION

이 함수는 래치카운터의 동작 상태를 반환합니다.

PARAMETER

▶ ***hDevice*** : DX_GnLoadDevice()로부터 얻어온 Device 의 핸들값입니다.

▶ ***Channel*** : Latch 카운터의 채널 번호를 지정합니다. 채널 번호는 0 부터 시작합니다.

▶ ***IsActive*** : Latch 카운터의 동작 상태입니다.

Value	Meaning
0 (dxFALSE)	Latch 카운터 동작 중
1 (dxTRUE)	Latch 카운터가 동작 중이지 않음.

RETURN VALUE

□ 함수 수행 성공 여부.

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

NAME

DX_LtcClear

- Pisiton Latch 초기화

INFORMATION

- Position Latch
- VC++ (6, 7, 8)/VB
- BCB/Delphi
- Level 1
- 위험 요소 없음

SYNOPSIS

❒ VT_I4 DX_LtcClear ([in] VT_HANDLE hDevice, [in]VT_I4 Channel)

DESCRIPTION

이 함수는 대상 디바이스의 카운터 채널에 대하여 래치 기능을 초기화 합니다.

PARAMETER

▶ ***hDevice*** : DX_GnLoadDevice()로부터 얻어온 Device 의 핸들값입니다.

▶ ***Channel*** : Latch 카운터의 채널 번호를 지정합니다. 채널 번호는 0 부터 시작합니다.

RETURN VALUE

□ 함수 수행 성공 여부.

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

NAME

DX_LtcGetDataCount

- Latch 카운터 데이터 개수 반환

INFORMATION

Position Latch

VC++ (6, 7, 8)/VB

BCB/Delphi

Level 1

위험 요소 없음

SYNOPSIS

❒ VT_I4 DX_LtcGetDataCount ([in] VT_HANDLE hDevice, [in]VT_I4 Channel, [out]VT_PI4 DataCount)

DESCRIPTION

이 함수는 대상 디바이스의 래치카운터 채널 메모리에 저장 된 데이터의 개수를 반환합니다.

PARAMETER

▶ ***hDevice*** : DX_GnLoadDevice()로부터 얻어온 Device 의 핸들값입니다.

▶ ***Channel*** : Latch 카운터의 채널 번호를 지정합니다. 채널 번호는 0 부터 시작합니다.

▶ ***DataCount*** : Latch 카운터에 저장된 데이터의 개수입니다.

RETURN VALUE

□ 함수 수행 성공 여부.

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

NAME

DX_LtcReadOneLatch

- Latch 카운터의 하나의 데이터 반환

INFORMATION

- Position Latch
- VC++ (6, 7, 8)/VB
- BCB/Delphi
- Level 1
- 위험 요소 없음

SYNOPSIS

❒ VT_I4 DX_LtcReadOneLatch ([in] VT_HANDLE hDevice, [in]VT_I4 Channel, [in]VT_I4 DataIdx, [out]VT_PI4 Data)

DESCRIPTION

이 함수는 대상 디바이스의 래치카운터 채널 메모리에 저장되어 있는 데이터 중 DataIdx 의 위치에 있는 데이터 값을 반환합니다.

PARAMETER

▶ ***hDevice*** : DX_GnLoadDevice()로부터 얻어온 Device 의 핸들값입니다.

▶ ***Channel*** : Latch 카운터의 채널 번호를 지정합니다. 채널 번호는 0 부터 시작합니다.

▶ ***DataIdx***: Latch 카운터 램에서 가져올 데이터의 인덱스입니다. 이 값은 512 보다 작아야 합니다.

주의	래치 카운터의 각 채널 램의 크기는 512bye 이며, 이 램은 환형 큐 메모리 방식으로 데이터가 저장됩니다. .

▶ ***Data***: Latch 카운터 램의 DataIdx 의 위치에 있는 데이터 값 입니다.

RETURN VALUE

□ 함수 수행 성공 여부.

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

NAME

DX_LtcReadAllLatch

- Latch 카운터의 여러개의 데이터 반환

INFORMATION

Position Latch

VC++ (6, 7, 8)/VB

BCB/Delphi

Level 1

위험 요소 없음

SYNOPSIS

❒ VT_I4 DX_LtcReadAllLatch([in] VT_HANDLE hDevice, [in]VT_I4 Channel, [in]VT_I4 StartCnt, [in]VT_I4 BufSize, [out]VT_PI4 DataList)

DESCRIPTION

이 함수는 대상 디바이스의 래치카운터 채널 메모리에 저장되어 있는 데이터 중 StartCnt 위치부터 BufSize 만큼의 Data 를 반환합니다.

PARAMETER

▶ ***hDevice*** : DX_GnLoadDevice()로부터 얻어온 Device 의 핸들값입니다.

▶ ***Channel*** : Latch 카운터의 채널 번호를 지정합니다. 채널 번호는 0 부터 시작합니다.

▶ ***StartCnt*** : Latch 카운터 램에서 연속적인 데이터를 전달 받기 위한 시작 위치 입니다.

▶ ***BufSize*** : Latch 카운터 램에서 연속적인 데이터를 전달 받기 위한 버퍼의 크기 입니다..

▶ ***DataList***: Latch 카운터 램에서 가져올 데이터를 전달 받을 버퍼를 지정합니다. 이 버퍼의 크기는 BufSize 에서 지정한 값보다 크거나 같아야 합니다.

RETURN VALUE

□ 함수 수행 성공 여부.

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

8.6 위치비교출력(CMP)

COMI-DX50x Series 디바이스는 각 채널 마다 위치비교 출력 기능을 제공합니다. 위치비교 출력 기능은 Encoder Counter 의 카운트 값이 사용자가 지정한 조건에 만족되면 CMP 출력핀을 통하여 트리거 펄스를 출력해주는 기능입니다.

일정 구간 비교 모드에서 DX_CmpSetRegularData() 함수를 사용하면 연속된 일정 포인트에서 위치비교 출력이 가능합니다.
임의 구간 비교 모드에서 DX_CmpSetRandomData() 함수를 사용하면 하나의 비교데이터 또는 여러 개의 비교데이터를 설정하여 각 포인트에서 위치비교 출력을 할 수 있습니다.

위치비교 출력 기능과 관련된 함수 리스트 및 사용 가능 디바이스는 다음과 같습니다.

함수 명	사용 가능 디바이스
DX_CmpSetConfig	COMI-DX501
DX_CmpGetConfig	
DX_CmpSetRegularData	
DX_CmpGetRegularData	
DX_CmpSetRandomData	
DX_CmpGetRandomData	
DX_CmpStart	
DX_CmpStop	
DX_CmpIsActive	
DX_CmpClear	

[표 8-3] 위치비교출력기에 관련된 함수 리스트 및 사용 가능 디바이스 안내

8.6.1 함수 요약

위치비교 출력 기능과 관련된 함수들의 리스트는 다음과 같습니다.

Summary of Functions
❒ VT_I4 DX_CmpSetConfig([in] VT_HANDLE hDevice, [in]VT_I4 Channel, [in]VT_I4 CmpMode, [in]VT_I4 CmpMethod, [in]VT_I4 CmpInv, [in]VT_I4 PlsWidth) 위치비교 출력 기능을 담당하는 비교기의 환경설정을 구성합니다.
❒ VT_I4 DX_CmpGetConfig([in] VT_HANDLE hDevice, [in]VT_I4 Channel, [out]VT_PI4 CmpMode, [out]VT_PI4 CmpMethod, [out]VT_PI4 CmpInv, [out]VT_PI4 PlsWidth) 위치비교 출력 기능을 담당하는 비교기의 환경설정을 반환합니다.
❒ VT_I4 DX_CmpSetRegularData([in] VT_HANDLE hDevice, [in]VT_I4 Channel, [in]VT_I4 IniPos, [in]VT_I4 Interval, [in]VT_I4 PlsCount, [in]VT_I4 RepeatInterval, [in]VT_I4 RepeatCount) 위치비교 출력 기능 중 일정간격 위치비교 출력을 위한 환경설정을 구성합니다.
❒ VT_I4 DX_CmpGetRegularData([in] VT_HANDLE hDevice, [in]VT_I4 Channel, [out]VT_PI4 IniPos, [out]VT_PI4 Interval, [out]VT_PI4 PlsCount, [out]VT_PI4 RepeatInterval, [out]VT_PI4 RepeatCount) 위치비교 출력 기능 중 일정간격 위치비교 출력을 위한 환경설정을 반환합니다.
❒ VT_I4 DX_CmpSetRandomData ([in] VT_HANDLE hDevice, [in]VT_I4 Channel, [in]VT_I4 DataCount, [in]VT_PI4 DataList) 위치비교 출력 기능 중 임의간격 위치비교 출력을 위한 환경설정을 구성합니다.
❒ VT_I4 DX_CmpGetRandomData ([in] VT_HANDLE hDevice, [in]VT_I4 Channel, [out]VT_I4 DataCount, [out]VT_PI4 DataList) 위치비교 출력 기능 중 임의간격 위치비교 출력을 위한 환경설정을 반환합니다.
❒ VT_I4 DX_CmpStart([in] VT_HANDLE hDevice, [in]VT_I4 Channel) 위치비교 출력 기능을 시작합니다.
❒ VT_I4 DX_CmpStop([in] VT_HANDLE hDevice, [in]VT_I4 Channel) 위치비교 출력 기능을 종료합니다.
❒ VT_I4 DX_CmpIsActive([in] VT_HANDLE hDevice, [in]VT_I4 Channel, [out]VT_PI4 IsActive) 위치비교 출력 기능의 동작 상태를 반환합니다.
❒ VT_I4 DX_CmpClear([in] VT_HANDLE hDevice, [in]VT_I4 Channel) 위치비교 출력 기능을 초기화합니다.

8.6.2 함수 설명

NAME

DX_CmpSetConfig / DX_CmpGetConfig

- 위치비교출력기의 환경 설정 / 반환

INFORMATION

CMP

VC++ (6, 7, 8)/VB

BCB/Delphi

Level 1

위험 요소 없음

SYNOPSIS

❒ VT_I4 DX_CmpSetConfig([in] VT_HANDLE hDevice, [in]VT_I4 Channel, [in]VT_I4 CmpMode, [in]VT_I4 CmpMethod, [in]VT_I4 CmpInv, [in]VT_I4 PlsWidth)

❒ VT_I4 DX_CmpGetConfig([in] VT_HANDLE hDevice, [in]VT_I4 Channel, [out]VT_PI4 CmpMode, [out]VT_PI4 CmpMethod, [out]VT_PI4 CmpInv, [out]VT_PI4 PlsWidth)

DESCRIPTION

이 함수는 위치비교출력기의 환경 설정을 구성 / 반환 합니다.

PARAMETER

▶ ***hDevice*** : DX_GnLoadDevice()로부터 얻어온 Device 의 핸들값입니다.

▶ ***Channel*** : 위치비교 출력기의 채널 번호를 지정합니다. 채널 번호는 0 부터 시작합니다.

▶ ***CmpMode*** : 위치비교 출력기의 동작 모드 입니다.

Value	Meaning
0 (dxCMP_REGULAR)	위치비교 출력기의 일정간격 출력 모드 입니다.
1 (dxCMP_RANDOM)	위치비교 출력기의 임의간격 출력 모드 입니다.

▶ ***CmpMethod*** : 위치비교 신호의 출력 조건입니다.

Value	Meaning
0 (dxFALSE)	CmpData = CmpSrc_Value (while counting up)

1 (dxTRUE)	CmpData = CmpSrc_Value (while counting down)

▶ ***PlsWidth*** : 위치비교 출력기의 비교펄스 Width 입니다.

Pulse Width 의 설정 범위 및 설정 방식

Pulse Width 는 0 ~ 65534 까지 설정 가능하며, 이 값에 따라 설정되는 실제 펄스 폭은 다음과 같습니다.

실제 Pulse Width = PlsWidth * 50ns

단, PlsWidth 값이 0 일경우는 50ns 로 적용 됩니다.

RETURN VALUE

□ 함수 수행 성공 여부.

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

NAME

DX_CmpSetRegularData/

DX_CmpGetRegularData

- 일정간격 출력 환경 설정 / 반환

INFORMATION

CMP

VC++ (6, 7, 8)/VB

BCB/Delphi

Level 1

위험 요소 없음

SYNOPSIS

❒ VT_I4 DX_CmpSetRegularData ([in] VT_HANDLE hDevice, [in]VT_I4 Channel, [in]VT_I4 IniPos, [in]VT_I4 Interval, [in]VT_I4 PlsCount, [in]VT_I4 RepeatInterval, [in]VT_I4 RepeatCount)

❒ VT_I4 DX_CmpGetRegularData ([in] VT_HANDLE hDevice, [in]VT_I4 Channel, [out]VT_PI4 IniPos, [out]VT_PI4 Interval, [out]VT_PI4 PlsCount, [out]VT_PI4 RepeatInterval, [out]VT_PI4 RepeatCount)

DESCRIPTION

이 함수는 위치비교출력기의 동작 모드 중 일정간격 출력 모드에 대한 환경 설정을 구성 / 반환 합니다.
이 기능은 디바이스가 일정한 위치 간격을 가지는 연속적인 위치 데이터를 자동으로 생성하여 등록하도록 합니다.
이 기능은 총 두개의 일정 구간을 가지고 있습니다. COMI-DX50x 의 일정구간 비교 출력은 일정구간 A 가 일정 구간 B 만큼 반복하도록 설정되어 있습니다.

두 개의 일정 구간 주의 조건

일정구간 A 의 전체 크기는 일정구간 B 의 크기보다 작아야 합니다.

(일정구간 A 개수 X 일정구간 A 간격) < 일정구간 B 간격

PARAMETER

▶ ***hDevice*** : DX_GnLoadDevice()로부터 얻어온 Device 의 핸들값입니다.

▶ ***Channel*** : 위치비교 출력기의 채널 번호를 지정합니다. 채널 번호는 0 부터 시작합니다.

▶ ***IniPos*** : 위치비교 출력의 시작 위치값 입니다. 이 값은 논리적 거리 단위를 사용하는 절대 좌표값으로 설정합니다. (0 ~ 65535)

▶ ***Interval*** : 일정 구간 A 의 위치 간격 입니다. 이 값은 논리적 거리 단위로 설정합니다. (0 ~ 65534)

▶ ***PlsCount*** : 일정 구간 A 의 개수 입니다. 디바이스에 이 값만큼 일정 구간을 자동으로 생성합니다.

▶ ***RepeatInterval*** : 일정 구간 B 의 위치 간격 입니다. 이 값은 논리적 거리 단위로 설정합니다. (0 ~ 65534)

▶ ***RepeatCount*** : 일정 구간 B 의 개수 입니다. 디바이스에 이 값만큼 일정 구간을 자동으로 생성합니다.

.

RETURN VALUE

□ 함수 수행 성공 여부.

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

NAME

DX_CmpSetRandomData/

DX_CmpGetRandomData

- 임의 간격 출력 환경 설정 / 반환

INFORMATION

CMP

VC++ (6, 7, 8)/VB

BCB/Delphi

Level 1

위험 요소 없음

SYNOPSIS

❒ VT_I4 DX_CmpSetRandomData ([in] VT_HANDLE hDevice, [in]VT_I4 Channel, [in]VT_I4 DataCount, [in]VT_PI4 DataList)

❒ VT_I4 DX_CmpGetRandomData ([in] VT_HANDLE hDevice, [in]VT_I4 Channel, [out]VT_I4 DataCount, [out]VT_PI4 DataList)

DESCRIPTION

이 함수는 위치비교출력기의 동작 모드 중 임의 간격 출력 모드에 대한 환경 설정을 구성 / 반환 합니다. 이 기능은 위치비교출력기에 임의의 비교 데이터를 사용자가 설정하면, 디바이스는 카운터의 값을 비교데어터값과 비교하여 비교조건을 충족하면 CMP 출력을 내보냅니다.

PARAMETER

▶ ***hDevice*** : DX_GnLoadDevice()로부터 얻어온 Device 의 핸들값입니다.

▶ ***Channel*** : 위치비교 출력기의 채널 번호를 지정합니다. 채널 번호는 0 부터 시작합니다.

▶ ***DataCount*** : 위치비교 출력기에 입력할 비교데이터 버퍼의 크기 입니다.
비교데이터 버퍼의 크기는 총 2047byte 이므로 최대 데이터 버퍼의 크기는 2047byte 입니다.

▶ ***DataList*** : 위치비교 출력기에 입력한 데이터 버퍼의 배열 또는 포인터 입니다. 이 버퍼의 크기는 반드시 DataCount 보다 크거나 같아야 합니다.

RETURN VALUE

□ 함수 수행 성공 여부.

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

0 (dxERR_NONE)	수행 성공

NAME

DX_CmpStart / DX_CmpStop

- 위치비교출력 시작 / 종료

INFORMATION

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

SYNOPSIS

- ❒ VT_I4 DX_CmpStart ([in] VT_HANDLE hDevice, [in]VT_I4 Channel)
- ❒ VT_I4 DX_CmpStop ([in] VT_HANDLE hDevice, [in]VT_I4 Channel)

DESCRIPTION

이 함수는 위치비교출력기를 시작 / 종료 합니다.

PARAMETER

▶ ***hDevice*** : DX_GnLoadDevice()로부터 얻어온 Device 의 핸들값입니다.

▶ ***Channel*** : 위치비교 출력기의 채널 번호를 지정합니다. 채널 번호는 0 부터 시작합니다.

RETURN VALUE

□ 함수 수행 성공 여부.

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

NAME

DX_CmpIsActive

- 위치비교출력 구동 상태 반환

INFORMATION

CMP

VC++ (6, 7, 8)/VB

BCB/Delphi

Level 1

위험 요소 없음

SYNOPSIS

❒ VT_I4 DX_CmpIsActive ([in] VT_HANDLE hDevice, [in]VT_I4 Channel, [out]VT_PI4 IsActive)

DESCRIPTION

이 함수는 대상 디바이스의 위치비교출력 채널에 대하여 위치비교출력 기능의 구동 상태를 반환합니다.

PARAMETER

▶ ***hDevice*** : DX_GnLoadDevice()로부터 얻어온 Device 의 핸들값입니다.

▶ ***Channel*** : 위치비교 출력기의 채널 번호를 지정합니다. 채널 번호는 0 부터 시작합니다.

▶ ***IsActive*** : 위치비교 출력 기능의 구동 상태입니다.

Value	Meaning
0 (dxFALSE)	위치비교 출력기가 동작 중
1 (dxTRUE)	위치비교 출력기가 동작 중이지 않음.

RETURN VALUE

□ 함수 수행 성공 여부.

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

NAME

DX_CmpClear

- 위치비교출력 기 초기화

INFORMATION

CMP

VC++ (6, 7, 8)/VB

BCB/Delphi

Level 1

위험 요소 없음

SYNOPSIS

❒ VT_I4 DX_CmpClear([in] VT_HANDLE hDevice, [in]VT_I4 Channel)

DESCRIPTION

이 함수는 대상 디바이스의 위치비교출력 채널에 대하여 위치비교출력 기능을 초기화 합니다.

PARAMETER

▶ ***hDevice*** : DX_GnLoadDevice()로부터 얻어온 Device 의 핸들값입니다.

▶ ***Channel*** : 위치비교 출력기의 채널 번호를 지정합니다. 채널 번호는 0 부터 시작합니다.

RETURN VALUE

□ 함수 수행 성공 여부.

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

9 인터럽트 함수

이 단원에서는 인터럽트와 관련된 함수를 소개합니다. 인터럽트는 특정 상황이 발생 되었을 때 사용자(또는 프로그래머)에게 이것을 알려주기 위한 것입니다. 인터럽트는 폴링(Polling) 방식과 달리 CPU 에 부하를 주지 않는 것이 장점입니다. 윈도우 환경에서는 일반 Application 레벨에서 인터럽트를 처리 할 수 없으므로 이벤트를 통하여 Application 에게 인터럽트가 발생하였음을 알려줍니다.

COMI-DX 시리즈는 Interrupt Event 라는 이벤트를 제공합니다. 사용자는 이 이벤트 핸들러를 등록하여 손쉽게 이벤트를 통지 받을 수 있습니다. 이벤트 핸들러를 등록하는 방법은 일반적인 윈도우 리소스의 이벤트를 등록하는 것과 동일합니다.

사용자 프로그램에서 인터럽트 이벤트 핸들러를 등록하였으면 인터럽트가 발생할 때마다 자동적으로 해당 이벤트 핸들러가 호출됩니다. 사용자는 이벤트 핸들러에서 아래표와 같이 인터럽트 처리 관련 함수들을 사용하여 인터럽트의 발생 여부와 그에 대한 사후 처리 루틴을 작성 하여야 합니다. ㈜커미조아 DX-SDK 에서는 '윈도우 메시지' 방식, 'Call Back 함수 방식', '이벤트 객체' 방식의 3 가지의 이벤트 처리 방식을 사용 할 수 있습니다.

9.1 인터럽트 함수

인터럽트 처리에 관련된 이벤트 및 함수들은 다음과 같습니다.

함수 명	사용 가능 디바이스
DX_IntHandlerSetup	COMI-DX10x, COMI-DX20x, COMI-DX30x, COMI-DX50x
DX_IntDiHandlerEnable	
DX_IntDiHandlerEnable_Ex	
DX_IntCntHandlerEnable	
DX_IntAdScanHandlerEnable	COMI-DX10x, COMI-DX20x
DX_IntHandlerDisable	COMI-DX10x, COMI-DX20x, COMI-DX30x, COMI-DX50x
DX_IntAllDisable	
DX_IntGetEnbleState	

[표 8-2] 인터럽트와 관련된 함수 리스트 및 사용 가능 디바이스 안내

9.1.1 함수 요약

인터럽트와 관련된 함수들의 리스트는 다음과 같습니다.

Summary of Functions
❒ VT_I4 DX_IntHandlerSetup ([in] VT_HANDLE hDevice, [in]VT_I4 HandlerType, [in]VT_HANDLE Handler, [in]VT_I4 nMessage, [in]VT_HANDLE lParam, [out]VT_PI4 EventID) DX-SDK 가 지원하는 3 가지 유형의 인터럽트 처리 방식을 배경으로, 인터럽트가 발생 시 호출될 핸들러를 비롯한 환경 설정을 구성합니다.
❒ VT_I4 DX_IntDiHandlerEnable ([in] VT_HANDLE hDevice, [in]VT_I4 EventID, [in]VT_I4 IntMode, [in]VT_I4 Channel, [in]VT_I4 IntType) Digital Input 인터럽트를 시작합니다.
❒ VT_I4 DX_IntCntHandlerEnable ([in] VT_HANDLE hDevice, [in]VT_I4 EventID, [in]VT_I4 Channel, [in]VT_I4 RefCount, [in]VT_I4 RefBand, [in]VT_I4 UpDown) Counter 인터럽트를 시작합니다.
❒ VT_I4 DX_IntAdScanHandlerEnable ([in] VT_HANDLE hDevice, [in]VT_I4 EventID, [in]VT_I4 ChannelOrder, [in]VT_R8 RefVolt, [in]VT_R8 RefBand, [in]VT_I4 EdgeType) Ad Scan 인터럽트를 시작합니다.
❒ VT_I4 DX_IntHandlerDisable ([in] VT_HANDLE hDevice, [in]VT_I4 EventID) 특정 EventID 로 설정한 인터럽트 이벤트를 종료 합니다.
❒ VT_I4 DX_IntAllDisable ([in] VT_HANDLE hDevice) 활성화 된 모든 인터럽트 기능을 종료합니다.
❒ VT_I4 DX_IntGetEnableState ([in] VT_HANDLE hDevice, [out]VT_STRUCT IntrState) 활성화 되어 있는 인터럽트를 반환합니다.

9.1.2 함수 설명

NAME

DX_IntHandlerSetup

- 인터럽트 환경 설정

INFORMATION
Frequency Checker
VC++ (6, 7, 8)/VB
BCB/Delphi
Level 1
위험 요소 없음

SYNOPSIS

❒ VT_I4 DX_IntHandlerSetup ([in] VT_HANDLE hDevice, [in]VT_I4 HandlerType, [in]VT_HANDLE Handler, [in]VT_I4 nMessage, [in]VT_HANDLE lParam, [out]VT_PI4 EventID)

DESCRIPTION

이 함수는 이벤트 핸들러를 등록합니다. 윈도우 메시지 방식, 이벤트 객체 방식, 콜백 함수 방식등의 인터럽트 처리를 수행 할수 있으며, 지정된 HandlerType 에 따른 처리가 이루어지게 됩니다.
이벤트 처리에 대한 보다 자세한 정보는 ㈜ 커미조아 DX-SDK 에서 제공하는 예제 프로그램의 실제 코드를 통해서 확인 하시는 것도 좋은 방법 입니다.

PARAMETER

▶ ***hDevice*** : DX_GnLoadDevice()로부터 얻어온 Device 의 핸들값입니다.

▶ ***HandlerType*** : 인터럽트 핸들러의 종류를 지정 합니다. 이 값의 종류는 아래와 같이 3 가지로 지정할 수 있습니다..

Value	Meaning
0 (dxIHT_MESSAGE)	윈도우 메시지 전달방식을 통하여 인터럽트를 통지합니다. • 장점: GUI 관련 작업을 이벤트핸들러에서 직접 수행할 수 있다. • 단점: 윈도우가 메시지루프 상황에 따라서 상당한 지연시간을 가질 수 있다.
1 (dxIHT_EVENT)	이벤트 객체를 통하여 인터럽트를 통지합니다. • 장점: 메시지 전달 방식보다 비교적 적은 지연시간을 가질 수 있다. • 단점: 사용하기가 복잡하다.
2 (dxIHT_CALLBACK)	콜백 함수를 통하여 인터럽트를 통지합니다. 이 방식이 가장 권장되는 방식입니다. • 장점: 메시지 전달의 지연시간이 3 가지 방식 중에서 가장 적다. • 단점: GUI 관련 작업을 수행할 수 없다.

▶ ***Handler*** : 이 매개변수의 의미는 HandlerTYpe 의 설정에 따라서 다음과 같이 달라집니다.

HandlerType 설정 값	Handler 매개변수의 의미
0 (메시지 방식) 일때	Handler 매개변수는 윈도우 핸들을 의미합니다.
1 (이벤트 방식) 일때	Handler 매개변수는 이벤트 객체를 의미합니다.
2 (콜백 방식) 일때	Handler 매개변수는 콜백 함수를 의미합니다.

▶ ***nMessage*** : 이 매개변수는 HandlerType 이 dxIHT_MESSAGE 설정 되었을 때만 유효한 것으로서 윈도우 메시지 번호를 설정합니다.

▶ ***lParam*** : 인터럽트가 처리되는 함수에 전달 될 핸들 값을 설정합니다.

▶ ***EventID*** : 핸들러 함수를 등록할때마다 내부적으로 인터럽트 처리를 위한 EventID 가 생성됩니다. 이 EventID 는 DI, AI, CNT 인터럽트를 Enable 할 때 해당 인터럽트에 대한 정보를 할당 할 때 사용 되며, 해당 인터럽트가 Disable 될 때 이러한 정보는 내부 메모리에서 해제 됩니다.

.

RETURN VALUE

□ 함수 수행 성공 여부.

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

NAME

DX_IntDiHandlerEnable

- Digital Input 인터럽트 활성 / 비활성

INFORMATION

Frequency Checker

VC++ (6, 7, 8)/VB

BCB/Delphi

Level 1

위험 요소 없음

SYNOPSIS

❒ VT_I4 DX_IntDiHandlerEnable ([in] VT_HANDLE hDevice, [in]VT_I4 EventID, [in]VT_I4 IntMode, [in]VT_I4 Channel, [in]VT_I4 IntType)

DESCRIPTION

디지털 입력 신호 이벤트를 활성/ 비활성 합니다. 이 함수는 대상디바이스의 해당 디지털 입력 채널에 대하여 디지털 입력 신호가 입력되었을 때 (ON)에 인터럽트를 발생 시킵니다.

PARAMETER

▶ ***hDevice*** : DX_GnLoadDevice()로부터 얻어온 Device 의 핸들값입니다.

▶ ***EventID*** : 인터럽트 EventID 를 설정합니다. 이 값은 DX_IntHandlerSetup() 함수를 통해 가져옵니다..

▶ ***IntMode*** :

Value	Meaning
0 (dxINT_DI_SINGLE)	하나의 디지털 입력 채널에 대한 인터럽트 모드 입니다.
1 (dxINT_DI_MULTI)	다수의 디지털 입력 채널에 대한 인터럽트 모드 입니다. 이 모드는 지정한 다수의 채널이 모두 지정한 생태로 변화하였을 때 인터럽트가 발생합니다.
2 (dxINT_DI_MULTI_ONE)	다수의 디지털 입력 채널에 대한 인터럽트 모드 입니다. 이 모드는 지정한 다수의 채널 중 하나의 채널이라도 지정한 상태로 변화하였을 때 인터럽트가 발생합니다.

▶ ***Channel*** : IntMode 에 따라 적용 되는 값이 다릅니다. 각 값은 아래와 같습니다.

IntMode	Channel Mean
0 (dxINT_DI_SINGLE)	디지털 입력 채널 번호를 지정합니다. 채널 번호는 0 부터 시작합니다.

1 (dxINT_DI_MULTI)	다수의 디지털 입력 채널에 대한 채널 마스크를 설정합니다. Ex) 0x3 : 0~1 Channel, 0xf : 0~4 Channel
2 (dxINT_DI_MULTI_ONE)	다수의 디지털 입력 채널에 대한 채널 마스크를 설정합니다. Ex) 0x3 : 0~1 Channel, 0xf : 0~4 Channel

▶ ***IntType*** :

Value	Meaning
0 (dxINT_ON)	지정한 채널의 입력신호가 ON 상태일 경우 인터럽트를 발생시킵니다.
1 (dxINT_OFF)	지정한 채널의 입력 신호가 OFF 상태일 경우 인터럽트를 발생 시킵니다.
2 (dxINT_BI)	지정한 채널의 입력 신호가 ON/OFF 상태일 경우 인터럽트를 발생 시킵니다.

RETURN VALUE

□ 함수 수행 성공 여부.

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

EXAMPLE

□ 콜백함수 예제 입니다.

[C / C++]

```
// 0 번 채널이 ON 상태 일 경우 인터럽트 발생

// Callback Function
void WINAPI HandlerFunc( LPVOID lParam )
{
    // Data Process…
}

int main(void)
```

```
{
    DX_IntHandlerSetup( hDevice, dxINT_DI, dxIHT_CALLBACK, HandlerFunc, NULL, this);
    DX_IntDiHandleEnable(hDevice, EVENT_ID, dxDI_SINGLE, DEF_CH0, dxINT_UP);
}
void OnDestroy()
{
    DX_IntHandlerDisable(hDevice, EVENT_ID);
}
```

□ 메시지 방식 예제 입니다.

[C / C++]

```
// .h 파일 추가
// 0 번 채널이 OFF 상태 일 경우 인터럽트 발생
#define WMU_INTERRUPT (WM_APP + 1) // User Message
// BEGIN_MESSAGE_MAP 추가
ON_MESSAGE(WMU_INTERRUPT, OnMessageFunc)
//Windows Message 처리 함수
LONG OnMessageFunc(WPARAM wParam, LPARAM lParam)
{
    // Data Process …
}
int main(void)
{
    DX_IntHandlerSetup( hDevice, dxINT_DI, dxIHT_MESSAGE, GetSafeHwnd(),WMU_INTERRUPT, this);
    DX_IntDiHandleEnable(hDevice, EVENT_ID, dxDI_SINGLE, DEF_CH0, dxINT_OFF);
}
void OnDestroy()
{
    DX_IntHandlerDisable(hDevice, EVENT_ID);
}
```

□ 이벤트 방식 예제 입니다.

[C / C++]

```
int main(void)
{
    HANDLE hEvent;
    hEvent = CreateEvent(NULL, FALSE, FALSE, NULL);
```

```
    DX_IntHandlerSetup( hDevice, dxINT_DI, dxIHT_EVENT, hEvent, NULL, NULL);

    DX_IntDiHandleEnable(hDevice, EVENT_ID, dxDI_SINGLE, DEF_CH0, dxINT_UP);

    while(1) {
            WaitForSingleObject(hEvent, INFINITE);
            ResetEvent(hEvent);

            // Data Process…
    }

    CloseHandle(hEvent);

DX_IntHandlerDisable(hDevice, EVENT_ID);
}
```

NAME

DX_IntCntHandlerEnable

- Counter 인터럽트 활성 / 비활성

INFORMATION
Frequency Checker
VC++ (6, 7, 8)/VB
BCB/Delphi
Level 1
위험 요소 없음

SYNOPSIS

❒ VT_I4 DX_IntCntHandlerEnable ([in] VT_HANDLE hDevice, [in]VT_I4 EventID, [in]VT_I4 Channel, [in]VT_I4 RefCount, [in]VT_I4 RefBand, [in]VT_I4 UpDown)

DESCRIPTION

카운터 이벤트를 활성/ 비활성 합니다. 이 함수는 대상디바이스의 해당 카운터 채널에 대하여 기준 카운터 값에 대한 Band 를 설정한후 카운터 방향에 따라 인터럽트를 발생시킵니다.

	Counter Interrupt 기능은 32Bit Counter 기능 동작중에 사용할 수 있습니다. DX_CntStart() 함수를 통해 카운터 기능이 Enable 이된 상태에서 동작합니다..

PARAMETER

▶ ***hDevice*** : DX_GnLoadDevice()로부터 얻어온 Device 의 핸들값입니다.

▶ ***EventID*** : 인터럽트 EventID 를 설정합니다. 이 값은 DX_IntHandlerSetup() 함수를 통해 가져옵니다..

▶ ***Channel*** : 카운터 채널 번호를 지정합니다. 채널 번호는 0 부터 시작합니다.

▶ ***RefCount*** : 인터럽트를 발생 시킬 카운터 채널의 카운트 값 입니다.

▶ ***RefBand*** : 유효한 Reference 값을 참조하기 위한 Band 를 설정하기 위한 값입니다.
이 값의 단위는 카운터 값입니다. 예를 들어 RefCount 가 100 이고 RefBand 가 50 이라면 Band 는 75~125 입니다.

▶ ***UpDown*** : 인터럽트를 발생 시킬 카운터의 방향 설정 값 입니다.

Value	Meaning
0 (dxINT_UP)	카운터 값이 증가할 때 인터럽트가 발생.
1 (dxINT_DOWN)	카운터 값이 감소 할 때 인터럽트가 발생.
2 (dxINT_BI)	카운터 값이 증가/감소 할 때 인터럽트가 발생.

RETURN VALUE

□ 함수 수행 성공 여부.

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

NAME

DX_IntAdScanHandlerEnable

- AdScan 인터럽트 활성 / 비활성

INFORMATION
Frequency Checker
VC++ (6, 7, 8)/VB
BCB/Delphi
Level 1
위험 요소 없음

SYNOPSIS

❒ VT_I4 DX_IntAdScanHandlerEnable ([in] VT_HANDLE hDevice, [in]VT_I4 EventID, [in]VT_I4 ChannelOrder, [in]VT_R8 RefVolt, [in]VT_R8 RefBand, [in]VT_I4 EdgeType)

DESCRIPTION

Ad Scan 이벤트를 활성/ 비활성 합니다. 이 함수는 대상디바이스의 해당 아날로그 입력 채널에 대하여 아날로그 입력값이 Volt +/- Range 가 되었을 때 UpDown 비교조건에 따라서 인터럽트를 발생 시킵니다..

주의	ADScan Interrupt 기능은 AD Scan 기능 중 Filter Mode 로 동작시에만 사용할 수 있습니다. DX_AdScanStart() 함수를 통해 AD Scan 기능이 Enable 된 상태에서 동작합니다.

PARAMETER

▶ ***hDevice*** : DX_GnLoadDevice()로부터 얻어온 Device 의 핸들값입니다.

▶ ***EventID*** : 인터럽트 EventID 를 설정합니다. 이 값은 DX_IntHandlerSetup() 함수를 통해 가져옵니다..

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

▶ ***RefVolt*** : 인터럽트를 발생 시킬 아날로그 입력 값 입니다.

▶ ***RefBand*** : 유효한 Reference 값을 참조하기 위한 Band 를 설정하기 위한 값입니다.
이 값의 단위는 A/D 입력 범위에 대한 백분율(%)입니다. 예를 들어, 만일 input Range 가 -10~10 이고, RefBand 값이 1 인 경우에는 Range 크기가 20 이므로 Band 크기는 20*0.01 = 0.2Volt 가 됩니다.

▶ ***EdgeType*** : 인터럽트를 발생시키기위한 아날로그 파형의 Edge 조건입니다.

Value	Meaning
0 (dxTE_POSITIVE)	Positive edge 에서 인터럽트 발생
1 (dxTE_NEGATIVE)	Negative edge 에서 인터럽트 발생
2 (dxTE_ALL)	Positive edge & Negative edge 양쪽에서 인터럽트 발생

RETURN VALUE

□ 함수 수행 성공 여부.

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

NAME

DX_IntHandlerDisable

- 인터럽트 종료

INFORMATION

Frequency Checker

VC++ (6, 7, 8)/VB

BCB/Delphi

Level 1

위험 요소 없음

SYNOPSIS

❒ VT_I4 DX_IntHandlerDisable ([in] VT_HANDLE hDevice, [in]VT_I4 EventID)

DESCRIPTION

이 함수는 사용자가 지정한 EventID 에 따른 인터럽트를 종료합니다.

PARAMETER

▶ ***hDevice*** : DX_GnLoadDevice()로부터 얻어온 Device 의 핸들값입니다.

▶ ***EventID*** : 인터럽트 EventID 를 설정합니다. 이 값은 다른 인터럽트 EventID 와 독립적이어야 합니다.

RETURN VALUE

□ 함수 수행 성공 여부.

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

NAME

DX_IntAllDisable

- 인터럽트 타입별 종료

INFORMATION

Frequency Checker

VC++ (6, 7, 8)/VB

BCB/Delphi

Level 1

위험 요소 없음

SYNOPSIS

❒ VT_I4 DX_IntAllDisable ([in] VT_HANDLE hDevice)

DESCRIPTION

이 함수는 DX_IntHandlerSetup() 함수와 각각의 인터럽트 Enable 함수를 통하여 활성화된 모든 인터럽트 체크 기능을 종료 합니다.

PARAMETER

▶ ***hDevice*** : DX_GnLoadDevice()로부터 얻어온 Device 의 핸들값입니다.

RETURN VALUE

□ 함수 수행 성공 여부.

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

NAME

DX_IntGetEnableState

- 인터럽트 동작 상태 반환

INFORMATION

- Frequency Checker
- VC++ (6, 7, 8)/VB
- BCB/Delphi
- Level 1
- 위험 요소 없음

SYNOPSIS

❒ VT_I4 DX_IntGetEnableState ([in] VT_HANDLE hDevice, [out]VT_STRUCT IntrState)

DESCRIPTION

이 함수는 현재 활성화 되어 있는 인터럽트들의 개수와 타입, 설정된 채널 정보를 TDXInterruptState 구조체로 반환 합니다.

PARAMETER

▶ ***hDevice*** : DX_GnLoadDevice()로부터 얻어온 Device 의 핸들값입니다.

▶ ***InterState*** : 활성화 되어 잇는 인터럽트 개수, 타입, 채널 정보를 가지고 있는 구조체 입니다. 구조체의 인자는 아래와 같습니다.

Value	Meaning
nNumCount	활성화 되어 있는 인터럽트의 개수 입니다.
nIntrType	활성화 되어 있는 인터럽트들의 타입 배열입니다. nNumCount 만큼의 데이터를 가지고 있으며, 인터럽트는 최고 10 개까지 가능합니다.
nChanList	활성화 되어 있는 인터럽트의 채널 배열 입니다. nNumCount 만큼의 데이터를 가지고 있으며, 인터럽트는 최고 10 개까지 가능합니다.

RETURN VALUE

□ 함수 수행 성공 여부.

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

Appendix

A

Supporting Functions

제품별 함수 지원 구분은 각 디바이스에 대해 지원되는 함수를 Device 별로 구분하여 정리하였습니다. 사용자는 디바이스에 따라 사용할 수 있는 함수를 확인하고, 지원되는 기능을 사용하시기 바랍니다 .

I COMIZOA DAQ Device

I.I COMI-DX Series

Class	Function	DX10x	DX20x	DX30x	DX501
General	DX_GnLoadDevice	V	V	V	V
	DX_GnUnloadDevice	V	V	V	V
	DX_ErrGetLastErrorCode	V	V	V	V
	DX_ErrShowLastError	V	V	V	V
	DX_DLogSetup	V	V	V	V
	DX_DLogSetFilePath	V	V	V	V
	DX_DLogAddComment	V	V	V	V
	DX_DLogGetCurType	V	V	V	V
	DX_DLogGetCurLevel	V	V	V	V
	DX_DLogGetCurFilePath	V	V	V	V
Analog Input	DX_AdSetInputType	V	V		
	DX_AdSetRange	V	V		
	DX_AdGetRange	V	V		
	DX_AdGetDigit	V	V		
	DX_AdGetVolt	V	V		
	DX_AdScanSetRange	V	V		
	DX_AdScanGetRange	V	V		
	DX_AdScanSetChannelList	V	V		
	DX_AdScanSetTriggerMode	V	V		
	DX_AdScanGetTriggerMode	V	V		
	DX_AdScanStart	V	V		
	DX_AdScanFilterStart	V	V		
	DX_AdScanStop	V	V		
	DX_AdScanClear	V	V		
	DX_AdScanChangeFreq	V	V		
	DX_AdScanIsBufFull	V	V		
	DX_AdScanResume	V	V		
	DX_AdScanRetrChannelI2	V	V		

	DX_AdScanRetrChannelF4	V	V		
	DX_AdScanRetrChannelF8	V	V		
	DX_AdScanRetrBlockI2	V	V		
	DX_AdScanRetrBlockF4	V	V		
	DX_AdScanRetrBlockF8	V	V		
	DX_AdScanFilterConfig	V	V		
	DX_AdScanGetMinMax	V	V		
Analog Output	DX_DaClear	V		V	
	DX_DaSetRange	V		V	
	DX_DaGetRange	V		V	
	DX_DaOut	V		V	

Class	Function	DX10x	DX20x	DX30x	DX501
Digital In/Out	DX_DioSetUsage	V	V	V	V
	DX_DioGetUsage	V	V	V	V
	DX_DiGetOne	V	V	V	V
	DX_DiGetAll	V	V	V	V
	DX_DoGetOne	V	V	V	V
	DX_DoGetAll	V	V	V	V
	DX_DoPutOne	V	V	V	V
	DX_DoPutAll	V	V	V	V
Counter	DX_CntClear	V	V	V	V
	DX_CntSetClearMode	V	V	V	V
	DX_CntGetClearMode	V	V	V	V
	DX_CntSetFilter	V	V	V	V
	DX_CntGetFilter	V	V	V	V
	DX_CntSetConfig	V	V	V	V
	DX_CntGetConfig	V	V	V	V
	DX_CntStart	V	V	V	V
	DX_CntIsActive	V	V	V	V
	DX_CntStop	V	V	V	V

	DX_CntSetDefault	V	V	V	V
	DX_CntGetDefault	V	V	V	V
	DX_CntGetCount	V	V	V	V
	DX_PgSetConfig	V	V	V	V
	DX_PgGetConfig	V	V	V	V
	DX_PgSetInverse	V	V	V	V
	DX_PgGetInverse	V	V	V	V
	DX_PgStart	V	V	V	V
	DX_PgStop	V	V	V	V
	DX_PgIsActive	V	V	V	V
	DX_FcSetCounter	V	V	V	V
	DX_FcStart	V	V	V	V
	DX_FcStop	V	V	V	V
	DX_FcGetFrequency	V	V	V	V
	DX_FcIsActive	V	V	V	V

Class	Function	DX10x	DX20x	DX30x	DX501
Counter	DX_EcSetConfig				V
	DX_EcGetConfig				V
	DX_EcFilterEnable				V
	DX_EcFilterDisable				V
	DX_EcSetMaxCount				V
	DX_EcGetMaxCount				V
	DX_EcSetDefault				V
	DX_EcGetDefault				V
	DX_EcGetCount				V
	DX_EcZClearEnable				V
	DX_EcZClearDisable				V
	DX_EcZSetMaxCount				V
	DX_EcZGetMaxCount				V
	DX_EcZSetDefault				V
	DX_EcZGetDefault				V

	DX_EcZGetCount				V
	DX_EcStart				V
	DX_EcStop				V
	DX_EcIsActive				V
	DX_EcClear				V
	DX_LtcSetMode				V
	DX_LtcGetMode				V
	DX_LtcStart				V
	DX_LtcStop				V
	DX_LtcIsActive				V
	DX_LtcClear				V
	DX_LtcGetDataCount				V
	DX_LtcReadOneLatch				V
	DX_LtcReadAllLatch				V
	DX_CmpSetConfig				V
	DX_CmpGetConfig				V
	DX_CmpSetRegularData				V
	DX_CmpGetRegularData				V
	DX_CmpSetRandomData				V
	DX_CmpGetRandomData				V
	DX_CmpStart				V
	DX_CmpStop				V
	DX_CmpIsActive				V
	DX_CmpClear				V

Class	Function	DX10x	DX20x	DX30x	DX501
Interrupt	DX_IntHandlerSetup	V	V	V	V
	DX_IntDiHandlerEnable	V	V	V	V
	DX_IntDiHandlerEnable_Ex	V	V	V	V
	DX_IntCntHandlerEnable	V	V	V	V
	DX_IntAdScanHandlerEnable	V	V		
	DX_IntAllDisable	V	V	V	V
	DX_IntGetEnalbeState	V	V	V	V

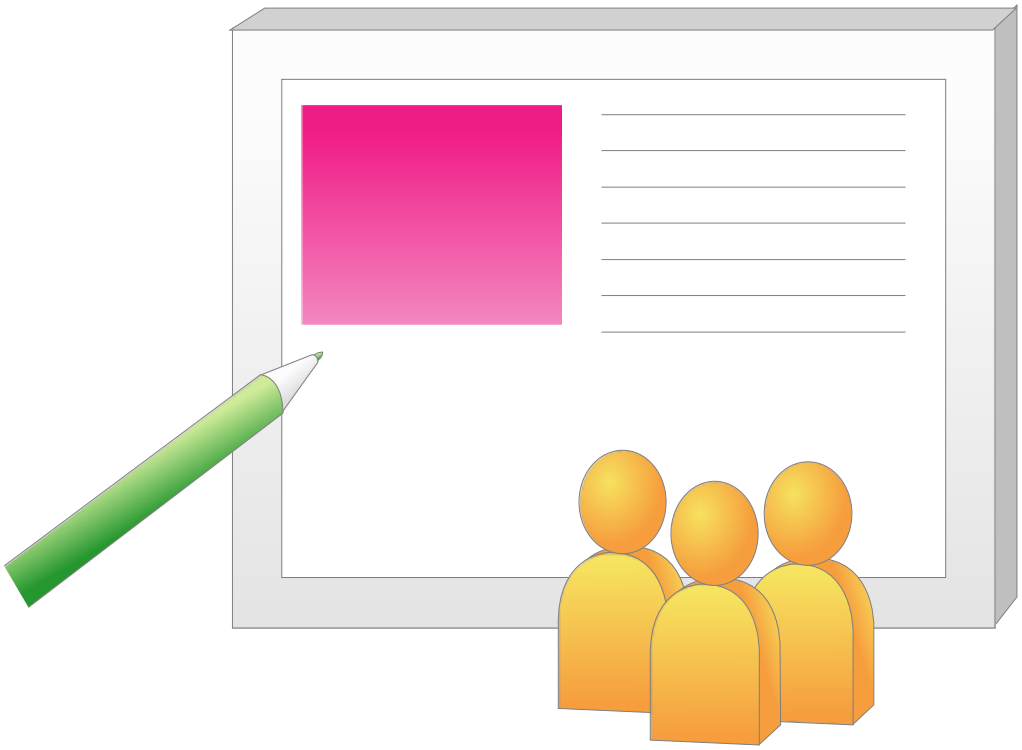
Appendix

B

List of Error Codes

에러코드는 모든 응용 프로그램에서 필요한 주요 정보 중 하나입니다. 커미조아의 CMD-SDK 에서는 자세하고, 일괄적인 에러코드를 제공함으로써, 에러 상태에 대한 명확한 원인과 분석이 가능할 수 있도록 돕고 있습니다. 본 장에서 열거한 에러코드는 다양한 에러 코드 상황에서, CMD-SDK 를 통해 에러 상태를 효율적으로 자세하고 명확히 판단할 수 있는 근거를 제공합니다.

라이브러리에서 제공하는 에러코드 리스트를 일람표로 정리하여 수록하였으며, 명시된 에러코드를 통해 효과적으로 에러처리를 하는 방법에 대해서 안내하고 있습니다. 일반적으로 CMD-SDK 사용하다가 에러가 발생하면 에러코드에 대한 의미를 에러코드 일람표를 참조하여 파악하시기 바랍니다.



II 에러 코드

II.I 에러 코드 일람표

ERROR CODE	VALUE	MEANING
dxERR_NONE	0	No error
dxERR_DEV_NOE_EXIST	-1000	Available device is not exist.
dxERR_UNSUPORTED_FUNC	-1001	User called an unsupported function for the specified product.
dxERR_INVALID_DEVICE_HANDLE	-1010	Device handle is not valid. This means that loading a device has been failed or not performed. Refer to “DX_GnLoadDevice” function.
dxERR_INVALID_PARAMETER	-1011	Some of the function parameters are invalid.
dxERR_INVALID_CHANNEL	-1012	The channel setting parameter(s) is(are) invalid
dxERR_INVALID_RANGE	-1013	User selected invalid analog input voltage range
dxERR_INVALID_SETTING	-1014	User selected invalid setting parameter.
dxERR_PARAMETER_NONE	-1015	The parameter is NULL.
dxERR_DEVICE_LOAD_FAIL	-1200	Device load fail.
dxERR_FILE_OPEN_FAIL	-1201	File creation / open has been failed
dxERR_MEM_ALLOC_FAIL	-2000	Buffer memory allocated fail
dxERR_CNT_MODE_DISABLE	-3000	(use the counter functions) The counter clear mode is disabled.
dxERR_CNT_MODE_EXTERNAL	-3001	(use the counter functions) The counter clear mode is a external clear mode. Please, set ‘dxCMODE_SRC_SOFT0’ or ‘dxCMODE_Src_SOFT1’
dxERR_DSP_DOES_NOT_REPLY	-4000	Internal CPU of the device is not working well. Please, Reboot the PC or reinstall the driver.
dxERR_AD_SCAN_RUNNING	-5000	(use the Adsacan functions) Some of the function is not working when running the adscan.
dxERR_AD_SCAN_NEED_CH_SET	-5001	(use the Adsacan functions) User need to set ‘DX_AdScanSetChannelList’.
dxERR_AD_SCAN_ONLY_FILTER	-5002	(use the Adsacan functions) Some of functions are working well in Adscan filter mode.

ERROR CODE	VALUE	MEANING
dxERR_INT_NEED_HANDLER	-6000	(use the interrupt functions) User need to set ‘DX_IntHandlerSetup’.

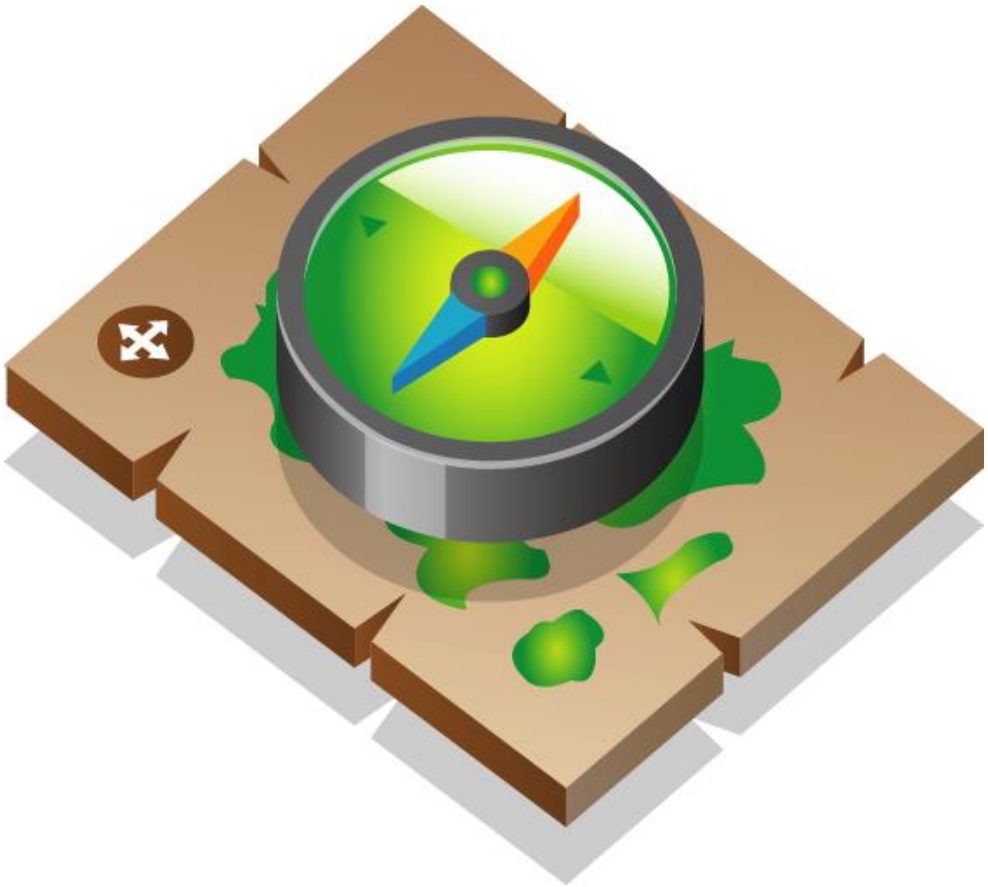
Appendix

C

Index of DX-SDK functions

필요한 함수를 가장 빠르고 쉽게 찾으십시오. DX-SDK 매뉴얼에서는 고객님들께서 원하시는 함수들을 일목 요연하게 정리하였습니다. PDF 문서에서는 필요한 함수를 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 DX-SDK Functions

I.I Quick Reference to DX-SDK Functions

General Functions

TEST & MEASUREMENT & AUTOMATION / COMIZOA

DX-SDK Manual

2014 년 08 월 01 일 1 판 인쇄
매뉴얼 자료 번호 : 1.0.6

® COMIZOA
http://www.comizoa.com
Tel) +82 - 42 - 936 - 6500~6
Fax) +82 - 42 - 936 - 6507