

Library User Manual

AXL library family
AXD



Product Information

Full information about other AJINEXTEK products is available by visiting our Web Site at:

Home Page : www.ajinextek.com
E-Mail : support@ajinextek.com

Useful Contact Information

Customer Support Seoul

Tel : 82-031-436-2180~2

Fax: 82-031-436-2183

Customer Support Cheonan

Tel : 82-041-555-9771~2 Fax: 82-041-555-9773

Customer Support Deagu

Tel : 82-053-593-3700~2 Fax: 82-053-593-3703



AJINEXTEK's sales team is always available to assist you in making your decision the final choice of boards or systems is solely and wholly the responsibility of the buyer. AJINEXTEK's entire liability in respect of the board or systems is as set out in AJINEXTEK's standard terms and conditions of sale

Content

CONTENT.....	3
DIO 라이브러리 사용법	8
DIO 개요.....	8
DIO 함수의 기본 규칙.....	9
Axd 사용 접두어별 그룹.....	11
초기화 및 종료.....	12
라이브러리의 초기화.....	12
라이브러리의 종료.....	13
입출력 포트의 신호레벨 설정	16
Offset의 설명.....	16
함수 설명.....	18
입출력포트 읽기.....	20
함수 설명.....	20
출력포트 쓰기.....	24
함수 설명.....	24
인터럽트	29
Interrupt Rising / Falling Edge의 설정	29
Interrupt Rising / Falling Edge의 설정값 확인	30
인터럽트 사용여부의 설정	31
인터럽트 처리 방식의 설정	31
고급 DIO함수	38
신호의 변화 여부 확인.....	38
신호의 지속여부 확인	40
일정시간동안 신호유지.....	43
신호의 토글.....	46

DIO COMMAND 매뉴얼 정보	49
헤더 파일	49
함수 용어	50
본 매뉴얼의 함수 이름	50
본 매뉴얼의 인자 이름	50
Interrupt Rising / Falling Edge Register	53
DIO COMMAND QUICK LIST	54
보드 및 모듈 정보	54
인터럽트	54
입력 레벨 설정 확인	55
출력 레벨 설정 확인	55
입력 포트 읽기	55
출력 포트 읽기 쓰기	56
고급함수	56
Define문	58
DIO COMMAND FUNCTION LIST	59
보드 및 모듈 정보	59
AxdInfoIsDIOModule	60
AxdInfoGetModuleNo	62
AxdInfoGetModuleCount	64
AxdInfoGetInputCount	66
AxdInfoGetOutputCount	68
AxdInfoGetModule	70
AxdInfoGetModuleStatus	72
인터럽트	74
AxdIInterruptSetModule	75
AxdIInterruptSetModuleEnable	77
AxdIInterruptGetModuleEnable	79

AxdIInterruptRead	81
AxdIInterruptEdgeSetBit	83
AxdIInterruptEdgeSetByte	85
AxdIInterruptEdgeSetWord	87
AxdIInterruptEdgeSetDword	89
AxdIInterruptEdgeGetBit	91
AxdIInterruptEdgeGetByte	93
AxdIInterruptEdgeGetWord	95
AxdIInterruptEdgeGetDword	97
AxdIInterruptEdgeSet	99
AxdIInterruptEdgeGet	101
입력 레벨 설정 확인	103
AxdILevelSetImportBit	104
AxdILevelSetImportByte	106
AxdILevelSetImportWord	108
AxdILevelSetImportDword	110
AxdILevelGetImportBit	112
AxdILevelGetImportByte	114
AxdILevelGetImportWord	116
AxdILevelGetImportDword	118
AxdILevelSetImport	120
AxdILevelGetImport	122
출력 레벨 설정 확인	124
AxdoLevelSetOutportBit	125
AxdoLevelSetOutportByte	127
AxdoLevelSetOutportWord	129
AxdoLevelSetOutportDword	131
AxdoLevelGetOutportBit	133
AxdoLevelGetOutportByte	135
AxdoLevelGetOutportWord	137
AxdoLevelGetOutportDword	139

AxdoLevelSetOutport	141
AxdoLevelGetOutport	143
입력 포트 읽기.....	145
AxdiReadImportBit.....	146
AxdiReadImportByte	148
AxdiReadImportWord.....	150
AxdiReadImportDword.....	152
AxdiReadImport.....	154
출력 포트 읽기 쓰기	156
AxdoWriteOutportBit	157
AxdoWriteOutportByte	159
AxdoWriteOutportWord.....	161
AxdoWriteOutportDword	163
AxdoReadOutportBit	165
AxdoReadOutportByte.....	167
AxdoReadOutportWord	169
AxdoReadOutportDword	171
AxdoWriteOutport	173
AxdoReadOutport	175
고급함수	177
AxdiIsPulseOn	178
AxdiIsPulseOff.....	180
AxdiIsOn.....	182
AxdiIsOff	185
AxdoOutPulseOn	188
AxdoOutPulseOff.....	190
AxdoToggleStart	192
AxdoToggleStop	194
AxdoSetNetworkErrorAct.....	196
AxdSetContactNum	198
AxdGetContactNum.....	200

에러코드 테이블 (Error Code Table) 확인.....	202
-------------------------------------	-----

DIO 라이브러리 사용법

DIO 개요

하나의 장비가 구동되는데는 수많은 신호 입출력이 필요하다. 일반적으로 이러한 신호들은 디지털 신호가 주류를 이룬다. 이러한 디지털 신호를 입력받고, 출력하기 위한 제어모듈이 DIO (Digital Input Output) 모듈이다.

디지털 입출력에 대해서는 개별적인 비트들의 입출력 상태를 검출하고 사용하는 함수 뿐만 아니라, 8비트, 16비트, 32비트 단위로의 입출력을 지원하는 함수들도 포함하고 있다. 게다가, 각각의 입력비트에 대해서 상승 및 하강 에지에서의 인터럽트를 사용할 수도 있다. DIO 모듈에 대한 라이브러리 함수는 크게 다음과 같이 나누어 진다.

- 초기화: 모듈을 초기화 시키는데 관련된 함수.
- 모듈정보 확인: 모듈의 각종 정보를 확인하는 함수.
- 신호 입출력: 각각의 접점을 비트/바이트/워드/더블워드 단위로 입력을 확인하고 출력하는데 관련된 함수.
- 인터럽트: 입력비트들에서 상승 또는 하강에지에 대해 인터럽트를 발생시키는 것을 설정 및 확인하는 것에 관련된 함수.

본User's Guide에서는 여러 디지털 입출력 모듈 중 16개의 입력비트와 16개의 출력 비트를 가지는 SIO-DB32P모듈을 기준으로 설명한다.

AXD 라이브러리에서 지원하는 제품 내역은 다음과 같다.

이름	설명	인터럽트 지원 여부	비고
SIO-DB32	입/출력 각 16점	지원	Local Board 제품
SIO-DI32	입력 32 점	지원	Local Board 제품
SIO-DO32	출력 32 점	미지원	Local Board 제품
PCI-DB64R	입/출력 각 32 점	지원	Local Board 제품
PCI-DI64R	입력 64점	지원	Local Board 제품
PCI-DO64R	출력 64점	미지원	Local Board 제품
SIO-RDI32	입력 32 점	미 지원	RTEX 제품
SIO-RDO32	출력 32 점	미 지원	RTEX 제품
JEPMC-IO2310	입/출력 각 32 점	미 지원	메카트로링크 II 제품

DIO 함수의 기본 규칙

모든 AXL함수의 기본적인 규칙이기도 하지만, DIO함수군인 AXD에서도 각각의 함수들은 기본적으로 함수의 수행 결과를 리턴한다. 리턴값은 아래의 표에서와 같이 해당 함수가 정상적으로 수행되었는지, 아니면 어떠한 이유로 실행되지 못했는지를 알려준다.

함수의 리턴값을 확인하고자 한다면, 다음과 같이 한다.

```
DWORD Code = AxdInfoIsDIOModule(&dwStatus);

if(Code == AXT_RT_SUCCESS)
    printf("정상적으로 실행 되었습니다.");
else
    printf("AxdInfoIsDIOModule() : ERROR ( Return FALSE ) code 0x%x\n",Code);
```

Error Code	Description
[0000] AXT_RT_SUCCESS	API 함수 수행 성공
[1001] AXT_RT_OPEN_ERROR	라이브러리 오픈 되지않음
[1002] AXT_RT_OPEN_ALREADY	라이브러리 오픈 되어있고 사용 중임
[1053] AXT_RT_NOT_OPEN	라이브러리 초기화 실패
[1054] AXT_RT_NOT_SUPPORT_VERSION	지원하지않는 하드웨어
[1101] AXT_RT_INVALID_BOARD_NO	유효하지않는 보드 번호
[1102] AXT_RT_INVALID_MODULE_POS	유효하지않는 모듈 위치
[1103] AXT_RT_INVALID_LEVEL	유효하지않는 레벨
[1151] AXT_RT_ERROR_VERSION_READ	라이브러리 버전을 읽을수 없음
[3001] AXT_RT_DIO_OPEN_ERROR	DIO 모듈 오픈실패
[3051] AXT_RT_DIO_NOT_MODULE	DIO 모듈 없음
[3052] AXT_RT_DIO_NOT_INTERRUPT	DIO 인터럽트 설정안됨
[3101] AXT_RT_DIO_INVALID_MODULE_NO	유효하지않는 DIO 모듈 번호
[3102] AXT_RT_DIO_INVALID_OFFSET_NO	유효하지않는 DIO OFFSET 번호
[3103] AXT_RT_DIO_INVALID_LEVEL	유효하지않는 DIO 레벨
[3104] AXT_RT_DIO_INVALID_MODE	유효하지않는 DIO 모드
[3105] AXT_RT_DIO_INVALID_VALUE	유효하지않는 값 설정
[3106] AXT_RT_DIO_INVALID_USE	DIO 함수 사용못함

또한, 함수들은 매개변수를 통해 필요한 인자들을 입력받고, 출력또한 변수의 포인터를 넘겨받아 처리결과를 저장하여 사용자에게 전달한다. 아래에서와 같이 특정 함수 AxdExFunction를 실행시키는데, inPara1, inPara2,의 값이 필요하고, 결과값은 outPara1, outPara2에 받아온다고 하면, 아래와 같이 한다.

```
double inPara1, inPara2;  
double outPara1, outPara2;  
AxsdExFunction(inPara1, inPara2, &outPara1, &outPara2);
```

Axd 사용 접두어별 그룹

DIO 함수는 다음과 같은 함수 접두어 그룹으로 나뉜다.

접두어	설명
AxdInfo	보드 및 모듈 정보 확인 함수
AxdI	Digital 입력 관련 대분류
AxdO	Digital 출력 관련 대분류
AxdIInterrupt	입력 인터럽트 설정 확인
AxdIInterruptEdge	입력 인터럽트 상승 / 하강 에지시 인터럽트 발생 설정 확인
AxdILevel	입력 신호 레벨 설정 확인
AxdIRead	입력 신호 읽기
AxdORead	출력 신호 읽기
AxdOWrite	출력 신호 쓰기
AxdOLevel	출력 신호 레벨 설정 확인

초기화 및 종료

AXL 라이브러리의 초기화 작업은 AxIOpen함수를 실행함으로써 이루어진다. AxIOpen함수를 실행하면, 내부적으로 라이브러리의 초기화뿐만 아니라 설치되어 있는 보드 및 보드에 존재하는 모듈을 모두 초기화하게 된다. 그러므로 사용자는 AIO라이브러리를 사용하기 전에 반드시 AIO모듈이 존재 하는가를 확인하여야 한다.

라이브러리의 초기화

● 라이브러리의 초기화

AxIOpen : AxIOpen 함수는 라이브러리를 초기화 하는 함수이다. 사용자가 인터럽터를 사용할 경우 인터럽터를 처리할 IRQ 번호를 등록한다. 특히 PCI타입의 베이스보드에서는 IRQ번호를 자동으로 생성하므로 라이브러리 초기화 시 입력한 IRQ번호는 무시된다.

이 함수는 리턴 값으로 에러코드가 아닌 BOOL값을 리턴하므로 아래에서처럼 사용할 수 있다.

```
// 라이브러리를 초기화 한다.  
// 7은 IRQ를 뜻한다. PCI에서는 자동으로 IRQ가 설정된다.  
if(AxIOpen(7) == AXT_RT_SUCCESS)  
    printf("라이브러리가 초기화 되었습니다.");
```

● 라이브러리 초기화 여부의 확인

AxIsOpened : AxIsOpened 함수는 라이브러리의 초기화 여부를 확인하는 함수이다. 이 함수 또한 에러코드가 아닌 BOOL값을 리턴한다. 라이브러리가 초기화되어 있다면, TRUE를 반환하고, 초기화되어 있지 않다면 FALSE를 리턴한다.

```
// 라이브러리가 초기화 되어 있는지 확인한다.  
if(AxIsOpened())  
    printf("라이브러리가 초기화 되어 있습니다.");  
else  
    printf("라이브러리가 초기화 되지 않았습니다.");
```

● AIO모듈의 존재 여부 확인

[AxdInfoIsDIOModule](#): AxdInfoIsDIOModule 함수는 실제 DIO 모듈이 존재하는가 여부를 확인하는 함수이다. DIO관련 함수를 사용하기 전에 DIO모듈이 있는지를 먼저 확인하기 위해 사용한다. 모듈이 존재하는지를 확인하기 위해 아래와 같이 한다.

```
// DIO 모듈이 있는지 확인한다.  
DWORD dwStatus;  
AxInfoIsDIOModule(&dwStatus);  
if(dwStatus == STATUS_EXIST)  
    printf("DIO 모듈이 존재합니다.");  
else  
    printf("DIO 모듈이 존재하지 않습니다.");
```

● 입출력 모듈 및 채널의 개수 확인

AxdInfoGetModuleCount: AxdInfoGetModuleCount 함수는 전체 시스템에 장착되어 있는 DIO모듈의 개수를 확인하는 함수이다.

```
// DIO 모듈의 개수를 확인한다.
long lCount;
AxdInfoGetModuleCount (&lCount);
printf ("DIO 모듈의 개수는 %d개 입니다.", lCount);
```

AxdInfoGetInputCount: AxdInfoGetInputCount함수는 지정한 DIO모듈에서 입력채널의 개수를 확인하는 함수이다.

```
// 0번째 모듈의 입력 채널 개수를 확인한다.
long lCount;
AxdInfoGetInputCount (0, &lCount);
printf ("0번째 DIO모듈의 입력 채널 개수는 %d개 입니다.", lCount);
```

AxdInfoGetOutputCount: AxdInfoGetOutputCount 함수는 지정한 DIO모듈에서 출력채널의 개수를 확인하는 함수이다.

```
// 0번째 모듈의 출력 채널 개수를 확인한다.
long lCount;
AxdInfoGetOutputCount (0, &lCount);
printf ("0번째 DIO모듈의 출력 채널 개수는 %d개 입니다.", lCount);
```

AxdInfoGetModuleNo : 초기화 되어 있는 전체 DIO 모듈 번호를 확인한다.

```
// 0번 보드, 0번째 모듈에의 모듈 번호를 확인
long lBoardNo=0, lModulePos=0, lModuleNo;
Code = AxdInfoGetModuleNo(lBoardNo, lModulePos, & lModuleNo);
printf ("0번 보드, 0번째 모듈의 모듈 번호 : %d\n", lModuleNo);
```

라이브러리의 종료

AxIClose : AxIClose 함수는 라이브러리를 종료하는 함수이다. 라이브러리를 사용한 후에는 마지막에 반드시 종료시켜서 할당된 메모리를 반환하여야 한다. 함수가 정상 실행되어 라이브러리가 종료되면 TRUE를 반환하고, 그렇지 않은 경우에는 FALSE를 반환한다.

```
// 라이브러리를 종료 한다.
if(AxIClose())
    printf("라이브러리가 종료 되었습니다.");
```

```
// Ex1_AXD_InitAndClose.cpp : Defines the entry point for the console application.
// 라이브러리를 초기화하고, 입출력 정보를 확인한 후 종료한다.
```

```
#include "stdafx.h"
#include "AXL.h"
#include <conio.h>
#include "stdio.h"
void main(void)
{
```

```
// 라이브러리를 초기화 한다.  
// 7은 IRQ를 뜻한다. PCI에서는 자동으로 IRQ가 설정된다.  
DWORD Code = Ax1Open(7);  
if(Code == AXT_RT_SUCCESS)  
{  
    printf("라이브러리가 초기화 되었습니다.\n");  
  
    // DIO 모듈이 있는지 검사  
    DWORD dwStatus;  
    Code = Ax1dInfoIsDIOModule(&dwStatus);  
    if(Code == AXT_RT_SUCCESS)  
    {  
        if(dwStatus == STATUS_EXIST)  
        {  
            printf("DIO모듈이 존재 합니다.\n");  
  
            // DIO모듈의 개수를 확인  
            long lModuleCounts;  
            Code = Ax1dInfoGetModuleCount(&lModuleCounts);  
            if (Code == AXT_RT_SUCCESS)  
                printf("DIO모듈의 개수 : %d\n",lModuleCounts);  
            else  
                printf("Ax1dInfoGetModuleCounts() : ERROR code 0x%x\n",Code);  
  
            // 입력 및 출력 점점의 개수를 확인  
            long lInputCounts;  
            long lOutputCounts;  
            for(int ModuleNo=0;ModuleNo<lModuleCounts;ModuleNo++)  
            {  
                Code = Ax1dInfoGetInputCount(ModuleNo,&lInputCounts);  
                if(Code == AXT_RT_SUCCESS)  
                {  
                    DWORD Code2 = Ax1dInfoGetOutputCount(ModuleNo,&lOutputCounts);  
                    if(Code2 == AXT_RT_SUCCESS)  
                    {  
                        printf("%d번째 모듈 : 입력점점 %d개, 출력점점 %d개\n", ModuleNo, lInputCounts,  
                               lOutputCounts);  
                    }  
                    else  
                        printf("Ax1dInfoGetOutputCounts() : ERROR code 0x%x\n",Code);  
                }  
                else  
                    printf("Ax1dInfoGetInputCounts() : ERROR code 0x%x\n",Code);  
            }  
        }  
        else  
            printf("Ax1dInfoIsDIOModule() : ERROR ( NOT STATUS_EXIST ) code 0x%x\n",Code);  
    }  
    else
```

```
        printf("AxdInfoIsDIOModule() : ERROR ( Return FALSE ) code 0x%x\n",Code);
    }
else
    printf("AxlOpen() : ERROR code 0x%x\n",Code);

// 라이브러리를 종료 한다.
if(AxlClose())
    printf("라이브러리가 종료 되었습니다.\n");
else
    printf("라이브러리가 정상적으로 종료되지 않았습니다.\n");
}
```

입출력 포트의 신호레벨 설정

입출력 포트를 사용하기 위해서는 먼저 각각의 입출력 접점의 신호 레벨을 설정해 주어야 한다. 이는 프로그램상에서 1이라는 값이 High인지 Low인지를 설정해주는 것이며, 입출력 접점의 신호레벨을 설정하기 위해서 아래의 그림에서와 같이 여러 접두어를 이용하여 총 20가지의 함수를 제공한다.

Offset의 설명

Offset이란 DIO 모듈의 입출력 접점들의 일련 번호를 의미하며 입력과 출력 각각 독립적인 일련번호를 가진다.

예를 들어 SIO-DB32 모듈의 경우 입력과 출력이 하나의 모듈에 16개씩 총 32개의 접점이 존재하는데. 각 인덱스는 Bit 단위로 신호를 접근 할 경우 입력 offset이 0~15의 범위를 가지며 출력 offset이 0~15의 Offset을 가진다.

만약, 시스템에 SIO-DB32 / SIO-DI32 / SIO-DO32 모듈이 하나씩 장착되어 있고 할당된 Module No가 각각 0,1,2 로 순차적이라면 각 모듈별 할당 Offset의 범위와 각 함수의 연관성은 아래 표와 같다.

입력 Offset

함수 모듈	SIO-DB32 (In16/Out16)	SIO-DI32 (In32)	Data Size	비고
AxdiLevelSetInport AxdiLevelGetInport	0 ~ 15	16~47	1	전체 모듈에서 Offset 적용됨
AxdiLevelSetInportBit AxdiLevelGetInportBit	0 ~ 15	0~31	1	각각의 모듈 Offset 적용
AxdiLevelSetInportByte AxdiLevelGetInportByte	0 ~ 1	0~3	8	각각의 모듈 Offset 적용
AxdiLevelSetInportWord AxdiLevelGetInportWord	0	0 ~ 1	16	각각의 모듈 Offset 적용
AxdiLevelSetInportDword AxdiLevelGetInportDword	0	0	32	각각의 모듈 Offset 적용

출력 Offset

함수 모듈	SIO-DB32 (In16/Out16)	SIO-DI32 (In32)	Data Size	비고
AxdoLevelSetOutport AxdoLevelGetOutport	0 ~ 15	16~47	1	전체 모듈에서 Offset 적용됨
AxdoLevelSetOutportBit AxdoLevelGetOutportBit	0 ~ 15	0~31	1	각각의 모듈 Offset 적용
AxdoLevelSetOutportByte AxdoLevelGetOutportByte	0 ~ 1	0~3	8	각각의 모듈 Offset 적용
AxdoLevelSetOutportWord AxdoLevelGetOutportWord	0	0 ~ 1	16	각각의 모듈 Offset 적용

함수 설명

AxdILevelSetInport: 전체 입력접점 모듈의 offset 번지에서 bit단위로 해당 bit의 신호레벨을 설정한다. 만약 PC에 여러 개의 DIO모듈이 장착되어 있다고 하면, 모듈 인덱스 순서대로 순차적으로 번호가 매겨진다. 예를 들어 SIO-DB32모듈이 2개 장착되어 있고, 첫번째 모듈에 있는 32개의 접점중 입력 접점 16개가 먼저 0번부터 15번까지 번호가 매겨지고, 다음모듈에 있는 32개의 접점중 입력 접점 16개가 16번부터 31번까지 번호가 매겨진다. 이 offset번지를 이용해서 해당 입력 접점의 신호레벨을 설정하는 함수가 AxdILevelSetInport 함수이다.

전체 입력 접점중에 20번 비트의 신호레벨을 HIGH로 설정하고자 한다면, 다음과 같이 한다.

```
// 전체 모듈, offset 20번지에서 bit 단위로 데이터 레벨을 HIGH로 설정한다.
AxdILevelSetInport(20, 1);
```

AxdOLevelSetOutport: 출력접점 신호레벨을 설정하는 함수로서 사용 방법은 입력 함수인 AxdILevelSetInport 와 동일하다.

AxdILevelSetInportBit: 사용자가 지정한 DIO모듈의 입력 접점 offset번지에서 bit단위로 해당 bit의 신호레벨을 설정한다. 입력 접점 offset번지는 앞에서 설명한 바와 같이 입력접점에 해당하는 일련번호인데, 특정 모듈을 지정하는 함수에서의 경우에는 하나의 모듈 내에서 입력 접점들에 붙여진 일련번호를 뜻한다. 예를 들어 ‘2’개의 SIO-DB32모듈이 장착되어 있다고 할 때 ‘0’번 모듈에 ‘0’번부터 ‘15’번까지의 입력 offset값이 있고, ‘1’번 모듈에 또한 0번부터 15번으로 개별적인 번호가 붙는다.

‘0’번 모듈에서 1번 비트의 신호레벨을 설정하고자 한다면 다음과 같이 한다.

```
// 0번째 모듈의 Offset 0번지에서 bit 단위로 데이터 레벨을 HIGH로 설정한다.
AxdILevelSetInportBit(0, 0, 1);
```

AxdOLevelSetOutportBit: 출력접점 신호레벨을 설정하는 함수로서 사용 방법은 입력접점 신호레벨을 설정하는 함수인 AxdILevelSetInportBit 와 동일하다.

AxdILevelSetInportByte: 사용자가 지정한 DIO모듈의 입력 offset번지에서 byte단위(동시 8개 접점)로 해당 bit의 신호레벨을 설정한다. offset번지는 해당 입력접점이 속해있는 포트 번호와 같으며. 순차적으로 ‘0’부터 ‘3’까지의 offset값이 있으며, 8bit의 byte단위로 신호레벨을 설정할 수 있다. SIO-DB32모듈의 경우 하나의 모듈에 유효한 Offset의 범위는 0 ~ 10I 된다.

‘0’번 모듈에서 ‘0’번 offset의 신호레벨을 설정하고자 한다면 다음과 같이 한다.

```
// 0번째 모듈의 Offset 0번지에서 byte 단위로 데이터 레벨을 모두 HIGH로 설정한다.
AxdILevelSetInportByte(0, 0, 0xFF);
```

AxdOLevelSetOutportByte: 출력접점 신호레벨을 설정하는 함수로서 사용 방법은 입력접점 신호레벨을 설정하는 함수인 AxdOLevelSetOutportByte 와 동일하다.

AxdILevelSetInportWord: AxdILevelSetInportWord 함수는 앞에서 설명한 AxdILevelSetInportByte 함수와 유사한데, 차이점이라면 word단위(16개 접점)로 접점의 신호레벨을 설정 한다는 것이다. 16개의 접점을 한 단위로 묶어 사용하므로, offset값은 ‘0’번과 ‘1’번 밖에 없다. SIO-DB32모듈의 경우 하나의 모듈에 유효한 Offset은 0번 밖에 없다. AxdSetPortLevelWord 함수에서도 마찬가지로 어느 특정 offset의 신호레벨을 설정하고자 한다면, 다음과 같이 한다.

```
// 0번째 모듈의 Offset 0번지에서 word 단위로 데이터 레벨을 모두 HIGH로 설정한다.
AxdILevelSetInportWord(0, 0, 0xFFFF);
```

[AxdoLevelSetOutportWord](#): 출력접점 신호레벨을 설정하는 함수로서 사용 방법은 입력접점 신호레벨을 설정하는 함수인 [AxdILevelSetImportWord](#) 와 동일하다.

[AxdILevelSetImportDword](#): AxdiLevelSetImportDword 함수는 앞에서 설명한 [AxdILevelSetImportWord](#) 함수와 유사한데, 차이점이라면 Dword단위(32개 접점)로 입력 접점의 신호레벨을 설정 한다는 것이다. 32개의 접점을 한 단위로 묶어 사용하므로, offset값은0번 밖에 없다.

[AxdILevelSetImportWord](#) 함수에서도 마찬가지로 어느 특정 offset의 신호레벨을 설정하고자 한다면, 다음과 같이 한다.

```
// 0번째 모듈의 Offset 0번지에서 dword 단위로 데이터 레벨을 모두 HIGH로 설정한다.  
AxdiLevelSetImportDword(0, 0, 0xFFFFFFFF);
```

[AxdoLevelSetOutportDword](#): 출력 접점 신호레벨을 설정하는 함수로서 사용 방법은 입력 함수인 [AxdILevelSetImportDword](#) 와 동일하다.

입출력포트 읽기

함수 설명

특정 DIO모듈에 있는 입출력 포트의 현재 상태를 확인하기 위해서 입력과 출력 각각 5개씩의 함수가 있다.

[AxdiReadInport](#): 시스템에 장착된 전체 DIO모듈의 입력 점점 offset번지에서 bit단위로 해당 bit의 정보를 읽어와서 1 또는 0을 반환한다.

0번 모듈에서 1번 비트의 값을 확인하고자 한다면 다음과 같이 한다.

```
// 0번째 모듈에서 offset 1번지에 데이터를 bit 단위로 읽는다.
DWORD dwValue;
AxdiReadInport(0, 1, &dwValue);
```

[AxdoReadOutport](#): 출력 점점의 상태를 읽는 함수로서 사용 방법은 입력 함수인 [AxdiReadInport](#) 와 동일하다.

[AxdiReadInportBit](#): 사용자가 지정한 DIO모듈의 입력 offset번지에서 bit단위로 해당 bit의 정보를 읽어와서 1 또는 0을 반환한다.

0번 모듈에서 1번 비트의 값을 확인하고자 한다면 다음과 같이 한다.

```
// 0번째 모듈에서 offset 1번지에 데이터를 bit 단위로 읽는다.
DWORD dwValue;
AxdiReadInportBit(0, 1, &dwValue);
```

[AxdoReadOutportBit](#): 출력 점점의 상태를 읽는 함수로서 사용 방법은 입력 함수인 [AxdoReadOutportBit](#) 와 동일하다.

[AxdiReadInportByte](#): 사용자가 지정한 모듈의 입력 offset 번지에서 byte 단위로 데이터를 읽는다. offset번지는 0,1,2,3의 값을 가진다. AxdiReadInportByte함수는 byte 단위의 값을 리턴 하므로 만약 특정 비트의 신호를 확인하고자 한다면 다음과 같이 비트연산을 하면 된다.

```
// 0번째 모듈의 Offset 0번지에서 byte 단위로 데이터를 읽는다.
DWORD dwValue;
AxdiReadInportByte(0, 0, &dwValue);

// 0번 모듈의 0번Input bit의 정보를 bit연산을 이용하여 확인한다.
BOOL Bit0 = dwValue&0x01 ? 1 : 0 ;
BOOL Bit1 = dwValue&0x02 ? 1 : 0 ;
BOOL Bit2 = dwValue&0x04 ? 1 : 0 ;
BOOL Bit3 = dwValue&0x08 ? 1 : 0 ;
```

[AxdoReadOutportByte](#): 출력 점점의 상태를 읽는 함수로서 사용 방법은 입력 함수인 [AxdiReadInportByte](#) 와 동일하다.

[AxdiReadInportWord](#): 사용자가 지정한 모듈의 입력 offset 번지에서 word 단위로 데이터를 읽는다. offset번지는 0,1의 값을 가진다. AxdiReadInportWord함수는 word 단위의 값을 리턴 하므로 만약 특정 비트의 신호를 확인하고자 한다면 다음과 같이 비트연산을 하면 된다.

```
// 0번째 모듈의 Offset 0번지에서 word 단위로 데이터를 읽는다.
DWORD dwValue;
AxdiReadInportWord(0, 0, &dwValue);
```

```
// 0번 모듈의1번Input bit의 정보를 bit연산을 이용하여 확인한다.
BOOL Bit1 = dwValue&0x0002 ? 1 : 0 ;
```

AxdoReadOutportWord: 출력접점의 상태를 읽는 함수로서 사용 방법은 입력접점의 상태를 읽는 함수인 AxdReadInportWord 와 동일하다.

AxdReadInportDword: 사용자가 지정한 모듈의 입력 offset 번지에서 dword 단위로 데이터를 읽는다. offset 번지는 0의 값을 가진다. AxdiReadInportDword함수는 dword 단위의 값을 리턴 하므로 만약 특정 비트의 신호를 확인하고자 한다면 다음과 같이 비트연산을 하면 된다.

```
// 0번째 모듈의 Offset 0번지에서 dword 단위로 데이터를 읽는다.
DWORD dwValue;
AxdiReadInportDword(0, 0, &dwValue);

// 0번 모듈의2번Input bit의 정보를 bit연산을 이용하여 확인한다.
BOOL Bit3 = dwValue&0x00000004 ? 1 : 0 ;
```

AxdoReadOutportDword: 출력접점의 상태를 읽는 함수로서 사용 방법은 입력접점의 상태를 읽는 함수인 AxdReadInportDword 와 동일하다.

```
// Ex2_AXD_ReadPort.cpp : Defines the entry point for the console application.
// DIO모듈의 입력 포트에 대한 정보를 확인한다.
// 0 번 모듈이 32 입력 접점 모듈로 가정한다.

#include "stdafx.h"
#include "AXL.h"
#include <conio.h>
#include "stdio.h"

void main(void)
{
    // 라이브러리를 초기화 한다.
    // 7은 IRQ를 뜻한다. PCI의 경우 자동으로 IRQ가 설정된다.
    DWORD Code = AxlOpen(7);
    if(Code == AXT_RT_SUCCESS)
    {
        printf("라이브러리가 초기화 되었습니다.\n");

        // DIO 모듈이 있는지 검사
        DWORD dwStatus;
        Code = AxdInfoIsDIOModule(&dwStatus);
        if(Code == AXT_RT_SUCCESS)
        {
            if(dwStatus == STATUS_EXIST)
            {
                // DIO모듈의 개수를 확인
                long lModuleCounts;
                AxdInfoGetModuleCount (&lModuleCounts);
                // DIO모듈의 입출력 신호레벨을 High로 설정한다.
            }
        }
    }
}
```

```

for(int ModuleNo=0;ModuleNo<lModuleCounts;ModuleNo++)
{
    if(AxdInfoGetInputCount (ModuleNo) > 0)
    {
        AxdiLevelSetImportDword (ModuleNo,0,0xffffffff);
    }
    if(AxdInfoGetOutputCount (ModuleNo) > 0)
    {
        AxdoLevelSetOutportDword (ModuleNo,0,0xffffffff);
    }
}

// 무한루프를 돌면서 Port의 현재 상태를 반환한다.
printf("INFORMATION*****\n");
printf("아무키나 누르면 종료 합니다. \n");
printf("*****\n");
printf("\n\n0번 모듈의 입력 포트의 현재 상태\n");

BOOL fExit = FALSE;
DWORD Status_Module0[32];
while(!fExit)           // 무한 루프
{
    if(kbhit())          // 아무키나 누르면 무한루프를 빠져나간다.
        fExit = TRUE;

    // bit 단위로 해당 offset의상태 확인
    AxdiReadImportBit(0,0,&Status_Module0[0]);
    AxdiReadImportBit(0,1,&Status_Module0[1]);
    // byte 단위로 해당 offset 상태 확인
    AxdiReadImportByte(0,0,&Status_Module0[2]);
    AxdiReadImportByte(0,0,&Status_Module0[3]);
    // word 단위로 해당 offset 상태 확인
    AxdiReadImportWord(0,0,&Status_Module0[4]);
    AxdiReadImportWord(0,0,&Status_Module0[5]);

    // double word 단위로 해당 offset 상태 확인
    AxdiReadImportDword(0,0,&Status_Module0[16]);
    AxdiReadImportDword(0,0,&Status_Module0[17]);
    // byte 단위로 해당 offset 상태 확인
    AxdiReadImportByte(0,3,&Status_Module0[18]);
    AxdiReadImportByte(0,3,&Status_Module0[19]);
    // word 단위로 해당 offset 상태 확인
    AxdiReadImportWord(0,1,&Status_Module0[20]);

    // Input Port의 현재상태를 반환한다.
    printf("\r0[%d],1[%d],2[%d],3[%d],4[%d].. 18[%d],19[%d],20[%d]...",
        Status_Module0[0],Status_Module0[1], Status_Module0[2]&0x04 ? 1 : 0,
        Status_Module0[3]&0x08 ? 1 : 0, Status_Module0[4]&0x0010 ? 1 : 0,
        Status_Module0[5]&0x0020 ? 1 : 0, Status_Module0[16]&0x00010000 ? 1 : 0,

```

```
        Status_Module0[17]&0x00020000 ? 1 : 0, Status_Module0[18]&0x04 ? 1 : 0,
        Status_Module0[19]&0x08 ? 1 : 0, Status_Module0[20]&0x0010 ? 1 : 0);
    }
}
else
    printf("AxdInfoIsDIOModule() : ERROR ( NOT STATUS_EXIST ) code 0x%x\n",Code);
}
else
    printf("AxdInfoIsDIOModule() : ERROR ( Return FALSE ) code 0x%x\n",Code);
}
else
    printf("Ax1Open() : ERROR code 0x%x\n",Code);

// 라이브러리를 종료 한다.
if(Ax1Close())
    printf("라이브러리가 종료 되었습니다.\n");
else
    printf("라이브러리가 정상적으로 종료되지 않았습니다.\n");
}
```

출력포트 쓰기

Output Port는 모듈에서부터 신호를 출력하는 포트로서 SIO-DO32모듈에는 32개의 출력접점이 존재하고, SIO-DB32모듈에는 16개의 출력접점이 존재한다. Output Port를 이용하여 신호를 출력하기 위해 다음과 같이 5 가지의 함수가 제공된다.

각 함수의 할당 Offset의 의미와 범위는 단위 별로 1.4.1 항에 설명된 Level 설정 함수와 동일하다.

함수 설명

AxdWriteOutportBit: 사용자가 지정한 DIO모듈의 출력 offset번지에서 bit단위로 해당 출력접점으로부터 출력을 내보낸다.

0번 모듈에서 1번 출력접점을 통해 출력을 내보내고자 한다면 다음과 같이 한다.

```
// 0번째 모듈에서 1번 출력접점 (Offset : 1)에 데이터를 bit 단위로 출력한다.
AxdWriteOutportBit(0, 1, 1);
```

AxdWriteOutportByte: 사용자가 지정한 모듈의 출력 offset 번지에서 byte 단위로 데이터를 출력한다. offset 번지는 0 ~ 3의 값을 가진다. '0'번째 모듈의 출력 Offset 2번지(16번째 접점부터 8개접점 출력)에서 byte 단위로 데이터를 출력하고자 한다면 다음과 같이 한다.

```
// 0번째 모듈의 Offset 2번지에서 byte 단위로 데이터를 출력한다.
AxdWriteOutportByte(0, 2, 0xFF);
```

AxdWriteOutportWord: 사용자가 지정한 모듈의 출력 offset 번지에서 word 단위로 데이터를 출력한다. offset번지는 0 ~ 1의 값을 가진다. 0번째 모듈의 출력 Offset 0번지(0번째 접점부터 16개 접점 출력)에서 word 단위로 데이터를 출력하고자 한다면 다음과 같이 한다.

```
// 0번째 모듈의 Offset 1번지에서 word 단위로 데이터를 읽는다.
AxdWritePortWord(0, 0, 0xFFFF);
```

AxdWriteOutportDword: 사용자가 지정한 모듈의 offset 번지에서 Dword 단위로 데이터를 출력한다. offset 번지는 0의 값을 가진다. 32개의 출력 접점을 한번에 출력한다.

'0'번째 모듈의 Offset '0'번지에서 Dword 단위로 데이터를 출력하고자 한다면 다음과 같이 한다.

```
// 0번째 모듈의 Offset 0번지에서 dword 단위로 데이터를 출력한다.
AxdWritePortDword(0, 0, 0xFFFFFFFF);
```

AxdWriteOutport: 시스템에 장착된 전체 출력접점 모듈의 offset 번지에서 bit단위로 해당 bit에 출력을 내보낸다. 전체 출력접점중에 15번째 출력접점에 Bit 단위로 데이터를 출력하고자 한다면 다음과 같이 한다.

```
// 전체 출력접점 중에서 15번째 출력접점에 '1'을 출력한다.
AxdWriteOutport(15,1);
```

```
// Ex4_AXD_WriteOutput.cpp : Defines the entry point for the console application.
// DIO모듈의 출력 포트에 출력을 내보내고 현재 상태를 확인한다.
```

```

#include "stdafx.h"
#include "AXL.h"
#include <conio.h>
#include "stdio.h"

void main(void)
{
    // 라이브러리를 초기화 한다.
    // 7은 IRQ를 뜻한다. PCI에서는 자동으로 IRQ가 설정된다.
    DWORD Code = Ax1Open(7);
    if(Code == AXT_RT_SUCCESS)
    {
        printf("라이브러리가 초기화 되었습니다.\n");

        // DIO 모듈이 있는지 검사
        DWORD dwStatus;
        Code = AxdInfoIsDIOModule(&dwStatus);
        if(Code == AXT_RT_SUCCESS)
        {
            if(dwStatus == STATUS_EXIST)
            {
                // DIO모듈의 개수를 확인
                long lModuleCounts;
                AxdInfoGetModuleCount(&lModuleCounts);

                // DIO모듈의 입출력 신호레벨을 High로 설정한다.
                for(int ModuleNo=0;ModuleNo<lModuleCounts;ModuleNo++)
                {
                    if(AxdInfoGetInputCount(ModuleNo) > 0)
                    {
                        AxdiLevelSetImportDword(ModuleNo,0,0xffffffff);
                    }
                    if(AxdInfoGetOutputCount(ModuleNo) > 0)
                    {
                        AxdoLevelSetOutportDword(ModuleNo,0,0xffffffff);
                    }
                }

                // 무한루프를 돌면서 Output Port의 현재 상태를 반환한다.
                printf("[INFORMATION]*****\n");
                printf("[ESC] : Exit \n");
                printf("[0 ~ 4] : 0 ~ 4번 출력접점(0번 모듈의0~4번 출력접점) On / Off \n");
                printf("[5 ~ 9] : 16 ~ 20번 출력접점(1번 모듈의0~4번 출력접점) On / Off \n";
                printf("*****\n");
                printf("\n\nOutput Port의 현재 상태\n");

                BOOL fExit = FALSE;
                while(!fExit)           // 무한 루프
                {

```

```
if(kbhit())          // 아무키나 누르면
{
    int ch = getch();
    DWORD OutPort;
    switch(ch)
    {
        case 27:           // Esc key
            fExit = TRUE;
            break;

        // 전체 출력비트에서 해당 offset번지를 On/Off
        case '0':
            AxdoReadOutport (0,&OutPort);
            if(OutPort) AxdoWriteOutport (0,0);
            else AxdoWriteOutport (0,1);
            break;

        case '1':
            AxdoReadOutport (1,&OutPort);
            if(OutPort) AxdoWriteOutport (1,0);
            else AxdoWriteOutport (1,1);
            break;

        case '2':
            AxdoReadOutport (2,&OutPort);
            if(OutPort) AxdoWriteOutport (2,0);
            else AxdoWriteOutport (2,1);
            break;

        case '3':
            AxdoReadOutport (3,&OutPort);
            if(OutPort) AxdoWriteOutport (3,0);
            else AxdoWriteOutport (3,1);
            break;

        case '4':
            AxdoReadOutport (4,&OutPort);
            if(OutPort) AxdoWriteOutport (4,0);
            else AxdoWriteOutport (4,1);
            break;

        case '5':
            AxdoReadOutport (16,&OutPort);
            if(OutPort) AxdoWriteOutport (16,0);
            else AxdoWriteOutport (16,1);
            break;

        case '6':
            AxdoReadOutport (17,&OutPort);
```

```

        if(OutPort) AxdoWriteOutport (17,0);
        else AxdoWriteOutport (17,1);
    break;

    case '7':
        AxdoReadOutport (18,&OutPort);
        if(OutPort) AxdoWriteOutport (18,0);
        else AxdoWriteOutport (18,1);
    break;

    case '8':
        AxdoReadOutport (19,&OutPort);
        if(OutPort) AxdoWriteOutport (19,0);
        else AxdoWriteOutport (19,1);
    break;

    case '9':
        AxdoReadOutport (20,&OutPort);
        if(OutPort) AxdoWriteOutport (20,0);
        else AxdoWriteOutport (20,1);
    break;
}
}

// Output Port의 현재상태를 반환한다.
DWORD Status_Output[32];
for(int OffsetNo=0;OffsetNo<32;OffsetNo++)
{
    Status_Output[OffsetNo] = 0;
    AxdoReadOutport (OffsetNo,&Status_Output[OffsetNo]);
}
printf("\r0[%d],1[%d],2[%d],..... 17[%d],18[%d],19[%d],20[%d]...",
    Status_Output[0], Status_Output[1], Status_Output[2], Status_Output[3],
    Status_Output[4], Status_Output[16], Status_Output[17], Status_Output[18],
    Status_Output[19], Status_Output[20]);
}

else
    printf("AxdInfoIsDIOModule() : ERROR ( NOT STATUS_EXIST ) code 0x%x\n",Code);
}
else
    printf("AxdInfoIsDIOModule() : ERROR ( Return FALSE ) code 0x%x\n",Code);
}
else
    printf("AxlOpen() : ERROR code 0x%x\n",Code);

// 라이브러리를 종료 한다.
if(AxlClose())
    printf("\n라이브러리가 종료 되었습니다.\n");

```

```
    else  
        printf("\n라이브러리가 정상적으로 종료되지 않았습니다.\n");  
}
```

인터럽트

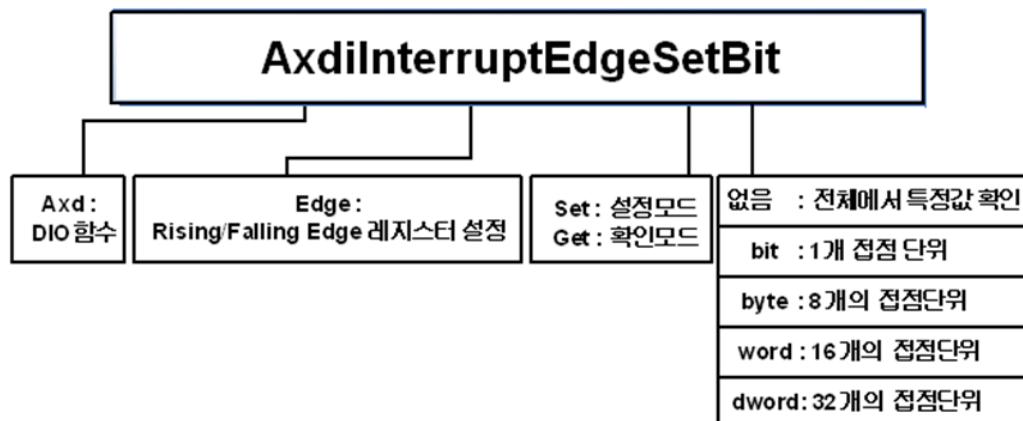
AXD에서는 입력접점에 대해서 Rising Edge(Up Edge) 와 Falling Edge(Down Edge)에 대해서 각각 인터럽트를 설정하여 활용할 수 있도록 함수들을 제공한다. 인터럽트를 사용하기 위해서는 먼저 Interrupt Rising Edge Register와 Interrupt Falling Edge Register를 이용하여 인터럽트를 발생시킬 위치를 설정하고, 인터럽트를 어떤 방식으로 처리할것인가를 설정한다. AXD에서는 인터럽트를 처리하는 방식으로 콜백함수 방식, 원도우 메시지 방식, 이벤트 방식의 3가지 방식을 지원하므로, 사용자의 개발환경에 맞추어 선택하면된다. 마지막으로 해당 모듈의 인터럽트의 사용 유무를 ENABLE로 설정하고 AXL 전체 인터럽트 사용 유무를 ENABLE(AxIIInterruptEnable 함수 사용) 하면 된다. 인터럽트 결과는 메시지 핸들러 및 콜백 함수내에서 리턴 값으로 확인하며,Event 방식에서는 해당 이벤트 발생 후 AxdiInterruptRead 함수를 사용하여 확인할 수 있다.

Interrupt Rising / Falling Edge의 설정

Interrupt Rising / Falling Edge Register는 해당 입력접점의 Rising Edge 또는 Falling Edge에서 인터럽트를 발생시킬지 여부를 설정하는 레지스터이다. 이 레지스터의 해당 비트를 1로 설정하면 해당 Edge가 발생할 때 인터럽트가 발생된다. 예를 들어 offset번지 '0'번의 Interrupt Rising Edge Register를 1로 설정하였다면, 0번 입력접점에 Rising Edge가 발생할 때 인터럽트가 발생하게 된다.

Rising Edge와 Falling Edge는 상대적인 개념으로 만약 신호레벨을 HIGH로 설정해두었다면 신호가 없을때는 LOW였다가 신호가 들어올 때 HIGH로 바뀌면서 Rising Edge가 발생하고, 다시 신호가 사라지면 LOW가 되면서 Falling Edge가 발생하게 된다. 만약 신호레벨이 LOW로 설정되었다면, 이와 반대로 신호가 들어올때 Falling Edge가 발생하고, 신호가 사라질때 Rising Edge가 발생한다.

이러한 레지스터를 설정하거나 설정된 값을 확인하는 함수로는 전체 입력접점에서 특정 offset주소에 해당하는 비트를 설정 또는 확인하는 함수와, 특정 모듈에서의 특정 offset번지에 해당하는 입력접점을 bit단위, byte단위, word단위, dword단위로 설정 또는 확인 하는 함수로 구성된다.



AxdilInterruptEdgeSet: AxdiInterruptEdgeSet 함수는 전체 입력접점에서 특정 offset번지에 있는 입력접점에서 Rising 또는 Falling Edge가 발생할 때 인터럽트가 발생하도록 레지스터를 설정하는 함수이다. offset번지는 '0'번부터 전체 입력접점의 수 - 1까지 있는데, 예를 들어 2개의 SIO-DB32모듈이 설치되어 있다면, 0번 모듈의 16개 입력 접점들은 차례로 0 ~ 15까지의 offset값을 갖고, 1번 모듈에 있는 16개의 입력접점들은 차례로 16 ~ 31까지의 offset값을 가지게 된다. 설정하고자 하는 Edge에 따라 UP_EDGE 또는 DOWN_EDGE를

입력하고, 설정값은 ‘1’로 하면 인터럽트를 발생시키고, ‘0’으로 하면 인터럽트를 발생시키지 않는다.

전체 입력접점중에 2번째 입력접점의 Rising Edge에서 인터럽트를 발생시키고자 한다면 다음과 같이 한다.

```
// 전체 입력접점중에 2번째 입력접점의 UpEdge에 인터럽트를 설정한다.
AxdiInterruptEdgeSet(2, UP_EDGE, 1);
```

[AxdiInterruptEdgeSetBit](#): AxdiInterruptEdgeSetBit함수는 특정 모듈에서 특정 offset번지에 있는 입력접점에서 Rising 또는 Falling Edge가 발생할 때 인터럽트가 발생하도록 레지스터를 설정하는 함수이다. offset번지는 ‘0’번부터 해당모듈의 입력접점의 수 - 1까지 있는데, 예를 들어 2개의 SIO-DB32모듈이 설치되어 있다면, ‘0’번 모듈의 16개 입력 접점들은 차례로 0 ~ 15까지의 offset값을 갖고, 1번 모듈에 있는 16개의 입력접점들 또한 차례로 0 ~ 15까지의 offset값을 가지게 된다. 앞에서와 마찬가지로 설정하고자 하는 Edge에 따라 UP_EDGE 또는 DOWN_EDGE를 입력하고, 설정값을 ‘1’로 하면 인터럽트를 발생시키고, ‘0’으로 하면 인터럽트를 발생시키지 않는다.

‘0’번 모듈에서 ‘2’번째 입력접점의 Falling Edge에서 인터럽트를 발생시키고자 한다면 다음과 같이 한다.

```
// 0번 모듈의 입력접점중에 2번째 입력접점의 UpEdge에 인터럽트를 설정한다.
AxdiInterruptEdgeSetBit(0, 2, DOWN_EDGE, 1);
```

특정 모듈에서 특정 offset번지에 해당하는 입력접점들의 인터럽트 설정을 bit단위로 할 수도 있지만, 8개의 접점으로 구성된 byte 단위, 16개의 접점으로 구성된 word 단위, 32개의 접점으로 구성된 dword단위로도 설정할 수 있다. Default 값은 ‘0’으로 설정되어 있으므로, 사용할 비트들만 1로 설정하여 사용하면 된다.

만약 0번 모듈이 SIO-DB32모듈이고, 출수 번호의 입력접점에서 Rising Edge가 발생할때만 인터럽트를 발생 시키고자 한다면 다음과 같이 한다.

```
// 0번 모듈의 입력접점중에 출수번호 입력접점의 UpEdge에 인터럽트를 설정한다.
AxdiInterruptEdgeSetWord(0, 0, UP_EDGE, 0xAAAA);
```

Interrupt Rising / Falling Edge의 설정 값 확인

인터럽트 상승/하강 에지 레지스터에 설정된 값을 확인하기 위해서 제공되는 함수도 위에서 설명한 것과 같이 전체 입력비트에서 특정 offset번지에 해당하는 입력접점의 Rising Edge에 대한 설정을 확인하는 [AxdiInterruptEdgeGet](#)함수와 특정 모듈에서의 특정 offset번지에 해당하는 값을 bit단위로 확인하는 [AxdiInterruptEdgeGetBit](#)함수, 8개의 접점으로 구성된 byte단위로 확인하는 [AxdiInterruptEdgeGetByte](#)함수, 16개의 접점으로 구성된 word단위로 확인하는 [AxdiInterruptEdgeGetWord](#)함수, 32개의 접점으로 구성된 dword 단위로 확인하는 [AxdiInterruptEdgeGetDword](#) 함수들이 있다.

[AxdiInterruptEdgeGet](#): 전체 입력접점중에 2번째 입력접점의 Rising Edge Register의 설정값을 확인하고자 한다면 다음과 같이 한다. offset값은 AxdiInterruptEdgeSet함수에서와 마찬가지로 ‘0’부터 전체접점수 - 1까지의 값을 순서대로 가진다.

```
// 전체 입력접점중에 2번째 입력접점의 RisingEdge Register의 설정값을 확인한다.
DWORD dwEdge;
AxdiInterruptEdgeGet(2, UP_EDGE, &dwEdge);
```

[AxdiInterruptEdgeGetBit](#): 0번 모듈의 입력접점중에 2번째 입력접점의 Falling Edge Register의 설정값을 bit단위로 확인하고자 한다면 다음과 같이 한다. offset값은 AxdiInterruptEdgeSetBit함수에서와 마찬가지로 0부터 해당 모듈의 접점수-1까지의 값을 순서대로 가진다.

```
// 0번 모듈의 입력접점중에 2번째 입력접점의 FallingEdge Register의 설정값을 확인한다.
DWORD dwEdge;
AxdiInterruptEdgeGetBit(0, 2, DOWN_EDGE, &dwEdge);
```

그 외 [AxdiInterruptEdgeGetByte](#), [AxdiInterruptEdgeGetWord](#), [AxdiInterruptEdgeGetDword](#) 함수들의 offset값들은 각각 Set함수와 동일하므로, 참고하여 사용하면 된다.

인터럽트 사용여부의 설정

인터럽트의 사용여부는 Default로 사용하지 않음(DISABLE)으로 설정되어 있다. 그러므로 사용함(ENABLE)으로 설정을 해 주어야 상승/하강에지 레지스터에 설정한 대로 인터럽트가 발생한다. 인터럽트의 사용 여부를 설정하는 함수는 [AxdiInterruptSetModuleEnable](#) 함수이다.

만약 '0'번 모듈에서 인터럽트를 사용함으로 설정하고자 한다면 다음과 같이 한다.

```
// 0번 모듈에서 인터럽트를 사용함으로 설정
AxdiInterruptSetModuleEnable(0, ENABLE);
```

인터럽트 처리 방식의 설정

인터럽트가 발생했을 때 이를 처리하는 방법으로 AXD에서는 콜백함수 방식, 원도우 메시지 방식, 이벤트 방식의 3가지를 제공한다. 콜백 함수 방식은 이벤트 발생 시점에 즉시 콜백 함수가 호출됨으로 가장 빠르게 이벤트를 통지받을 수 있는 장점이 있으나 콜백 함수가 완전히 종료될 때까지 메인 프로세스가 정체되어 있게 된다. 즉, 콜백 함수 내에 부하가 걸리는 작업이 있을 경우에는 사용에 주의를 요한다. 그러나, 이벤트 방식은 이벤트의 발생 여부를 감시하는 특정 쓰레드를 사용하여 메인 프로세스와 별개로 동작되므로 MultiProcessor 시스템등에서 자원을 가장 효율적으로 사용할 수 있게 되어 특히 권장하는 방식이다.

이벤트 처리 방식을 설정하기 위해서는 [AxdiInterruptSetModule](#)함수를 사용하여 다음과 같이 설정해주면 된다.

1) 콜백함수 방식

콜백함수 방식을 사용하기 위해서는 아래와 같이 설정한다.

```
// 0번 모듈의 이벤트 처리 방식을 콜백함수 방식으로 설정한다.
AxdiInterruptSetModule(0, NULL, NULL, OnDIOInterruptCallback, NULL);
```

OnDIOInterruptCallback은 인터럽트를 처리할 콜백함수의 포인터이다. 콜백함수는 전역함수로 다음과 같이 설정하면 된다. 인터럽트가 발생하면 자동으로 인터럽트를 처리할 콜백 함수를 호출하게 된다.

```
void CExDIOInterruptDlg::OnBtnStartInterruptCallback()
{
    // 인터럽트 처리 방식 설정
    AxdiInterruptSetModule(MODULE_NO, NULL, NULL, OnDIOInterruptCallback, NULL);

    // 특정 모듈에서 Dword 단위로 인터럽트 마스크를 설정
    AxdiInterruptEdgeSetDword(MODULE_NO, 0, UP_EDGE, 0xFFFFFFFF);

    // 인터럽트 허용여부 설정
    AxdiInterruptSetModuleEnable(MODULE_NO, ENABLE);
```

```

}

// 인터럽트를 처리할 콜백함수
void OnDIOInterruptCallback(long nModuleNo, DWORD uFlag)
{
    // nModuleNo : 인터럽트 발생 모듈 번호, uFlag : 인터럽트 발생 Flag

    DWORD dwValue;
    CString IntMessage;
    Int InputCounts = 32;      // 전체 입력점점의 수

    for(int ChkBit=0; ChkBit<InputCounts; ChkBit++)
    {
        if((uFlag >> ChkBit) & 0x01)
        {
            AxdiReadInportBit(nModuleNo, ChkBit, &dwValue);
            if(dwValue)
                printf("INTERRUPT(MESSAGE) :\n %d번 모듈의 %d번째 입력 Bit에서 Rising 인터럽트 발생",
                       nModuleNo,ChkBit);
            else
                printf("INTERRUPT(MESSAGE) :\n %d번 모듈의 %d번째 입력 Bit에서 Falling 인터럽트 발생",
                       nModuleNo,ChkBit);
        }
    }
}

```

2) 윈도우 메시지 방식

윈도우 메시지 방식을 사용하기 위해서는 아래와 같이 설정한다. C에서는 윈도우 메시지방식을 사용하지 못하므로 이부분은 VC++을 사용하여 설명한다. 윈도우 메시지 방식을 사용하기 위해서는 메시지를 받을 원도우 헤들과 원도우 메시지를 입력한다. 윈도우 메시지를 사용하지 않거나 디폴트 값을 사용하고자 한다면 0을 입력한다.

```

// 0번 모듈의 이벤트 처리 방식을 윈도우 이벤트 방식으로 설정한다.
AxdiInterruptSetModule(lModuleNo,this->m_hWnd,WM_DIO_INTERRUPT,NULL,NULL);

```

사용방법을 전체적으로 살펴보면 아래와 같으며, OnDIOInterruptMessage함수에서 인터럽트를 처리한다.

```

#define WM_DIO_INTERRUPT 1011    // 인터럽트 메시지를 받아오기위한 메세지 정의
#define MODULE_NO 0              // DIO모듈 번호

...(중략)...

// 메시지와 함수를 연결
BEGIN_MESSAGE_MAP(CExDIOInterruptDlg, CDialog)
    //{{AFX_MSG_MAP(CExDIOInterruptDlg)
        ON_MESSAGE(WM_DIO_INTERRUPT,OnDIOInterruptMessage)
    }}AFX_MSG_MAP(CExDIOInterruptDlg)

```

```

// } }AFX_MSG_MAP
END_MESSAGE_MAP()

... (중략)...

void CExDIOInterruptDlg::OnBtnStartInterruptMessage()
{
    // 인터럽트 처리 방식 설정 (윈도우 메세지 방식)
    AxdiInterruptSetModule(MODULE_NO, this->m_hWnd, WM_DIO_INTERRUPT, NULL, NULL);

    // 0번 모듈에서 Dword 단위로 인터럽트 마스크를 설정
    AxdiInterruptEdgeSetDword(MODULE_NO, 0, UP_EDGE, 0xFFFFFFFF);

    // 인터럽트 허용여부 설정
    AxdiInterruptSetModuleEnable(MODULE_NO, ENABLE);
}

void CExDIOInterruptDlg::OnDIOInterruptMessage(WPARAM nModule, LPARAM nIntrFlag)
{
    // nModule : 인터럽트 발생 모듈 번호 ( 0, 1, 2, 3 )
    // nIntrFlag : 인터럽트 발생 Flag ( Input Port에서 인터럽트 발생 확인 )

    DWORD dwValue;
    CString IntMessage;
    int InputCounts = 16; // 입력 Bit의 총 개수

    for(int ChkBit=0; ChkBit<InputCounts; ChkBit++)
    {
        if((nIntrFlag >> ChkBit) & 0x01)
        {
            AxdiReadImportBit(nModule, ChkBit, &dwValue);
            if(dwValue)
                IntMessage.Format("INTERRUPT (MESSAGE) : %d번 모듈의 %d번째 입력 Bit에서 Rising 인터럽트 발생", nModule, ChkBit);
            else
                IntMessage.Format("INTERRUPT (MESSAGE) : %d번 모듈의 %d번째 입력 Bit에서 Falling 인터럽트 발생", nModule, ChkBit);
            m_LIST_Message.InsertItem(0, IntMessage);
        }
    }
}

```

3) 이벤트 방식

이벤트 방식을 사용하기 위해서는 아래와 같이 설정한다. 이벤트 방식은 이벤트의 발생 여부를 감시하는 특정 쓰레드를 사용하여 메인 프로세스와 별개로 동작되므로 MultiProcessor 시스템등에서 자원을 가장 효율적으로 사용할 수 있게 되어 특히 권장하는 방식이다.

이부분 또한 VC++을 사용하여 설명한다. 이벤트 방식을 사용하기 위해서는 이벤트 ID를 받을 HANDLE형 멤

버변수 `m_hInterruptEvent`를 미리 선언해두고 사용하여야 한다.

```
// 0번 모듈의 이벤트 처리 방식을 윈도우 이벤트 방식으로 설정한다.
AxdiInterruptSetModule(0,NULL,NULL,NULL,&m_hInterruptEvent);
```

사용 방법을 전체적으로 살펴보면 아래와 같으며, `ThreadProc`함수에서 인터럽트를 처리한다. 여기서 주의 할 점은 반드시 [AxdiInterruptSetModule](#)함수를 쓰레드가 실행되기 전에 호출하여야 한다는 것이다.

인터럽트가 발생하면 인터럽트 이벤트가 발생하게 되고 쓰레드는 이 이벤트를 감지하여 인터럽트가 발생했다는 것을 알게 된다. 그리고 쓰레드 내부에서 `AxdReadInterrupt` 함수를 호출하여 현재 인터럽트가 발생한 모듈 번호와 인터럽트 플래그 정보를 확인하여 어느 모듈의 어느 입력 Bit에 인터럽트가 발생하였는지를 확인 한다.

```
// 이벤트방식 인터럽트 사용에 필요한 변수
HANDLE m_hThreadHandle;
BOOL m_bThread;
HANDLE m_hInterruptEvent;
```

```
void CExDIOInterruptDlg::OnBtnStartInterruptEvent()
{
    // 인터럽트 처리 방식 설정
    AxdiInterruptSetModule(0,NULL,NULL,NULL,&m_hInterruptEvent); // 이벤트 방식

    // 특정 모듈에서 Dword 단위로 인터럽트 마스크를 설정
    AxdiInterruptEdgeSetDword(MODULE_NO,0,UP_EDGE,0xFFFFFFFF);

    // 인터럽트 허용여부 설정
    AxdiInterruptSetModuleEnable(MODULE_NO,ENABLE);

    // 이벤트 방식을 사용하기 위해서 쓰레드를 만든다.
    m_bThread = TRUE;
    CreateThread(NULL, 0, (LPTHREAD_START_ROUTINE)ThreadProc, this, NULL, NULL);
}

void ThreadProc(LPVOID lpData)
{
    CExDIOInterruptDlg *pDlg = (CExDIOInterruptDlg *)lpData;
    long lModuleNo;
    DWORD dwFlag;
    DWORD dwValue;
    CString IntMessage;
    int InputCounts = 32; // 입력 Bit의 총 개수

    while(pDlg->m_bThread)
    {
        if(WaitForSingleObject(pDlg->m_hInterruptEvent, 0) == WAIT_OBJECT_0)
        {
            // 이벤트 방식일 경우 인터럽트 정보를 읽어온다.
            AxdiInterruptRead(&lModuleNo, &dwFlag);
```

```

        for(int ChkBit=0; ChkBit<InputCounts; ChkBit++)
        {
            if((dwFlag >> ChkBit) & 0x01)
            {
                AxdiReadInportBit(lModuleNo, ChkBit, &dwValue);
                if(dwValue)
                    IntMessage.Format("INTERRUPT(MESSAGE) : %d번 모듈의 %d번째 입력 Bit에서 Rising 인
                                      터럽트 발생", lModuleNo,ChkBit);
                else
                    IntMessage.Format("INTERRUPT(MESSAGE) : %d번 모듈의 %d번째 입력 Bit에서 Falling
                                      인터럽트 발생", lModuleNo,ChkBit);
                pDlg->m_LIST_Message.InsertItem(0,IntMessage);
            }
        }
    }
}

```

```

// Ex5_AXD InterruptUsing.cpp : Defines the entry point for the console application.
// 인터럽트를 발생시키고, 현재 상태를 확인한다.

```

```

#include "stdafx.h"
#include "stdio.h"
#include <conio.h>
#include "AXL.h"

// 인터럽트를 처리할 콜백함수
void OnDIOInterruptCallback(long nModuleNo, DWORD uFlag)
{
    // nModuleNo : 인터럽트 발생 모듈 번호, uFlag : 인터럽트 발생 Flag

    DWORD dwValue;
    int InputCounts = 32;      // 전체 입력점점의 수

    for(int ChkBit=0; ChkBit<InputCounts; ChkBit++)
    {
        if((uFlag >> ChkBit) & 0x01)
        {
            AxdiReadInportBit(nModuleNo, ChkBit, &dwValue);
            if (dwValue)
                printf("\n %d번 모듈의 %d번째 입력 Bit에서 Rising 인터럽트 발생", nModuleNo,ChkBit);
            else
                printf("\n %d번 모듈의 %d번째 입력 Bit에서 Falling 인터럽트 발생", nModuleNo,ChkBit);
        }
    }
}

void main()

```

```
{  
    // 라이브러리를 초기화 한다.  
    // 7은IRQ를 뜻한다. PCI에서 자동으로 IRQ가 설정된다.  
    DWORD Code = Ax1Open(7);  
    if(Code == AXT_RT_SUCCESS)  
    {  
        printf("라이브러리가 초기화 되었습니다.\n");  
  
        // DIO 모듈이 있는지 검사  
        DWORD dwStatus;  
        Code = AxdInfoIsDIOModule(&dwStatus);  
        if(Code == AXT_RT_SUCCESS)  
        {  
            if(dwStatus == STATUS_EXIST)  
            {  
                // DIO모듈의 개수를 확인  
                long lModuleCounts;  
                int ModuleNo;  
                AxdInfoGetModuleCount(&lModuleCounts);  
  
                // DIO모듈의 출력 신호레벨을 High로 설정한다.  
                for(ModuleNo=0; ModuleNo < lModuleCounts; ModuleNo++)  
                    AxdSetPortLevelDword(ModuleNo, 0, 0xffffffff);  
  
                printf("[INFORMATION]*****\n");  
                printf("[ESC] : Exit \n");  
                printf("*****\n\n");  
  
                // 인터럽트 설정  
                for(ModuleNo=0; ModuleNo < lModuleCounts; ModuleNo++)  
                {  
                    // 인터럽트 처리 방식 설정 (콜백함수 방식)  
                    AxdiInterruptSetModule(ModuleNo, NULL, NULL, OnDIOInterruptCallback, NULL);  
                    // 특정 모듈에서 Dword 단위로 인터럽트 마스크를 설정 (상승/하강 모두1)  
                    AxdiInterruptEdgeSetDword(ModuleNo, 0, UP_EDGE, 0xFFFFFFFF);  
                    // 인터럽트 허용여부 설정  
                    AxdiInterruptSetModuleEnable(ModuleNo, ENABLE);  
                }  
  
                BOOL fExit = FALSE;  
                while(!fExit)      // 무한 루프  
                {  
                    if(kbhit())    // 아무키나 누르면  
                    {  
                        int ch = getch();  
                        switch(ch)  
                        {  
                            case 27:    // Esc key  
                                fExit = TRUE;  
                        }  
                    }  
                }  
            }  
        }  
    }  
}
```

```
        break;
    }
}
}
else
    printf("AxdInfoIsDIOModule() : ERROR ( NOT STATUS_EXIST ) code 0x%x\n",Code);
}
else
    printf("AxdInfoIsDIOModule() : ERROR ( Return FALSE ) code 0x%x\n",Code);
}
else
printf("AxlOpen() : ERROR code 0x%x\n",Code);

// 라이브러리를 종료 한다.
if(AxlClose())
    printf("\n라이브러리가 종료 되었습니다.\n");
else
    printf("\n라이브러리가 정상적으로 종료되지 않았습니다.\n");
}
```

고급 **DIO**함수

AXL 라이브러리에서 위에서 언급한 기본적인 입출력 뿐만 아니라, 현장에서 빈번히 사용되는 몇가지 고급 DIO기능 함수들을 제공한다.

신호의 변화 여부 확인

AxdIsPulseOn: 지정한 모듈의 특정 입력 offset번호에 해당하는 입력접점에서 들어오는 신호가 off에서 on으로 바뀌었는지를 확인하는 함수이다. AxdIsPulseOn함수를 이용하여 신호의 변화 여부를 확인하기 위해서는 AxdIsPulseOn함수를 최소한 두번 이상 호출하여야 한다. 이 함수는 이전에 호출되었을때의 신호레벨이 Off이면서 현재 호출될 때On 이면 TRUE를 반환하고, 아니면 FALSE를 반환한다.

0번 모듈의 0번offset에 해당하는 입력접점의 신호가 Off에서 On으로 바뀌었는지 확인하고자 한다면 다음과 같이 한다.

```
// 0번 모듈의 0번 offset에 해당하는 입력접점의 신호가 off에서 on으로 바뀌었는지 확인한다.
DWORD uValue;
AxdIsPulseOn(0, 0, &uValue);
if(uValue)
    printf("0번 모듈의 0번offset에 해당하는 입력접점의 신호가 off에서 on으로 바뀌었습니다.");
```

AxdIsPulseOff: 지정한 모듈의 특정 offset번호에 해당하는 입력접점에서 들어오는 신호가 On에서 Off로 바뀌었는지를 확인하는 함수이다. 이 함수는 이전에 호출되었을 때의 신호레벨이 On이면서 현재 호출될 때Off이면 TRUE를 반환하고, 아니면 FALSE를 반환한다. 0번 모듈의 1번 offset에 해당하는 입력접점의 신호가 Off에서 On으로 바뀌었는지 확인하고자 한다면 다음과 같이 한다.

```
// 0번 모듈의1번offset에 해당하는 입력접점의 신호가 on에서 off로 바뀌었는지 확인한다.
DWORD uValue;
AxdIsPulseOff(0, 1, &uValue);
if(uValue)
    printf("0번 모듈의 1번 offset에 해당하는 입력접점의 신호가 on에서 off로 바뀌었습니다.");
```

```
// Ex6_AXD_IsPulseOnOff.cpp : Defines the entry point for the console application.
// 특정 입력포트에서 신호의 변화유무를 확인한다.

#include "stdafx.h"
#include "AXL.h"
#include <conio.h>
#include "stdio.h"

void main(void)
{
    // 라이브러리를 초기화 한다.
    // 7은IRQ를 뜻한다. PCI에서 자동으로 IRQ가 설정된다.
    DWORD Code = Ax1Open(7);
    if(Code == AXT_RT_SUCCESS)
    {
```

```

printf("라이브러리가 초기화 되었습니다.\n");
// DIO 모듈이 있는지 검사.
DWORD dwStatus;
Code = AxdiInfoIsDIOModule(&dwStatus);
if(Code == AXT_RT_SUCCESS)
{
    if(dwStatus == STATUS_EXIST)
    {
        // DIO모듈의 개수를 확인
        long lModuleCount;
        AxdiInfoGetModuleCount(&lModuleCount);

        // DIO모듈의 입출력 신호레벨을 High로 설정한다.
        for(int ModuleNo=0;ModuleNo<lModuleCount;ModuleNo++)
            AxdiSetPortLevelDword(ModuleNo,0,0xffffffff);

        // 무한루프를 돌면서 Output Port의 현재 상태를 반환한다.
        printf("[INFORMATION]*****\n");
        printf("[ESC] : Exit \n");
        printf("*****\n");

        BOOL fExit = FALSE;
        while(!fExit)      // 무한 루프
        {
            if(kbhit()) // 아무키나 누르면
            {
                int ch = getch();
                switch(ch)
                {
                    case 27:   // Esc key
                        fExit = TRUE;
                        break;
                }
            }

            // Input Port의 현재상태를 반환한다.
            DWORD Status_Input;
            DWORD uValue;
            AxdiReadInport(0,&Status_Input);
            printf("\r0번 모듈의0번offset의 현재 상태 : %d",Status_Input);

            // 0번 모듈의 0번 offset 해당하는 입력접점 신호가 off에서 on으로 바뀌었는지 확인
            AxdiIsPulseOn(0,0,&uValue);
            if(uValue)
                printf("\n0번 모듈의0번 offset 해당하는 입력접점 신호가 off에서 on으로 바뀌었습니다.\n");

            // 0번 모듈의 0번 offset 해당하는 입력접점 신호가 on에서 off로 바뀌었는지 확인한다.
            AxdiIsPulseOff(0,0,&uValue);
        }
    }
}

```

```

        if(uValue)
            printf("\n0번 모듈의0번offset에 해당하는 입력접점의 신호가 On에서 Off로 바뀌었습니다.\n");
    }
}
else
    printf("AxdInfoIsDIOModule() : ERROR ( NOT STATUS_EXIST ) code 0x%x\n",Code);
}
else
    printf("AxdInfoIsDIOModule() : ERROR ( Return FALSE ) code 0x%x\n",Code);
}
else
    printf("AxlOpen() : ERROR code 0x%x\n",Code);

// 라이브러리를 종료 한다.
if(AxlClose())
    printf("\n라이브러리가 종료 되었습니다.\n");
else
    printf("\n라이브러리가 정상적으로 종료되지 않았습니다.\n");
}

```

신호의 지속여부 확인

AxdIsOn: 지정한 모듈의 특정 offset번호에 해당하는 입력접점에서 들어오는 신호가 AxdIsOn함수를 n번 만큼 호출할동안 On을 유지하고 있는지를 확인하는 함수이다. 이 함수는 설정한 횟수 만큼 AxdIsOn함수를 호출하는 동안에 지속적으로 해당 입력접점의 신호가 On이면 마지막 n번째부터의 호출에서 TRUE를 반환하고, 아니면 FALSE를 반환한다. N - 1번째 호출 까지의 반환값은 FALSE이다.

0번 모듈의 0번 offset에 해당하는 입력접점의 신호가 10회 호출될 동안에 On을 유지하는지 확인하고자 한다면 다음과 같이 한다.

```

// 0번 모듈의 0번 offset에 해당하는 입력접점의 신호가 10회 호출될동안 On을 유지하는지 확인
DWORD uValue;

AxdIsOn(0,0,10,&uValue, 1);           // 최초 시작
for(int i = 0; i < 9; i++)
{
    // 실제 프로그램에서는 Timer 나 Thread 를 통한 일정 주기를 가진 호출 루틴이 사용..
    AxdIsOn(0,0,10,&uValue, 0);
}
if(uValue)
    printf("0번 모듈의 0번 offset에 해당하는 입력접점의 신호가 On을 유지하였습니다.");

```

AxdIsOff: 지정한 모듈의 특정 offset번호에 해당하는 입력접점에서 들어오는 신호가 AxdIsOff함수를 n번 만큼 호출할동안 Off를 유지하고 있는지를 확인하는 함수이다. 이 함수는 설정한 횟수 만큼 AxdIsOff함수를 호출하는 동안에 지속적으로 해당 입력접점의 신호가 Off이면 마지막 n번째부터의 호출에서 TRUE를 반환하고, 아니면 FALSE를 반환한다. n-1번째 호출 까지의 반환값은 FALSE이다.

0번 모듈의 0번 offset에 해당하는 입력접점의 신호가 10회 호출될 동안에 지속적으로 Off를 유지하는지 확

인하고자 한다면 다음과 같이 한다.

```
// 0번 모듈의 0번 offset에 해당하는 입력접점의 신호가 10회 호출될동안 off를 유지하는지 확인
DWORD uValue;

AxdiIsOff(0,0,10,&uValue, 1); // 최초 시작
for(int i = 0; i < 9; i++)
{
    // 실제 프로그램에서는 Timer 나 Thread 를 통한 일정 주기를 가진 호출 루틴이 사용.
    AxdiIsOff(0,0,10,&uValue, 0);
}
if(uValue)
    printf("0번 모듈의 0번 offset에 해당하는 입력접점의 신호가 off를 유지하였습니다.\n");
```

```
// Ex7_AXD_IsOnOff.cpp : Defines the entry point for the console application.
// 특정 입력접점에서 신호의 지속여부를 확인
```

```
#include "stdafx.h"
#include "AXL.h"
#include <conio.h>
#include "stdio.h"

void main(void)
{
    // 라이브러리를 초기화 한다.
    // 7은 IRQ를 뜻한다. PCI는 자동으로 IRQ가 설정된다.
    DWORD Code = AxlOpen(7);
    if(Code == AXT_RT_SUCCESS)
    {
        printf("라이브러리가 초기화 되었습니다.\n");

        // DIO 모듈이 있는지 검사
        DWORD dwStatus;
        Code = AxdiInfoIsDIOModule(&dwStatus);
        if(Code == AXT_RT_SUCCESS)
        {
            if(dwStatus == STATUS_EXIST)
            {
                // DIO모듈의 개수를 확인
                long lModuleCount;
                AxdiInfoGetModuleCount(&lModuleCount);

                // DIO모듈의 입출력 신호레벨을 High로 설정한다.
                for(int ModuleNo=0;ModuleNo<lModuleCounts;ModuleNo++)
                {
                    if(AxdiInfoGetInputCount(ModuleNo) > 0)
                    {
                        AxdiLevelSetImportDword(ModuleNo, 0, 0xffffffff);
                    }
                }
            }
        }
    }
}
```

```
        }

        if (AxdiInfoGetOutputCount (ModuleNo) > 0)
        {
            AxdoLevelSetOutportDword (ModuleNo, 0, 0xffffffff);
        }
    }

    // 무한루프를 돌면서 Output Port의 현재 상태를 반환한다.
    printf("[INFORMATION]*****\n");
    printf("[ESC] : Exit \n");
    printf("[1] : 0번 모듈의0번offset 입력접점이 100회 호출될동안 On을 유지하는지 확인 \n");
    printf("[2] : 0번 모듈의0번offset 입력접점이 100회 호출될동안 Off를 유지하는지 확인 \n");
    printf("*****\n");

    BOOL fExit = FALSE;
    while(!fExit)      // 무한 루프
    {
        if(kbhit()) // 아무키나 누르면
        {
            int ch = getch();
            DWORD OutPort;
            DWORD uValue;
            int nCount;
            switch(ch)
            {
                case 27:   // Esc key
                    fExit = TRUE;
                    break;

                case '1':
                    // 0번 모듈의0번offset의 입력접점의 신호가 100회 호출될동안 On을 유지하는지 확인
                    AxdiIsOn(0,0,100,&uValue, 1); //최초 시작
                    for(nCount = 0; nCount < 99; nCount++)
                    {
                        AxdiIsOn(0, 0, 100, &uValue, 0);
                    }

                    if(uValue)
                        printf("\n0번 모듈의0번offset의 입력접점의 신호가 On을 유지하였습니다.\n");
                    else
                        printf("\n0번 모듈의0번offset의 입력접점의 신호가 On을 유지하지 못했습니다.\n");
                    break;

                case '2':
                    // 0번 모듈의0번offset의 입력접점의 신호가 100회 호출될 동안 Off를 유지하는지 확인
                    AxdiIsOff(0, 0, 100, &uValue, 1); // 최초 시작
                    for(nCount = 0; nCount < 99; nCount++)
                    {

```

```

        AxdiIsOff(0, 0, 100, &uValue, 0);
    }

    if(uValue)
        printf("\n0번 모듈의0번offset의 입력접점의 신호가 off를 유지하였습니
               다.\n");
    else
        printf("\n0번 모듈의0번offset의 입력접점의 신호가 off를 유지하지 못했습니
               다.\n");
    break;
}
}

// Input Port의 현재상태를 반환한다.
DWORD Status_Input;
DWORD uValue;
AxdiReadInput(0,&Status_Input);
printf("\r0번 모듈의0번offset의 현재 상태 : %d",Status_Input);
}

}

else
    printf("AxdInfoIsDIOModule() : ERROR ( NOT STATUS_EXIST ) code 0x%x\n",Code);
}
else
    printf("AxdInfoIsDIOModule() : ERROR ( Return FALSE ) code 0x%x\n",Code);
}
else
    printf("AxlOpen() : ERROR code 0x%x\n",Code);

// 라이브러리를 종료 한다.
if(AxlClose())
    printf("\n라이브러리가 종료 되었습니다.\n");
else
    printf("\n라이브러리가 정상적으로 종료되지 않았습니다.\n");
}

```

일정 시간동안 신호유지

AxdoOutPulseOn: 지정한 모듈의 특정 offset번호에 해당하는 출력접점에 mSec단위로 일정 시간동안 On상태를 유지하다가 Off시키는 함수이다. offset값은 출력접점들만 '0'부터 시작하는 것이 아니고, 전체 비트에서 사용자가 출력접점의 위치를 확인하여 사용해야 한다. 예를들어 SIO-DB32모듈이라고 하면, 0~15까지는 입력접점이고, 16~31까지가 출력접점이므로 offset값은 16~31까지의 값을 사용하여야 한다는 것이다.

0번 모듈의 0번째 출력접점(offset 16)의 신호를 1초동안 On상태를 유지하다가 Off시키고자 한다면 다음과 같이 한다.

```
// 0번 모듈의 16번 offset에 해당하는 출력접점의 신호를 1초(1000ms)동안 On시킨후 Off시킨다.
AxdoOutPulseOn(0,16,1000);
```

AxdoOutPulseOff: 지정한 모듈의 특정 offset번호에 해당하는 출력접점에 mSec단위로 일정시간동안 Off 상태를 유지하다가 On시키는 함수이다. offset값은 위에서와 마찬가지로 전체 비트에서 사용자가 출력접점의 위치를 확인하여 사용해야 한다.

0번 모듈의 0번째 출력접점(offset 16)의 신호를 1초 후에 On시키고자 한다면 다음과 같이 한다.

```
// 0번 모듈의 16번offset에 해당하는 출력접점의 신호를 1초(1000ms) 후에 On시킨다.
AxdoOutPulseOff(0,16,1000);
```

```
// Ex8_AXD_OutPulseOnOff.cpp : Defines the entry point for the console application.
// 특정 출력 포트에서 신호를 일정시간 On으로 유지 또는 일정 시간후 신호를 On시킨다.
```

```
#include "stdafx.h"
#include "AXL.h"
#include <conio.h>
#include "stdio.h"

void main(void)
{
    // 라이브러리를 초기화 한다.
    // 7은 IRQ를 뜻한다. PCI는 자동으로 IRQ가 설정된다.
    DWORD Code = Ax1Open(7);
    if(Code == AXT_RT_SUCCESS)
    {
        printf("라이브러리가 초기화 되었습니다.\n");

        // DIO 모듈이 있는지 검사
        DWORD dwStatus;
        Code = AxdInfoIsDIOModule(&dwStatus);
        if(Code == AXT_RT_SUCCESS)
        {
            if(dwStatus == STATUS_EXIST)
            {
                // DIO모듈의 개수를 확인
                long lModuleCount;
                AxdInfoGetModuleCount(&lModuleCount);

                // DIO모듈의 입출력 신호레벨을 High로 설정한다.
                for(int ModuleNo=0;ModuleNo<lModuleCounts;ModuleNo++)
                {
                    if(AxdInfoGetInputCount(ModuleNo) > 0)
                    {
                        AxdiLevelSetInportDword(ModuleNo,0,0xffffffff);
                    }
                    if(AxdInfoGetOutputCount(ModuleNo) > 0)
                    {
                        AxdoLevelSetOutportDword(ModuleNo,0,0xffffffff);
                    }
                }
            }
        }
    }
}
```

```

}

// 무한루프를 돌면서 Output Port의 현재 상태를 반환한다.
printf("[INFORMATION]*****\n");
printf("[ESC] : Exit \n");
printf("[1] : 신호를 일정시간후에 OFF 시킨다. \n");
printf("[2] : 신호를 일정시간후에 ON 시킨다. \n");
printf("*****\n");

BOOL fExit = FALSE;
while(!fExit)      // 무한 루프
{
    if(kbhit()) // 아무키나 누르면
    {
        int ch = getch();
        long nDelay = 1000; // [ms단위]

        int lOffset;
        switch(ch)
        {
            case 27: // Esc key
                fExit = TRUE;
                break;

            case '1':
                // 0번 모듈의 출력점점 신호를 순차적으로 off시킨다.
                for(lOffset=0;lOffset<32;lOffset++)
                    AxdoOutPulseOn(0,lOffset,nDelay*(lOffset-16));
                break;

            case '2':
                // 0번 모듈의 출력점점 신호를 순차적으로 off시킨다.
                for(lOffset=0;lOffset<32;lOffset++)
                    AxdoOutPulseOff(0,lOffset,nDelay*(lOffset-16));
                break;
        }
    }

    // Output Port의 현재상태를 반환한다.
    DWORD Status_Output[32];
    for(int OffsetNo=0; OffsetNo < 32; OffsetNo++)
        AxdoReadOutport(OffsetNo,&Status_Output[OffsetNo]);

    printf("\r16[%d],17[%d],18[%d],19[%d],20[%d],21[%d],22[%d]...",
          Status_Output[0], Status_Output[1], Status_Output[2], Status_Output[3],
          Status_Output[4], Status_Output[5], Status_Output[6], Status_Output[7],
          Status_Output[8]);
}
}

```

```

        else
            printf("AxdInfoIsDIOModule() : ERROR ( NOT STATUS_EXIST ) code 0x%x\n",Code);
    }
    else
        printf("AxdInfoIsDIOModule() : ERROR ( Return FALSE ) code 0x%x\n",Code);
}
else
    printf("AxlOpen() : ERROR code 0x%x\n",Code);

// 라이브러리를 종료 한다.
if(AxlClose())
    printf("\n라이브러리가 종료 되었습니다.\n");
else
    printf("\n라이브러리가 정상적으로 종료되지 않았습니다.\n");
}

```

신호의 토글

AxdoToggleStart: 지정한 모듈의 특정 출력 offset번호에 해당하는 출력 접점에 설정한 횟수 및 간격(mSec)으로 신호를 토글한 후 원래의 출력 상태를 유지하는 함수이다.

0번 모듈의 0번째 출력접점(offset 0)의 신호를 지정된 On/Off 시간(mSec) 간격으로 10회 토글한 후 원래의 상태로 돌아가고자 한다면 다음과 같이 한다. 무한대 반복을 하고자 하면 반복 횟수에 -1을 기입한다.

```

// 0번 모듈의 16번offset에 해당하는 출력접점의 신호를 초기 시작을 On으로 100 + 200mSec 간격으로 10회 토글한
// 다. (On time 100mSec , Off Time 200 mSec)
AxdoToggleStart(0, 0, 1, 100, 200, 10);

```

AxdoToggleStop: 지정한 모듈의 특정 출력 offset번호에 해당하는 출력접점에 토글중인 신호를 정지시키고 특정 신호로 고정시키는 함수이다. offset값은 위에서와 같다.

토글중인 0번 모듈의 0번째 출력접점(offset 0)의 신호를 정지시키고 신호상태를 On으로 하고자 한다면 다음과 같이 한다.

```

// 토글중인0번 모듈의 0번 출력 offset에 해당하는 출력접점의 신호를 정지시키고, On으로 유지한다.
AxdoToggleStop(0, 0, 1);

```

```

// Ex9_AXD_ToggleStartStop.cpp : Defines the entry point for the console application.
// 특정 출력포트에서 신호를 토글시킨다.

#include "stdafx.h"
#include "AXL.h"
#include <conio.h>
#include "stdio.h"

void main(void)
{
    // 라이브러리를 초기화 한다.
    // 7은IRQ를 뜻한다. PCI에서 자동으로 IRQ가 설정된다.

```

```

DWORD Code = Ax1Open(7);
if(Code == AXT_RT_SUCCESS)
{
    printf("라이브러리가 초기화 되었습니다.\n");

    // DIO 모듈이 있는지 검사
    DWORD dwStatus;
    Code = AxdiInfoIsDIOModule(&dwStatus);
    if(Code == AXT_RT_SUCCESS)
    {
        if(dwStatus == STATUS_EXIST)
        {
            // DIO모듈의 개수를 확인
            long lModuleCount;
            AxdiInfoGetModuleCount (&lModuleCount);

            // DIO모듈의 입출력 신호레벨을 High로 설정한다.
            for(int ModuleNo=0;ModuleNo<lModuleCounts;ModuleNo++)
            {
                if(AxdiInfoGetInputCount (ModuleNo) > 0)
                {
                    AxdiLevelSetImportDword (ModuleNo,0,0xffffffff);
                }
                if(AxdiInfoGetOutputCount (ModuleNo) > 0)
                {
                    AxdoLevelSetOutportDword (ModuleNo,0,0xffffffff);
                }
            }

            // 무한루프를 돌면서 Output Port의 현재 상태를 반환한다.
            printf("[INFORMATION]*****\n");
            printf("[ESC] : Exit \n");
            printf("[1] : 1초간격으로 신호를 10회동안 토글시킨다. \n");
            printf("[2] : 토글중인 신호를 정지하고 off로 고정한다. \n");
            printf("*****\n");

            BOOL fExit = FALSE;
            while(!fExit) // 무한 루프
            {
                if(kbhit()) // 아무키나 누르면
                {
                    int ch = getch();
                    long nDelay = 1000; // [ms단위]
                    int lOffset;
                    switch(ch)
                    {
                        case 27: // Esc key
                            fExit = TRUE;
                            break;
                    }
                }
            }
        }
    }
}

```

```
        case '1':
            // 0번 모듈의 출력점점 신호를 1초 간격으로 10회 토글시킨다.
            for(lOffset=0;lOffset<32;lOffset++)
                AxdoToggleStart(0,lOffset,nDelay,10);
            break;

        case '2':
            // 토글중인 신호를 정지시키고 off로 유지한다.
            for(lOffset=0;lOffset<32;lOffset++)
                AxdoToggleStop(0,lOffset,0);
            break;
        }
    }

    // Output Port의 현재상태를 반환한다.
    DWORD Status_Output[32];
    for(int OffsetNo=0;OffsetNo<32;OffsetNo++)
        AxdoReadOutport(OffsetNo,&Status_Output[OffsetNo]);

    printf("\r16[%d],17[%d],18[%d],19[%d],20[%d],21[%d],22[%d]...",
        Status_Output[0], Status_Output[1], Status_Output[2], Status_Output[3],
        Status_Output[4], Status_Output[5], Status_Output[6], Status_Output[7],
        Status_Output[8]);
    }
}
else
    printf("AxdInfoIsDIOModule() : ERROR ( NOT STATUS_EXIST ) code 0x%x\n",Code);
}
else
    printf("AxdInfoIsDIOModule() : ERROR ( Return FALSE ) code 0x%x\n",Code);
}
else
    printf("Ax1Open() : ERROR code 0x%x\n",Code);

// 라이브러리를 종료한다.
if(Ax1Close())
    printf("\n라이브러리가 종료 되었습니다.\n");
else
    printf("\n라이브러리가 정상적으로 종료되지 않았습니다.\n");
}
```

DIO Command 매뉴얼 정보

본 매뉴얼은 SIO-DI32/SIO-DO32P/SIO-DB32P/SIO-DO32T/SIO-DB32T모듈과 PCI-DB64R, PCI-DI64R, PCI-DO64R 보드 제품 및 NETWORK으로 구성된 RTEX, MECHATROLINK II의 제품을 Windows 98, Windows NT, Windows 2000 또는 Windows XP의 OS환경에서 Microsoft VC++6.0, Visual Basic, Borland C-Builder, Delphi등의 언어에서 구동하기 위해 필요한 매뉴얼이며, 포함된 라이브러리 함수를 기능별로 분류하여 설명하였다.

헤더 파일

C++

AXD.h

Visual Basic

AXD.bas

Delphi

AXD.pas

함수 용어

본 매뉴얼의 함수 이름

본 매뉴얼에서 사용된 함수 이름들은 접두어(Prefix)에 의해 동작을 구분할 수 있도록 되어있다.

라이브러리 함수 Prefix

Prefix/Suffix	설명
AxdInfo	보드 및 모듈 정보 확인 함수
Axdi	Digital 입력 관련 대분류
Apdo	Digital 출력 관련 대분류
AxdInterrupt	입력 인터럽트 설정 확인
AxdInterruptEdge	입력 인터럽트 상승 / 하강 에지시 인터럽트 발생 설정 확인
AxdLevel	입력 신호 레벨 설정 확인
AxdRead	입력 신호 읽기
ApdoRead	출력 신호 읽기
ApdoWrite	출력 신호 쓰기
ApdoLevel	출력 신호 레벨 설정 확인
Get	레지스터 값 확인
Set	레지스터 값 설정
Inport	입력 접점과 관련된 함수
Outport	출력 접점과 관련된 함수
Bit	1개의 접점 단위
Byte	8개의 접점 단위
Word	16개의 접점 단위
Dword	32개의 접점 단위

본 매뉴얼의 인자 이름

본 매뉴얼에서 사용된 함수들의 공통적인 인자들은 다음과 같은 의미를 가진다.

long lBoardNo: 초기화 된 베이스 보드(Base Board)의 첫 번째 보드부터 오름차순으로 자동 정렬된다. 보드 번호는 '0'부터 시작한다.

long lModuleNo: DIO 모듈 초기화 시에 첫 번째 보드의 모듈부터 오름차순으로 자동 정렬된다. 모듈 번호는 '0'부터 시작한다.

DWORD uLevel: uLevel은 ReadPort 및 WriteOutport 관련 함수들을 사용 시 Level을 설정하고 확인한다.

DWORD uOffset: uOffset은 사용 모듈과 사용 함수에 따라 유효 범위가 달라지는데 아래의 Table을 참조하여 설정한다.

Data Type	Offset	Value or Return	Related
All	0 ~ n-1	0(Off), 1(On)	
Bit	0 ~ 31	0(Off), 1(On)	Bit 배열 함수 군
Byte	0, 1, 2, 3	00h ~ FFh	Byte 배열 함수 군
Word	0, 1	0000h ~ FFFFh	Word 배열 함수 군
Dword	0	00000000h ~ FFFFFFFFh	Dword 배열 함수 군

만약, 시스템에 SIO-DB32 / SIO-DI32 / SIO-DO32 모듈이 하나씩 장착되어 있고 할당된 Module No가 각각 0,1,2로 순차적이라면 각 모듈별 할당 Offset의 범위와 각 함수의 연관성은 아래 표와 같다.

입력 Offset

함수 모듈	SIO-DB32 (In32/Out32)	SIO-DI32 (In32)	Data Size	비고
AxdiLevelSetInport AxdiLevelGetInport AxdiReadInport	0 ~ 15	16 ~ 47	1	전체 모듈에서 Offset 적용됨
AxdiLevelSetInportBit AxdiLevelGetInportBit AxdiReadInportBit	0 ~ 15	0 ~ 31	1	각각의 모듈 Offset 적용
AxdiLevelSetInportByte AxdiLevelGetInportByte AxdiReadInportByte	0 ~ 1	0 ~ 3	8	각각의 모듈 Offset 적용
AxdiLevelSetInportWord AxdiLevelGetInportWord AxdiReadInportWord	0	0 ~ 1	16	각각의 모듈 Offset 적용
AxdiLevelSetInportDword AxdiLevelGetInportDword AxdiReadInportDword	0	0	32	각각의 모듈 Offset 적용

출력 Offset

함수 모듈	SIO-DB32 (In32/Out32)	SIO-DI32 (In32)	Data Size	비고
AxdoLevelSetOutport AxdoLevelGetOutport AxdoReadOutport AxdoWriteOutport	0 ~ 15	16~47	1	전체 모듈에서 Offset 적용됨
AxdoLevelSetOutportBit AxdoLevelGetOutportBit AxdoReadOutportBit AxdoWriteOutportBit	0 ~ 15	0~31	1	각각의 모듈 Offset 적용
AxdoLevelSetOutportByte AxdoLevelGetOutportByte AxdoReadOutportByte AxdoWriteOutportByte	0 ~ 1	0~3	8	각각의 모듈 Offset 적용
AxdoLevelSetOutportWord AxdoLevelGetOutportWord AxdoReadOutportWord AxdoWriteOutportWord	0	0 ~ 1	16	각각의 모듈 Offset 적용
AxdoLevelSetOutportDword AxdoLevelGetOutportDword AxdoReadOutportDword AxdoWriteOutportDword	0	0	32	각각의 모듈 Offset 적용

Interrupt Rising / Falling Edge Register

Interrupt Rising(Up) Edge는 사용자가 원하는 해당 입력 비트의 Rising Edge에서 인터럽트를 검출하기 위해 설정하는 레지스터이다.

해당 비트를 ‘1’로 셋팅하면 Rising Edge시에 인터럽트가 발생한다.

Interrupt Falling(Down) Edge는 사용자가 원하는 해당 입력 비트의 Falling Edge에서 인터럽트를 검출하기 위해 설정하는 레지스터이다.

해당 비트를 ‘1’로 셋팅하면 Falling Edge시에 인터럽트가 발생한다.

단, 인터럽트가 Enable일 경우에 인터럽트가 발생 한다.

DIO Command Quick List

보드 및 모듈 정보

Function	Description
AxdiInfoIsDIOModule	DIO 모듈이 있는지 확인한다.
AxdiInfoGetModuleNo	DIO 모듈의 번호를 확인한다.
AxdiInfoGetModuleCount	DIO 입출력 모듈의 개수를 확인한다.
AxdiInfoGetInputCount	지정한 모듈의 입력 점점 개수를 확인한다.
AxdiInfoGetOutputCount	지정한 모듈의 출력 점점 개수를 확인한다.
AxdiInfoGetModule	지정한 모듈 번호로 베이스 보드 번호, 모듈 위치, 모듈 ID를 확인한다.

인터럽트

인터럽트 설정 확인

Function	Description
AxdiInterruptSetModule	지정한 모듈의 인터럽트 메시지를 받아오기 위하여 윈도우 메시지, 콜백 함수 또는 이벤트 방식을 사용한다.
AxdiInterruptSetModuleEnable	지정한 모듈의 인터럽트 사용 유무를 설정한다.
AxdiInterruptGetModuleEnable	지정한 모듈의 인터럽트 사용 유무를 확인한다.
AxdiInterruptRead	인터럽트 발생 위치를 확인한다.

인터럽트 상승 / 하강 에지 설정 확인

Function	Description
AxdiInterruptEdgeSetBit	지정한 입력 점점 모듈, Interrupt Rising / Falling Edge Register의 Offset 위치에서 bit 단위로 상승 또는 하강 에지 값을 설정한다.
AxdiInterruptEdgeSetByte	지정한 입력 점점 모듈, Interrupt Rising / Falling Edge Register의 Offset 위치에서 byte 단위로 상승 또는 하강 에지 값을 설정한다.
AxdiInterruptEdgeSetWord	지정한 입력 점점 모듈, Interrupt Rising Rising / Falling Edge Register의 Offset 위치에서 word 단위로 상승 또는 하강 에지 값을 설정한다.
AxdiInterruptEdgeSetDword	지정한 입력 점점 모듈, Interrupt Rising Rising / Falling Edge Register의 Offset 위치에서 double word 단위로 상승 또는 하강 에지 값을 설정한다.
AxdiInterruptEdgeGetBit	지정한 입력 점점 모듈, Interrupt Rising / Falling Edge Register의 Offset 위치에서 bit 단위로 상승 또는 하강 에지 값을 확인한다.
AxdiInterruptEdgeGetByte	지정한 입력 점점 모듈, Interrupt Rising / Falling Edge Register의 Offset 위치에서 Byte 단위로 상승 또는 하강 에지 값을 확인한다.
AxdiInterruptEdgeGetWord	지정한 입력 점점 모듈, Interrupt Rising / Falling Edge Register의 Offset 위치에서 word 단위로 상승 또는 하강 에지 값을 확인한다.

AxdiInterruptEdgeGetDword	지정한 입력 접점 모듈, Interrupt Rising / Falling Edge Register의 Offset 위치에서 double word 단위로 상승 또는 하강 에지 값을 확인한다.
AxdiInterruptEdgeSet	전체 입력 접점 모듈, Interrupt Rising / Falling Edge Register의 Offset 위치에서 bit 단위로 상승 또는 하강 에지 값을 설정한다.
AxdiInterruptEdgeGet	전체 입력 접점 모듈, Interrupt Rising / Falling Edge Register의 Offset 위치에서 bit 단위로 상승 또는 하강 에지 값을 확인한다.

입력 레벨 설정 확인

Function	Description
AxdiLevelSetInportBit	지정한 입력 접점 모듈의 Offset 위치에서 bit 단위로 레벨을 설정한다.
AxdiLevelSetInportByte	지정한 입력 접점 모듈의 Offset 위치에서 byte 단위로 레벨을 설정한다.
AxdiLevelSetInportWord	지정한 입력 접점 모듈의 Offset 위치에서 word 단위로 레벨을 설정한다.
AxdiLevelSetInportDword	지정한 입력 접점 모듈의 Offset 위치에서 double word 단위로 레벨을 설정한다.
AxdiLevelGetInportBit	지정한 입력 접점 모듈의 Offset 위치에서 bit 단위로 레벨을 확인한다.
AxdiLevelGetInportByte	지정한 입력 접점 모듈의 Offset 위치에서 byte 단위로 레벨을 확인한다.
AxdiLevelGetInportWord	지정한 입력 접점 모듈의 Offset 위치에서 word 단위로 레벨을 확인한다.
AxdiLevelGetInportDword	지정한 입력 접점 모듈의 Offset 위치에서 double word 단위로 레벨을 확인한다.
AxdiLevelSetInport	전체 입력 접점 모듈의 Offset 위치에서 bit 단위로 레벨을 설정한다.
AxdiLevelGetInport	전체 입력 접점 모듈의 Offset 위치에서 bit 단위로 레벨을 확인한다.

출력 레벨 설정 확인

Function	Description
AxdoLevelSetOutportBit	지정한 출력 접점 모듈의 Offset 위치에서 bit 단위로 레벨을 설정한다.
AxdoLevelSetOutportByte	지정한 출력 접점 모듈의 Offset 위치에서 byte 단위로 레벨을 설정한다.
AxdoLevelSetOutportWord	지정한 출력 접점 모듈의 Offset 위치에서 word 단위로 레벨을 설정한다.
AxdoLevelSetOutportDword	지정한 출력 접점 모듈의 Offset 위치에서 double word 단위로 레벨을 설정한다.
AxdoLevelGetOutportBit	지정한 출력 접점 모듈의 Offset 위치에서 bit 단위로 레벨을 확인한다.
AxdoLevelGetOutportByte	지정한 출력 접점 모듈의 Offset 위치에서 byte 단위로 레벨을 확인한다.
AxdoLevelGetOutportWord	지정한 출력 접점 모듈의 Offset 위치에서 word 단위로 레벨을 확인한다.
AxdoLevelGetOutportDword	지정한 출력 접점 모듈의 Offset 위치에서 double word 단위로 레벨을 확인한다.
AxdoLevelSetOutport	전체 출력 접점 모듈의 Offset 위치에서 bit 단위로 레벨을 설정한다.
AxdoLevelGetOutport	전체 출력 접점 모듈의 Offset 위치에서 bit 단위로 레벨을 확인한다.

입력 포트 읽기

Function	Description
AxdiReadInportBit	지정한 입력 접점 모듈의 Offset 위치에서 bit 단위로 데이터를 읽는다.
AxdiReadInportByte	지정한 입력 접점 모듈의 Offset 위치에서 byte 단위로 데이터를 읽는다.
AxdiReadInportWord	지정한 입력 접점 모듈의 Offset 위치에서 word 단위로 데이터를 읽는다.
AxdiReadInportDword	지정한 입력 접점 모듈의 Offset 위치에서 double word 단위로 데이터를 읽는다.
AxdiReadInport	전체 입력 접점 모듈의 Offset 위치에서 bit 단위로 데이터를 읽는다.

출력 포트 읽기 쓰기

Function	Description
AxdoWriteOutportBit	지정한 출력 접점 모듈의 Offset 위치에서 bit 단위로 데이터를 출력한다.
AxdoWriteOutportByte	지정한 출력 접점 모듈의 Offset 위치에서 byte 단위로 데이터를 출력한다.
AxdoWriteOutportWord	지정한 출력 접점 모듈의 Offset 위치에서 word 단위로 데이터를 출력한다.
AxdoWriteOutportDword	지정한 출력 접점 모듈의 Offset 위치에서 double word 단위로 데이터를 출력한다.
AxdoReadOutportBit	지정한 출력 접점 모듈의 Offset 위치에서 bit 단위로 데이터를 읽는다.
AxdoReadOutportByte	지정한 출력 접점 모듈의 Offset 위치에서 byte 단위로 데이터를 읽는다.
AxdoReadOutportWord	지정한 출력 접점 모듈의 Offset 위치에서 word 단위로 데이터를 읽는다.
AxdoReadOutportDword	지정한 출력 접점 모듈의 Offset 위치에서 double word 단위로 데이터를 읽는다.
AxdoWriteOutport	전체 출력 접점 모듈의 Offset 위치에서 bit 단위로 데이터를 출력한다.
AxdoReadOutport	전체 출력 접점 모듈의 Offset 위치에서 bit 단위로 데이터를 읽는다.

고급함수

Function	Description
AxdisPulseOn	지정한 입력 접점 모듈의 Offset 위치에서 신호가 Off에서 On으로 바뀌었는지 확인한다.
AxdisPulseOff	지정한 입력 접점 모듈의 Offset 위치에서 신호가 On에서 Off으로 바뀌었는지 확인한다.
AxdisOn	지정한 입력 접점 모듈의 Offset 위치에서 신호가 count 만큼 호출될 동안 On 상태로 유지하는지 확인한다.
AxdisOff	지정한 입력 접점 모듈의 Offset 위치에서 신호가 count 만큼 호출될 동안 Off 상태로 유지하는지 확인한다.
AxdoOutPulseOn	지정한 출력 접점 모듈의 Offset 위치에서 설정한 mSec동안 On을 유지하다가 Off 시킨다.
AxdoOutPulseOff	지정한 출력 접점 모듈의 Offset 위치에서 설정한 mSec동안 Off를 유지하다가 On 시킨다.
AxdoToggleStart	지정한 출력 접점 모듈의 Offset 위치에서 설정한 횟수, 설정한 간격으로 토글한

	후 원래의 출력상태를 유지한다.
<u>AxdoToggleStop</u>	지정한 출력 접점 모듈의 Offset 위치에서 토글중인 출력을 설정한 신호 상태로 정지 시킨다.

Define문

AXD Digital Input/Output Library에서 사용하고 있는 Define문 List는 다음과 같다

헤더파일 : #include "AXHS.h"

FALSE & TRUE

Definition	Value	Explanation
FALSE	00h	거짓
TRUE	01h	참

AXT_USE

Definition	Value	Explanation
DISABLE	00h	사용안함
ENABLE	01h	사용함

AXT_EXISTENCE

Definition	Value	Explanation
STATUS_NOTEXIST	00h	모듈이 존재하지않음
STATUS_EXIST	01h	모듈이 존재함

AXT_DIO_EDGE

Definition	Value	Explanation
DOWN_EDGE	00h	신호 하강에지
UP_EDGE	01h	신호 상승에지

AXT_DIO_STATE

Definition	Value	Explanation
OFF_STATE	00h	신호 Not active상태
ON_STATE	01h	신호 Active상태

DIO Command Function List

보드 및 모듈 정보

Function	Description
<u>AxdInfoIsDIOModule</u>	DIO 모듈이 있는지 확인한다.
<u>AxdInfoGetModuleNo</u>	DIO 모듈 Number를 확인한다
<u>AxdInfoGetModuleCount</u>	DIO 입출력 모듈의 개수를 확인한다.
<u>AxdInfoGetInputCount</u>	지정한 모듈의 입력 접점 개수를 확인한다.
<u>AxdInfoGetOutputCount</u>	지정한 모듈의 출력 접점 개수를 확인한다.
<u>AxdInfoGetModule</u>	지정한 모듈 번호로 베이스 보드 번호, 모듈 위치, 모듈 ID를 확인한다.
AxdInfoGetModuleStatus	지정한 모듈이 제어가 가능한 상태인지 반환한다.

AxdInfoIsDIOModule

Purpose

DIO 모듈이 있는지 확인한다.

Format

C++

```
DWORD AxdInfoIsDIOModule(DWORD *upStatus);
```

Visual Basic

```
Function AxdInfoIsDIOModule(ByRef upStatus As Long) As Long
```

Delphi

```
function AxdInfoIsDIOModule(upStatus : PDWord) : DWord; stdcall;
```

Parameters

[in/out] Name	[Init Value] Explanation
[<<]upStatus	DIO모듈의 존재여부: AXT_EXISTENCE - [00h] DIO모듈이 없음 - [01h] DIO모듈이 존재함

Return Values

[0000] AXT_RT_SUCCESS : AXL 라이브러리 초기화 성공

[1053] AXT_RT_NOT_OPEN : 라이브러리 초기화 실패

* See error code Table for more information on status error codes

Description

PC에 장착되어 있는 모든 DIO 모듈들을 초기화 한다.

Example

C++

```
#include "AXL.h"           // 라이브러리 추가
#include "AXHS.h"
#include "AXD.h"
DWORD dwStatus;
// DIO 모듈이 있는지 확인한다.
AxdInfoIsDIOModule(&dwStatus);
if(dwStatus == STATUS_EXIST)
    AfxMessageBox("SIO-Dx32 모듈들이 존재합니다.");
else
    AfxMessageBox("SIO-Dx32 모듈들이 존재하지 않습니다.");
```

Visual Basic

' 라이브러리 사용을 위해 AXD.bas와 AXL.bas, AXHS.bas 모듈을 추가 한다.

Dim lStatus As Long

' DIO 모듈이 있는지 확인한다.

```
AxdInfoIsDIOModule lStatus
If lStatus = STATUS_EXIST Then
    MsgBox "SIO-Dx32모듈들이 존재합니다.", vbOKCancel
Else
    MsgBox "SIO-Dx32 모듈들이 존재하지 않습니다.", vbOKCancel
End If
```

Delphi

```
Uses
AXL, AXHS, AXD { 라이브러리 추가 }
var
    dwStatus : DWord;
{ DIO 모듈이 있는지 확인한다. }
begin
    AxdInfoIsDIOModule(@dwStatus);
    if(dwStatus = STATUS_EXIST) then
        Application.MessageBox ('SIO-Dx32 모듈들이 존재합니다.', 'Ajinextek', MB_OK)
    else
        Application.MessageBox ('SIO-Dx32 모듈들이 존재하지 않습니다.', 'Ajinextek', MB_OK);
end;
```

See Also

[AxdInfoGetModuleNo](#), [AxdInfoGetModuleCount](#), [AxdInfoGetInputCount](#), [AxdInfoGetOutputCount](#),
[AxdInfoGetModule](#)

AxdInfoGetModuleNo

Purpose

DIO 모듈 Number 를 확인한다.

Format

C++

```
DWORD AxdInfoGetModuleNo(long lBoardNo, long lModulePos, long
*lpModuleNo);
```

Visual Basic

```
Function AxdInfoGetModuleNo(ByVal lBoardNo As Long, ByVal lModulePos
As Long, ByRef lpModuleNo As Long) As Long
```

Delphi

```
function AxdInfoGetModuleNo(lBoardNo: LongInt; lModulePos: LongInt;
lpModuleNo: PLongInt) : DWord; stdcall;
```

Parameters

[in/out] Name	[Init Value] Explanation
[>>]lBoardNo	보드 번호
[>>]lModulePos	모듈 위치
[<<]lpModuleNo	DIO 모듈 Number

Return Values

- [0000] AXT_RT_SUCCESS : AXL 라이브러리 초기화 성공
 - [1101] AXT_RT_INVALID_BOARD_NO : 라이브러리 초기화 실패
 - [1102] AXT_RT_INVALID_MODULE_POS : 유효하지 않는 모듈 번호
- * See error code Table for more information on status error codes

Description

초기화 되어 있는 DIO 모듈 Number 를 확인한다.

Example

C++

```
// DIO 모듈 Number를 확인한다.
Long lpModuleNo;
CString strData;

AxdInfoGetModuleNo (0, 0, &lpModuleNo);
strData.Format("0번 보드의 0번 Pos에 있는 DIO 모듈 Number는 %d개 입니다.", lpModuleNo);
AfxMessageBox(strData);
```

Visual Basic

```
' DIO 모듈 Number를 확인한다.
Dim lpModuleNo As Long
Dim strData As String

AxdInfoGetModuleNo 0 0 lCount
strData = "0번 보드의 0번 Pos에 있는 DIO 모듈 Number는 " + CStr(lpModuleNo) + "개 입니다."

MsgBox strData
```

Delphi

```
{ DIO 모듈 Number를 확인한다. }
var
    lpModuleNo: LongInt;
    strData : String;

begin
    AxdInfoGetModuleNo (0, 0, @ lpModuleNo);
    strData := '0번 보드의 0번 Pos에 있는 DIO 모듈 Number는 ' + IntToStr(lpModuleNo) + '개 입니다.';

    Application.MessageBox (PCHAR(strData), 'Ajinextek', MB_OK);
end;
```

See Also

[AxdInfoIsDIOModule](#), [AxdInfoGetModuleCount](#), [AxdInfoGetInputCount](#), [AxdInfoGetOutputCount](#),
[AxdInfoGetModule](#)

AxdInfoGetModuleCount

Purpose

DIO 모듈의 개수를 확인한다.

Format

C++

```
DWORD AxdInfoGetModuleCount(long *lpModuleCount);
```

Visual Basic

```
Function AxdInfoGetModuleCount(ByRef lpModuleCount As Long) As Long
```

Delphi

```
function AxdInfoGetModuleCount(lpModuleCount : PLongInt) : DWord;
  stdcall;
```

Parameters

[in/out] Name	[Init Value] Explanation
[<<]lpModuleCount	DIO 모듈 개수

Return Values

[0000] AXT_RT_SUCCESS : AXL 라이브러리 초기화 성공

[1053] AXT_RT_NOT_OPEN : 라이브러리 초기화 실패

[3051] AXT_RT_DIO_NOT_MODULE : DIO 모듈 없음

* See error code Table for more information on status error codes

Description

초기화 되어 있는 DIO 모듈의 개수를 확인한다.

Example

C++

```
// DIO 모듈의 개수를 확인한다.
long lCount;
CString strData;

AxdInfoGetModuleCount(&lCount);
strData.Format("DIO 모듈의 개수는 %d개 입니다.", lCount);

AfxMessageBox(strData);
```

Visual Basic

\ DIO 모듈의 개수를 확인한다.

```
Dim lCount As Long
Dim strData As String

AxdInfoGetModuleCount lCount
```

```
strData = "DIO 모듈의 개수는 " + CStr(lCount) + "개 입니다."  
MsgBox strData
```

Delphi

```
{ DIO 모듈의 개수를 확인한다. }  
var  
  lCount : LongInt;  
  strData : String;  
  
begin  
  AxdInfoGetModuleCount (@lCount);  
  strData := 'DIO 모듈의 개수는 ' + IntToStr(lCount) + '개 입니다.';  
  
  Application.MessageBox (PCHAR(strData), 'Ajinextek', MB_OK);  
end;
```

See Also

[AxdInfoIsDIOModule](#), [AxdInfoGetModuleNo](#), [AxdInfoGetInputCount](#), [AxdInfoGetOutputCount](#),
[AxdInfoGetModule](#)

AxdInfoGetInputCount

Purpose

지정한 모듈의 입력 접점 개수를 확인한다.

Format

C++

```
DWORD AxdInfoGetInputCount(long lModuleNo, long *lpCount);
```

Visual Basic

```
Function AxdInfoGetInputCount(ByVal lModuleNo As Long, ByRef lpCount As Long) As Long
```

Delphi

```
function AxdInfoGetInputCount(lModuleNo : LongInt; lpCount : PLongInt) : DWord; stdcall;
```

Parameters

[in/out] Name	[Init Value] Explanation
[>>]lModuleNo	모듈 번호 (0 ~ N-1)
[<<]lpCount	입력 접점 개수

Return Values

[0000] AXT_RT_SUCCESS : AXL 라이브러리 초기화 성공

[1053] AXT_RT_NOT_OPEN : 라이브러리 초기화 실패

* See error code Table for more information on status error codes

Description

사용자가 지정한 모듈의 입력 접점 수를 확인한다.

DIO 모듈 입출력 접점 수

Module	Input	Output
SIO-DI32	32	0
SIO-DO32P	0	32
SIO-DB32P	16	16
SIO-DO32T	0	32
SIO-DB32T	16	16

Example

C++

```
// 0번째 모듈의 입력 접점 개수를 확인한다.
long lCount;
CString strData;

AxdInfoGetInputCount(0, &lCount);
strData.Format("0번 모듈의 입력 접점 개수는 %d개 입니다.", lCount);

AfxMessageBox(strData);
```

Visual Basic

```
' 0번째 모듈의 입력 접점 개수를 확인한다.
Dim lCount As Long
Dim strData As String

AxdInfoGetInputCount 0, lCount
strData = "0번 모듈의 입력 접점 개수는 " + CStr(lCount) + "개 입니다."

MsgBox strData
```

Delphi

```
{ 0번째 모듈의 입력 접점 개수를 확인한다. }
var
  lCount : LongInt;
  strData : String;

begin
  AxdInfoGetInputCount (@lCount);
  strData := '0번 모듈의 입력 접점 개수는 ' + IntToStr(lCount) + '개 입니다.';

  Application.MessageBox (PCHAR(strData), 'Ajinextek', MB_OK);
end;
```

See Also

[AxdInfoIsDIOModule](#), [AxdInfoGetModuleNo](#), [AxdInfoGetModuleCount](#), [AxdInfoGetOutputCount](#),
[AxdInfoGetModule](#)

AxdInfoGetOutputCount

Purpose

지정한 모듈의 출력 접점 수를 확인한다.

Format

C++

```
DWORD AxdInfoGetOutputCount(long lModuleNo, long *lpCount);
```

Visual Basic

```
Function AxdInfoGetOutputCount (ByVal lModuleNo As Long, ByRef lpCount As Long) As Long
```

Delphi

```
function AxdInfoGetOutputCount(lModuleNo : LongInt; lpCount : PLongInt) : DWord; stdcall;
```

Parameters

[in/out] Name	[Init Value] Explanation
[>>]lModuleNo	모듈 번호 (0 ~ N-1)
[<<]lpCount	출력 접점 개수

Return Values

[0000] AXT_RT_SUCCESS : AXL 라이브러리 초기화 성공

[3051] AXT_RT_DIO_NOT_MODULE : DIO 모듈 없음

[3101] AXT_RT_DIO_INVALID_MODULE_NO : 유효하지않는 DIO 모듈 번호

* See error code Table for more information on status error codes

Description

사용자가 지정한 모듈의 출력 접점 개수를 확인한다.

DIO 모듈 입출력 접점 수

Module	Input	Output
SIO-DI32	32	0
SIO-DO32P	0	32
SIO-DB32P	16	16
SIO-DO32T	0	32
SIO-DB32T	16	16

Example

C++

```
// 0번째 모듈의 출력 접점 개수를 확인한다.
long lCount;
CString strData;

AxdInfoGetOutputCount(0, &lCount);
strData.Format("0번 모듈의 출력 접점 개수는 %d개 입니다.", lCount);

AfxMessageBox(strData);
```

Visual Basic

```
' 0번째 모듈의 출력 접점 개수를 확인한다.
Dim lCount As Long
Dim strData As String

AxdInfoGetOutputCount 0, lCount
strData = "0번 모듈의 출력 접점 개수는 " + CStr(lCount) + "개 입니다."

MsgBox strData
```

Delphi

```
{ 0번째 모듈의 출력 접점 개수를 확인한다. }
var
  lCount : LongInt;
  strData : String;

begin
  AxdInfoGetOutputCount (@lCount);
  strData := '0번 모듈의 출력 접점 개수는 ' + IntToStr(lCount) + '개 입니다.';

  Application.MessageBox (PCHAR(strData), 'Ajinextek', MB_OK);
end;
```

See Also

[AxdInfoIsDIOModule](#), [AxdInfoGetModuleNo](#), [AxdInfoGetModuleCount](#), [AxdInfoGetInputCount](#),
[AxdInfoGetModule](#)

AxdInfoGetModule

Purpose

지정한 모듈 번호로 베이스 보드 번호, 모듈 위치, 모듈 ID를 확인한다.

Format

C++

```
DWORD AxdInfoGetModule(long lModuleNo, long *lpBoardNo, long
    *lpModulePos, DWORD *upModuleID);
```

Visual Basic

```
Function AxdInfoGetModule(ByIdVal lModuleNo As Long, ByRef lpBoardNo As
    Long, ByRef lpModulePos As Long, ByRef upModuleID As Long) As Long
```

Delphi

```
function AxdInfoGetModule(lModuleNo : LongInt; lpBoardNo : PLongInt;
    lpModulePos : PLongInt; upModuleID : PDWord) : DWord; stdcall;
```

Parameters

[in/out] Name	[Init Value] Explanation
[>>]lModuleNo	모듈 번호 (0 ~ N-1)
[<<]lpBoardNo	베이스 보드 번호 (0 ~ N-1)
[<<]lpModulePos	모듈 위치 (0 ~ 3)
[<<]upModuleID	모듈 ID

Return Values

[0000] AXT_RT_SUCCESS : AXL 라이브러리 초기화 성공

[3051] AXT_RT_DIO_NOT_MODULE : DIO 모듈 없음

[3101] AXT_RT_DIO_INVALID_MODULE_NO : 유효하지않는 DIO 모듈 번호

* See error code Table for more information on status error codes

Description

사용자가 지정한 모듈 번호로 베이스 보드 번호, 모듈 위치, 모듈 ID를 확인한다.

DIO 모듈의 ID는 다음과 같다.

모듈 ID	HEX 값(Decimal)	모듈
AXT_SIO_DI32	0x97 (151)	Digital IN 32점
AXT_SIO_DO32P	0x98 (152)	Digital OUT 32점
AXT_SIO_DB32P	0x99 (153)	Digital IN 16점 / OUT 16점
AXT_SIO_DO32T	0x9E (158)	Digital OUT 16점, Power TR 출력
AXT_SIO_DB32T	0x9F(159)	Digital IN 16점 / OUT 16점, Power TR 출력
AXT_SIO_RDI32	0x95	Digital IN 32점

AXT_SIO_RDO32	0x96	Digital OUT 32점
AXT_SIO_RDB128MLII	0x94	Digital IN 64점 / OUT 64점

Example

C++

```
// 0번째 모듈의 베이스 보드 번호, 모듈 번호, 모듈 ID를 확인한다.
long lBoardNo;
long lModulePos;
DWORD dwModuleID;
CString strData;
AxdInfoGetModule(0, &lBoardNo, &lModulePos, &dwModuleID);
strData.Format("BoardNo=%d, ModulePos=%d, dwModuleID=%02Xh", lBoardNo, lModulePos,
    dwModuleID);
AfxMessageBox(strData);
```

Visual Basic

```
' 0번째 모듈의 베이스 보드 번호, 모듈 번호, 모듈 ID를 확인한다.
Dim lBoardNo As Long
Dim lModulePos As Long
Dim dwModuleID As DWord
Dim strData As String
AxdInfoGetModule 0, lBoardNo, lModulePos, dwModuleID
strData = "BoardNo=" + CStr(lBoardNo) + ", ModulePos=" + CStr(lModulePos) + ",
    ModuleID=" + CStr(dwModuleID)
MsgBox strData
```

Delphi

```
{ 0번째 모듈의 베이스 보드 번호, 모듈 번호, 모듈 ID를 확인한다. }
var
  BoardNo : LongInt;
  lModulePos : LongInt;
  dwModuleID : DWord;
  strData : String;

begin
  AxdInfoGetModule(0, @lBoardNo, @lModulePos, @dwModuleID);
  strData := 'BoardNo=' + IntToStr(lBoardNo) + ', ModulePos=' + IntToStr(lModulePos) +
    ', ModuleID=' + IntToStr(dwModuleID);
  Application.MessageBox (PCHAR(strData), 'Ajinextek', MB_OK);
end;
```

See Also

[AxdInfoIsDIOModule](#), [AxdInfoGetModuleNo](#), [AxdInfoGetModuleCount](#), [AxdInfoGetInputCount](#), [AxdInfoGetOutputCount](#)

AxdInfoGetModuleStatus

Purpose

지정한 모듈번호로 해당 모듈이 제어가 가능한 상태인지 반환한다.

Format

C++

```
DWORD AxdInfoGetModuleStatus (long lModuleNo);
```

Visual Basic

```
Function AxdInfoGetModuleStatus (ByVal lModuleNo As Long) As Long
```

Delphi

```
Function AxdInfoGetModuleStatus (lModuleNo : LongInt) : DWord;
  stdcall;
```

C#

```
AxdInfoGetModuleStatus (Int lModuleNo);
```

Parameters

[in/out] Name	[Init Value] Explanation
[>>] lModuleNo	모듈 번호 (0 ~ N-1)

Return Values

[0000] AXT_RT_SUCCESS : AXL 라이브러리 초기화 성공

[3051] AXT_RT_DIO_NOT_MODULE : DIO 모듈 없음

[3101] AXT_RT_DIO_INVALID_MODULE_NO : 유효하지않는 DIO 모듈 번호

* See error code Table for more information on status error codes

Description

사용자가 지정한 모듈 번호로 해당 모듈이 제어가 가능한 상태인지 반환 한다. 제어 불가능한 상태라면 AXT_RT_DIO_NOT_MODULE Return 값을 반환한다.

Example

C++

```
// 0번째 모듈의 제어 가능한 상태를 확인 한다.
DWORD dwReturn;
CString strData;

dwReturn = AxdInfoGetModuleStatus (0);

strData.Format ("0번 모듈의 상태는 %d 입니다.", dwReturn);
AfxMessageBox(strData);
```

Visual Basic

See Also

[AxdInfoIsDIOModule](#), [AxdInfoGetModuleNo](#), [AxdInfoGetModuleCount](#), [AxdInfoGetInputCount](#),
[AxdInfoGetOutputCount](#), [AxdInfoGetModuleStatus](#)

인터럽트

인터럽트 설정 확인

Function	Description
AxdilInterruptSetModule	지정 모듈의 인터럽트 메시지를 받아오기 위하여 윈도우 메시지, 콜백 함수 또는 이벤트 방식을 사용한다.
AxdilInterruptSetModuleEnable	지정한 모듈의 인터럽트 사용 유무를 설정한다.
AxdilInterruptGetModuleEnable	지정한 모듈의 인터럽트 사용 유무를 확인한다.
AxdilInterruptRead	인터럽트 발생 위치를 확인한다.

인터럽트 상승 / 하강 에지 설정 확인

Function	Description
AxdilInterruptEdgeSetBit	지정한 입력 접점 모듈, Interrupt Rising / Falling Edge Register의 Offset 위치에서 bit 단위로 상승 또는 하강 에지 값을 설정한다.
AxdilInterruptEdgeSetByte	지정한 입력 접점 모듈, Interrupt Rising / Falling Edge Register의 Offset 위치에서 byte 단위로 상승 또는 하강 에지 값을 설정한다.
AxdilInterruptEdgeSetWord	지정한 입력 접점 모듈, Interrupt Rising / Falling Edge Register의 Offset 위치에서 word 단위로 상승 또는 하강 에지 값을 설정한다.
AxdilInterruptEdgeSetDword	지정한 입력 접점 모듈, Interrupt Rising / Falling Edge Register의 Offset 위치에서 double word 단위로 상승 또는 하강 에지 값을 설정한다.
AxdilInterruptEdgeGetBit	지정한 입력 접점 모듈, Interrupt Rising / Falling Edge Register의 Offset 위치에서 bit 단위로 상승 또는 하강 에지 값을 확인한다.
AxdilInterruptEdgeGetByte	지정한 입력 접점 모듈, Interrupt Rising / Falling Edge Register의 Offset 위치에서 Byte 단위로 상승 또는 하강 에지 값을 확인한다.
AxdilInterruptEdgeGetWord	지정한 입력 접점 모듈, Interrupt Rising / Falling Edge Register의 Offset 위치에서 word 단위로 상승 또는 하강 에지 값을 확인한다.
AxdilInterruptEdgeGetDword	지정한 입력 접점 모듈, Interrupt Rising / Falling Edge Register의 Offset 위치에서 double word 단위로 상승 또는 하강 에지 값을 확인한다.
AxdilInterruptEdgeSet	전체 입력 접점 모듈, Interrupt Rising / Falling Edge Register의 Offset 위치에서 bit 단위로 상승 또는 하강 에지 값을 설정한다.
AxdilInterruptEdgeGet	전체 입력 접점 모듈, Interrupt Rising / Falling Edge Register의 Offset 위치에서 bit 단위로 상승 또는 하강 에지 값을 확인한다.

AxdiInterruptSetModule

Purpose

지정한 모듈의 인터럽트 메시지를 받아오기 위하여 원도우 메시지, 콜백 함수 또는 이벤트 방식을 사용한다.

Format

C++

```
DWORD AxdiInterruptSetModule(long lModuleNo, HWND hWnd, DWORD
uMessage, AXT_INTERRUPT_PROC pProc, HANDLE *pEvent);
```

Visual Basic

```
Function AxdiInterruptSetModule(ByVal lModuleNo As Long, ByVal hWnd
As Long, ByVal uMessage As Long, ByVal pProc As Long, ByRef pEvent As
Long) As Long
```

Delphi

```
function AxdiInterruptSetModule(lModuleNo : LongInt; hWnd : HWND;
uMessage : DWord; pProc : AXT_INTERRUPT_PROC; pEvent : PDWord) :
DWord; stdcall;
```

Parameters

[in/out] Name	[Init Value] Explanation
[>>]lModuleNo	입력 모듈 번호 (0 ~ N-1)
[>>]hWnd	원도우 핸들
[>>]uMessage	원도우 메시지
[>>]pProc	콜백 함수
[<<]pEvent	이벤트 ID

Return Values

[0000] AXT_RT_SUCCESS : AXL 라이브러리 초기화 성공

[3051] AXT_RT_DIO_NOT_MODULE : DIO 모듈 없음

[3101] AXT_RT_DIO_INVALID_MODULE_NO : 유효하지않는 DIO 모듈 번호

* See error code Table for more information on status error codes

Description

인터럽트는 모듈 단위로 받을 수 있다. 인터럽트 메시지를 받아오기 위해서는 원도우 핸들을 넘기고 콜백 함수를 사용할 때는 함수 포인터를 넘겨준다. 이벤트 방식을 사용할 때는 이벤트 ID 를 가져와서 인터럽트를 받는다.

함수 인자는 아래표를 참고한다.

인자	설명
lModuleNo	인터럽트를 받을 모듈 번호

hwnd	윈도우 핸들. 윈도우 메시지를 받을 때 사용. 사용하지 않으면 NULL
uMessage	윈도우 핸들의 메시지. 사용하지 않거나 디폴트값을 사용하려면 0
Proc	인터럽트 발생시 호출될 함수의 포인터. 사용하지 않으면 NULL
pEvent	이벤트 ID. 이벤트 방식일 경우 사용. 사용하지 않으면 NULL

Example

C++

```
// 0번 모듈의 인터럽트 메시지를 받아오기 위하여 윈도우 메시지 사용
// Header file에 추가
afx_msg LRESULT OnInterruptMessage (WPARAM wParam, LPARAM lParam) ;

BEGIN_MESSAGE_MAP(CDiodlg, CDialog)
//{{AFX_MSG_MAP
// 선언
ON_MESSAGE(WM_AXI_INTERRUPT, OnInterruptMessage)
//}}AFX_MSG_MAP
END_MESSAGE_MAP()

// 인터럽트 발생시 'OnInterruptMessage()' 호출됨
AxdiInterruptSetModule(0, m_hWnd, WM_AXI_INTERRUPT, NULL, NULL);
```

Visual Basic

```
' 0번 모듈의 인터럽트 메시지를 받아오기 위하여 윈도우 메시지 사용
' 선언
Private Sub AxtMsg1_OnMessage1(ByVal wParam As Long, ByVal lParam As Long)
' 인터럽트 메시지 처리 구문
End Sub

' 인터럽트 발생시 'AxtMsg1_OnMessage1()' 호출됨
AxtMsg1.Message1 = WM_AXI_INTERRUPT

AxdiInterruptSetModule 0, Me.hWnd, WM_AXI_INTERRUPT, 0, 0
```

Delphi

```
{ 0번 모듈의 인터럽트 메시지를 받아오기 위하여 윈도우 메시지 사용 }
{ 선언 }
procedure TForm1.OnInterruptMessage(var Msg : TMessage);
begin
{ 인터럽트 메시지 처리 구문 }
end;

{ 인터럽트 발생시 'OnInterruptMessage()' 호출됨 }
AxdiInterruptSetModule(0, Form1.Handle, WM_AXI_INTERRUPT, nil, nil);
```

See Also

[AxdiInterruptSetModuleEnable](#), [AxdiInterruptGetModuleEnable](#), [AxdiInterruptRead](#)

AxdiInterruptSetModuleEnable

Purpose

지정한 모듈의 인터럽트 사용 유무를 설정한다.

Format

C++

```
DWORD AxdiInterruptSetModuleEnable(long lpModuleNo, DWORD uUse);
```

Visual Basic

```
Function AxdiInterruptSetModuleEnable(ByVal lpModuleNo As Long, ByVal uUse As Long) As Long
```

Delphi

```
function AxdiInterruptSetModuleEnable(lpModuleNo : LongInt; uUse : DWord) : DWord; stdcall;
```

Parameters

[in/out] Name	[Init Value] Explanation
[>>]lpModuleNo	입력 모듈 번호 (0 ~ N-1)
[>>]uUse	인터럽트 사용여부: AXT_USE - [00h] 인터럽트 사용 해제 - [01h] 인터럽트 사용 설정

Return Values

[0000] AXT_RT_SUCCESS : AXL 라이브러리 초기화 성공

[3051] AXT_RT_DIO_NOT_MODULE : DIO 모듈 없음

[3101] AXT_RT_DIO_INVALID_MODULE_NO : 유효하지않는 DIO 모듈 번호

* See error code Table for more information on status error codes

Description

사용자가 지정한 모듈의 인터럽트 사용 유무를 설정한다.

Example

C++

```
// 0번째 모듈의 인터럽트를 사용 가능하게 설정한다.  
AxdiInterruptSetModuleEnable(0, ENABLE);
```

Visual Basic

```
' 0번째 모듈의 인터럽트를 사용 가능하게 설정한다.  
AxdiInterruptSetModuleEnable 0, ENABLE
```

Delphi

```
{ 0번째 모듈의 인터럽트를 사용 가능하게 설정한다. }  
AxdiInterruptSetModuleEnable(0, ENABLE);
```

See Also

[AxdIInterruptSetModule](#), [AxdIInterruptGetModuleEnable](#), [AxdIInterruptRead](#)

AxdiInterruptGetModuleEnable

Purpose

지정한 모듈의 인터럽트 사용 유무를 확인한다.

Format

C++

```
DWORD AxdiInterruptGetModuleEnable(long lpModuleNo, DWORD *upUse);
```

Visual Basic

```
Function AxdiInterruptGetModuleEnable(ByVal lpModuleNo As Long, ByRef upUse As Long) As Long
```

Delphi

```
function AxdiInterruptGetModuleEnable(lpModuleNo : LongInt; upUse : PDWord) : DWord; stdcall;
```

Parameters

[in/out] Name	[Init Value] Explanation
[>>]lpModuleNo	입력 모듈 번호 (0 ~ N-1)
[<<]upUse	인터럽트 사용여부: AXT_USE - [00h] 인터럽트 사용 해제 - [01h] 인터럽트 사용 설정

Return Values

[0000] AXT_RT_SUCCESS : AXL 라이브러리 초기화 성공

[3051] AXT_RT_DIO_NOT_MODULE : DIO 모듈 없음

[3101] AXT_RT_DIO_INVALID_MODULE_NO : 유효하지않는 DIO 모듈 번호

* See error code Table for more information on status error codes

Description

사용자가 지정한 모듈의 인터럽트 사용 유무를 확인한다.

Example

C++

```
// 0번 입력 모듈의 인터럽트 사용 가능 여부를 확인한다.
DWORD dwUse;
AxdiInterruptGetModuleEnable(0, &dwUse);
if (dwUse)
{
    AfxMessageBox("0번 모듈은 인터럽트 사용 가능합니다.");
}
else
{
    AfxMessageBox("0번 모듈은 인터럽트 사용 불가능합니다.");
}
```

Visual Basic

```
' 0번 입력 모듈의 인터럽트 사용 가능 여부를 확인한다.  
Dim lUse As Long  
  
AxdiInterruptGetModuleEnable 0, lUse  
  
If lUse = ENABLE Then  
    MsgBox "0번 모듈은 인터럽트 사용 가능합니다."  
Else  
    MsgBox "0번 모듈은 인터럽트 사용 불가능합니다."  
End If
```

Delphi

```
{ 0번 입력 모듈의 인터럽트 사용 가능 여부를 확인한다. }  
var  
    dwUse : DWord;  
  
begin  
    AxdiInterruptGetModuleEnable(0, @dwUse);  
    if (dwUse = ENABLE) then  
        Application.MessageBox ('0번 모듈은 인터럽트 사용 가능합니다.', 'Ajinextek', MB_OK)  
    else  
        Application.MessageBox ('0번 모듈은 인터럽트 사용 불가능합니다.', 'Ajinextek', MB_OK);  
end;
```

See Also

[AxdiInterruptSetModule](#), [AxdiInterruptSetModuleEnable](#), [AxdiInterruptRead](#)

AxdiInterruptRead

Purpose

인터럽트 발생 위치를 확인한다.

Format

C++

```
DWORD AxdiInterruptRead(long *lpModuleNo, DWORD *upFlag);
```

Visual Basic

```
Function AxdiInterruptRead(ByRef lpModuleNo As Long, ByRef upFlag As Long) As Long
```

Delphi

```
function AxdiInterruptRead(lpModuleNo : PLongInt; upFlag : PDWord) : DWord; stdcall;
```

Parameters

[in/out] Name	[Init Value] Explanation
[<<]lpModuleNo	모듈 번호
[<<]upFlag	인터럽트 발생 위치

Return Values

[0000] AXT_RT_SUCCESS : AXL 라이브러리 초기화 성공

[3052] AXT_RT_DIO_NOT_INTERRUPT: DIO 인터럽트 설정안됨

* See error code Table for more information on status error codes

Description

이벤트 방식의 인터럽트를 사용할 경우 인터럽트 발생 위치를 확인한다.

Example

C++

```
// 0번째 모듈의 인터럽트 발생 위치를 확인한다.  
DWORD dwFlag;  
  
AxdiInterruptRead(0, &dwFlag);
```

Visual Basic

```
' 0번째 모듈의 인터럽트 발생 위치를 확인한다.  
Dim lFlag As long  
  
AxdiInterruptRead 0, lFlag
```

Delphi

```
{ 0번째 모듈의 인터럽트 발생 위치를 확인한다. }
var
    dwFlag : DWord;

begin
    AxdiInterruptRead(0, @dwFlag);
end;
```

See Also

[AxdiInterruptSetModule](#), [AxdiInterruptSetModuleEnable](#), [AxdiInterruptGetModuleEnable](#)

AxdiInterruptEdgeSetBit

Purpose

지정한 입력 접점 모듈, Interrupt Rising / Falling Edge Register 의 Offset 위치에서 bit 단위로 상승 또는 하강 에지 값을 설정한다.

Format

C++

```
DWORD AxdiInterruptEdgeSetBit(long lModuleNo, long lOffset, DWORD uMode, DWORD uValue);
```

Visual Basic

```
Function AxdiInterruptEdgeSetBit(ByVal lModuleNo As Long, ByVal lOffset As Long, ByVal uMode As Long, ByVal uValue As Long) As Long
```

Delphi

```
function AxdiInterruptEdgeSetBit(lModuleNo : LongInt; lOffset : LongInt; uMode : DWord; uValue : DWord) : DWord; stdcall;
```

Parameters

[in/out] Name	[Init Value] Explanation
[>>]lModuleNo	모듈 번호
[>>]lOffset	입력 접점에 대한 Offset 위치
[>>]uMode	인터럽트 발생에지 설정: AXT_DIO_EDGE - [00h] 하강 에지 - [01h] 상승 에지
[>>]uValue	인터럽트 발생 설정 여부 (Boolean)

Return Values

[0000] AXT_RT_SUCCESS : AXL 라이브러리 초기화 성공

[3051] AXT_RT_DIO_NOT_MODULE : DIO 모듈 없음

[3101] AXT_RT_DIO_INVALID_MODULE_NO : 유효하지않는 DIO 모듈 번호

[3104] AXT_RT_DIO_INVALID_MODE : 유효하지 않는 Edge 를 설정했을 때

* See error code Table for more information on status error codes

Description

사용자가 지정한 입력 접점 모듈, Interrupt Rising / Falling Edge Register 의 Offset 위치에서 bit 단위로 레지스터 값을 설정한다.

지정한 모듈이 어떤 모듈인지 확인하여 아래의 Table 을 참조 하여 Offset 을 정한다.

Offset 범위와 Value 값

Module	Offset	Value
SIO-DI32	0 ~ 31	0(Off), 1(On)
SIO-DB32x	0 ~ 15	0(Off), 1(On)

Example

C++

```
// 0번째 입력 접점 모듈, Interrupt Rising Edge Register의 Offset 위치에서 bit 단위로 상승 에지  
// 값을 설정한다.  
AxdiInterruptEdgeSetBit(0, 0, UP_EDGE, 1);
```

Visual Basic

```
' 0번째 입력 접점 모듈, Interrupt Rising Edge Register의 Offset 위치에서 bit 단위로 상승 에지  
' 값을 설정한다.  
AxdiInterruptEdgeSetBit 0, 0, UP_EDGE, 1
```

Delphi

```
{ 0번째 입력 접점 모듈, Interrupt Rising Edge Register의 Offset 위치에서 bit 단위로 상승 에지 값을  
설정한다. }  
AxdiInterruptEdgeSetBit(0, 0, UP_EDGE, 1);
```

See Also

[AxdiInterruptEdgeSetByte](#), [AxdiInterruptEdgeSetWord](#), [AxdiInterruptEdgeSetDword](#),
[AxdiInterruptEdgeGetBit](#), [AxdiInterruptEdgeGetByte](#), [AxdiInterruptEdgeGetWord](#),
[AxdiInterruptEdgeGetDword](#), [AxdiInterruptEdgeSet](#), [AxdiInterruptEdgeGet](#)

AxdiInterruptEdgeSetByte

Purpose

지정한 입력 접점 모듈, Interrupt Rising / Falling Edge Register 의 Offset 위치에서 byte 단위로 상승 또는 하강 에지 값을 설정한다.

Format

C++

```
DWORD AxdiInterruptEdgeSetByte(long lModuleNo, long lOffset, DWORD uMode, DWORD uValue);
```

Visual Basic

```
Function AxdiInterruptEdgeSetByte(ByVal lModuleNo As Long, ByVal lOffset As Long, ByVal uMode As Long, ByVal uValue As Long) As Long
```

Delphi

```
function AxdiInterruptEdgeSetByte(lModuleNo : LongInt; lOffset : LongInt; uMode : DWord; uValue : DWord) : DWord; stdcall;
```

Parameters

[in/out] Name	[Init Value] Explanation
[>>]lModuleNo	모듈 번호
[>>]lOffset	입력 접점에 대한 Offset 위치
[>>]uMode	인터럽트 발생에지 설정: AXT_DIO_EDGE - [00h] 하강 에지 - [01h] 상승 에지
[>>]uValue	인터럽트 발생 설정 여부 (Byte)

Return Values

[0000] AXT_RT_SUCCESS : AXL 라이브러리 초기화 성공

[3051] AXT_RT_DIO_NOT_MODULE : DIO 모듈 없음

[3101] AXT_RT_DIO_INVALID_MODULE_NO : 유효하지않는 DIO 모듈 번호

[3104] AXT_RT_DIO_INVALID_MODE : 유효하지 않는 Edge 를 설정했을 때

* See error code Table for more information on status error codes

Description

사용자가 지정한 입력 접점 모듈, Interrupt Rising / Falling Edge Register 의 Offset 위치에서 byte 단위로 상승 또는 하강 에지 값을 설정한다.

지정한 모듈이 어떤 모듈인지 확인하여 아래의 Table에서 Offset 사용 범위를 참조하여 설정한다.

Offset 범위와 Value 값

Module	Offset	Value
SIO-DI32	0, 1, 2, 3	00h ~ FFh
SIO-DB32x	0, 1	00h ~ FFh

Example

C++

```
// 0번째 입력 접점 모듈, Interrupt Rising Edge Register의 Offset 위치에서 byte 단위로 상승 에지  
// 값을 설정한다.  
AxdiInterruptEdgeSetByte(0, 0, UP_EDGE, 0xFF);
```

Visual Basic

```
' 0번째 입력 접점 모듈, Interrupt Rising Edge Register의 Offset 위치에서 byte 단위로 상승 에지  
' 값을 설정한다.  
AxdiInterruptEdgeSetByte 0, 0, UP_EDGE, &hFF
```

Delphi

```
{ 0번째 입력 접점 모듈, Interrupt Rising Edge Register의 Offset 위치에서 byte 단위로 상승 에지 값을  
설정한다. }  
AxdiInterruptEdgeSetByte(0, 0, UP_EDGE, $FF);
```

See Also

[AxdiInterruptEdgeSetBit](#), [AxdiInterruptEdgeSetWord](#), [AxdiInterruptEdgeSetDword](#),
[AxdiInterruptEdgeGetBit](#), [AxdiInterruptEdgeGetByte](#), [AxdiInterruptEdgeGetWord](#),
[AxdiInterruptEdgeGetDword](#), [AxdiInterruptEdgeSet](#), [AxdiInterruptEdgeGet](#)

AxdiInterruptEdgeSetWord

Purpose

지정한 입력 접점 모듈, Interrupt Rising / Falling Edge Register 의 Offset 위치에서 word 단위로 상승 또는 하강 에지 값을 설정한다.

Format

C++

```
DWORD AxdiInterruptEdgeSetWord(long lModuleNo, long lOffset, DWORD uMode, DWORD uValue);
```

Visual Basic

```
Function AxdiInterruptEdgeSetWord(ByVal lModuleNo As Long, ByVal lOffset As Long, ByVal uMode As Long, ByVal uValue As Long) As Long
```

Delphi

```
function AxdiInterruptEdgeSetWord(lModuleNo : LongInt; lOffset : LongInt; uMode : DWord; uValue : DWord) : DWord; stdcall;
```

Parameters

[in/out] Name	[Init Value] Explanation
[>>]lModuleNo	모듈 번호
[>>]lOffset	입력 접점에 대한 Offset 위치
[>>]uMode	인터럽트 발생에지 설정: AXT_DIO_EDGE - [00h] 하강 에지 - [01h] 상승 에지
[>>]uValue	인터럽트 발생 설정 여부 (Word)

Return Values

[0000] AXT_RT_SUCCESS : AXL 라이브러리 초기화 성공

[3051] AXT_RT_DIO_NOT_MODULE : DIO 모듈 없음

[3101] AXT_RT_DIO_INVALID_MODULE_NO : 유효하지않는 DIO 모듈 번호

[3104] AXT_RT_DIO_INVALID_MODE : 유효하지 않는 Edge 를 설정했을 때

* See error code Table for more information on status error codes

Description

사용자가 지정한 입력 접점 모듈, Interrupt Rising / Falling Edge Register 의 Offset 위치에서 word 단위로 상승 또는 하강 에지 값을 설정한다.

지정한 모듈이 어떤 모듈인지 확인하여 아래의 Table에서 Offset 사용 범위를 참조하여 설정한다.

Offset 범위와 Value 값

Module	Offset	Value
SIO-DI32	0, 1	0000h ~ FFFFh
SIO-DB32x	0	0000h ~ FFFFh

Example

C++

```
// 0번째 입력 접점 모듈, Interrupt Rising Edge Register의 Offset 위치에서 word 단위로 상승 에지  
// 값을 설정한다.  
AxdiInterruptEdgeSetWord(0, 0, UP_EDGE, 0xFFFF);
```

Visual Basic

```
' 0번째 입력 접점 모듈, Interrupt Rising Edge Register의 Offset 위치에서 word 단위로 상승 에지 값을  
' 설정한다.  
AxdiInterruptEdgeSetWord 0, 0, UP_EDGE, &hFFFF
```

Delphi

```
{ 0번째 입력 접점 모듈, Interrupt Rising Edge Register의 Offset 위치에서 word 단위로 상승 에지 값을  
설정한다. }  
AxdiInterruptEdgeSetWord(0, 0, UP_EDGE, $FFFF);
```

See Also

[AxdiInterruptEdgeSetBit](#), [AxdiInterruptEdgeSetByte](#), [AxdiInterruptEdgeSetDword](#),
[AxdiInterruptEdgeGetBit](#), [AxdiInterruptEdgeGetByte](#), [AxdiInterruptEdgeGetWord](#),
[AxdiInterruptEdgeGetDword](#), [AxdiInterruptEdgeSet](#), [AxdiInterruptEdgeGet](#)

AxdiInterruptEdgeSetDword

Purpose

지정한 입력 접점 모듈, Interrupt Rising / Falling Edge Register 의 Offset 위치에서 double word 단위로 상승 또는 하강 에지 값을 설정한다.

Format

C++

```
DWORD AxdiInterruptEdgeSetDword(long lModuleNo, long lOffset, DWORD uMode, DWORD uValue);
```

Visual Basic

```
Function AxdiInterruptEdgeSetDword(ByVal lModuleNo As Long, ByVal lOffset As Long, ByVal uMode As Long, ByVal uValue As Long) As Long
```

Delphi

```
function AxdiInterruptEdgeSetDword(lModuleNo : LongInt; lOffset : LongInt; uMode : DWord; uValue : DWord) : DWord; stdcall;
```

Parameters

[in/out] Name	[Init Value] Explanation
[>>]lModuleNo	모듈 번호
[>>]lOffset	입력 접점에 대한 Offset 위치
[>>]uMode	인터럽트 발생에지 설정: AXT_DIO_EDGE - [00h] 하강 에지 - [01h] 상승 에지
[>>]uValue	인터럽트 발생 설정 여부 (Double Word)

Return Values

[0000] AXT_RT_SUCCESS : AXL 라이브러리 초기화 성공

[3051] AXT_RT_DIO_NOT_MODULE : DIO 모듈 없음

[3101] AXT_RT_DIO_INVALID_MODULE_NO : 유효하지않는 DIO 모듈 번호

[3104] AXT_RT_DIO_INVALID_MODE : 유효하지 않는 Edge 를 설정했을 때

* See error code Table for more information on status error codes

Description

사용자가 지정한 입력 접점 모듈, Interrupt Rising / Falling Edge Register 의 Offset 위치에서 double word 단위로 상승 또는 하강 에지 값을 설정한다.

지정한 모듈이 어떤 모듈인지 확인하여 아래의 Table에서 Offset 사용 범위를 참조하여 설정한다.

Offset 범위와 Value 값

Module	Offset	Value
SIO-DI32	0	00000000h ~ FFFFFFFFh
SIO-DB32x	0	00000000h ~ 0000FFFFh

Example

C++

```
// 0번째 입력 접점 모듈, Interrupt Rising Edge Register의 Offset 위치에서 double word 단위로  
// 상승 에지 값을 설정한다.  
AxdiInterruptEdgeSetDword(0, 0, UP_EDGE, 0xFFFFFFFF);
```

Visual Basic

```
' 0번째 입력 접점 모듈, Interrupt Rising Edge Register의 Offset 위치에서 double word 단위로 상승  
' 에지 값을 설정한다.  
AxdiInterruptEdgeSetDword 0, 0, UP_EDGE, &hFFFFFF
```

Delphi

```
{ 0번째 입력 접점 모듈, Interrupt Rising Edge Register의 Offset 위치에서 double word 단위로 상승  
에지 값을 설정한다. }  
AxdiInterruptEdgeSetDword(0, 0, UP_EDGE, $FFFFFF);
```

See Also

[AxdiInterruptEdgeSetBit](#), [AxdiInterruptEdgeSetByte](#), [AxdiInterruptEdgeSetWord](#),
[AxdiInterruptEdgeGetBit](#), [AxdiInterruptEdgeGetByte](#), [AxdiInterruptEdgeGetWord](#),
[AxdiInterruptEdgeGetDword](#), [AxdiInterruptEdgeSet](#), [AxdiInterruptEdgeGet](#)

AxdiInterruptEdgeGetBit

Purpose

지정한 입력 접점 모듈, Interrupt Rising / Falling Edge Register 의 Offset 위치에서 bit 단위로 상승 또는 하강 에지 값을 확인한다.

Format

C++

```
DWORD AxdiInterruptEdgeGetBit(long lModuleNo, long lOffset, DWORD
uMode, DWORD *upValue);
```

Visual Basic

```
Function AxdiInterruptEdgeGetBit(ByVal lModuleNo As Long, ByVal
lOffset As Long, ByVal uMode As Long, ByRef upValue As Long) As Long
```

Delphi

```
function AxdiInterruptEdgeGetBit(lModuleNo : LongInt; lOffset :
LongInt; uMode : DWord; upValue : PDWord) : DWord; stdcall;
```

Parameters

[in/out] Name	[Init Value] Explanation
[>>]lModuleNo	모듈 번호
[>>]lOffset	입력 접점에 대한 Offset 위치
[>>]uMode	인터럽트 발생에지 설정: AXT_DIO_EDGE - [00h] 하강 에지 - [01h] 상승 에지
[<<]upValue	인터럽트 발생 설정 여부 (Bit)

Return Values

[0000] AXT_RT_SUCCESS : AXL 라이브러리 초기화 성공

[3051] AXT_RT_DIO_NOT_MODULE : DIO 모듈 없음

[3101] AXT_RT_DIO_INVALID_MODULE_NO : 유효하지않는 DIO 모듈 번호

[3104] AXT_RT_DIO_INVALID_MODE : 유효하지 않는 Edge 를 설정했을 때

* See error code Table for more information on status error codes

Description

사용자가 지정한 입력 접점 모듈, Interrupt Rising / Falling Edge Register 의 Offset 위치에서 bit 단위로 상승 또는 하강 에지 값을 확인한다.

지정한 모듈이 어떤 모듈인지 확인하여 아래의 Table에서 Offset 사용 범위를 참조하여 확인한다.

Offset 범위와 Value 값

Module	Offset	Value
SIO-DI32	0 ~ 31	0(Off), 1(On)
SIO-DB32x	0 ~ 15	0(Off), 1(On)

Example

C++

```
// 0번째 입력 접점 모듈, Interrupt Rising Edge Register의 Offset 위치에서 bit 단위로 상승 에지  
// 설정 값을 읽는다.  
DWORD dwEdge  
  
AxdiInterruptEdgeGetBit(0, 0, UP_EDGE, &dwEdge);
```

Visual Basic

```
' 0번째 입력 접점 모듈, Interrupt Rising Edge Register의 Offset 위치에서 bit 단위로 상승 에지  
' 설정 값을 읽는다.  
Dim lEdge As long  
  
AxdiInterruptEdgeGetBit 0, 0, UP_EDGE, lEdge
```

Delphi

```
{ 0번째 입력 접점 모듈, Interrupt Rising Edge Register의 Offset 위치에서 bit 단위로 상승 에지 설정  
값을 읽는다. }  
var  
  dwEdge : DWord;  
  
begin  
  AxdiInterruptEdgeGetBit(0, 0, UP_EDGE, @dwEdge);  
end;
```

See Also

[AxdiInterruptEdgeSetBit](#), [AxdiInterruptEdgeSetByte](#), [AxdiInterruptEdgeSetWord](#),
[AxdiInterruptEdgeSetDword](#), [AxdiInterruptEdgeGetByte](#), [AxdiInterruptEdgeGetWord](#),
[AxdiInterruptEdgeGetDword](#), [AxdiInterruptEdgeSet](#), [AxdiInterruptEdgeGet](#)

AxdiInterruptEdgeGetByte

Purpose

지정한 입력 접점 모듈, Interrupt Rising / Falling Edge Register 의 Offset 위치에서 byte 단위로 상승 또는 하강 에지 값을 확인한다.

Format

C++

```
DWORD AxdiInterruptEdgeGetByte(long lModuleNo, long lOffset, DWORD
uMode, DWORD *upValue);
```

Visual Basic

```
Function AxdiInterruptEdgeGetByte(ByVal lModuleNo As Long, ByVal
lOffset As Long, ByVal uMode As Long, ByRef upValue As Long) As Long
```

Delphi

```
function AxdiInterruptEdgeGetByte(lModuleNo : LongInt; lOffset :
LongInt; uMode : DWord; upValue : PDWord) : DWord; stdcall;
```

Parameters

[in/out] Name	[Init Value] Explanation
[>>]lModuleNo	모듈 번호
[>>]lOffset	입력 접점에 대한 Offset 위치
[>>]uMode	인터럽트 발생에지 설정: AXT_DIO_EDGE - [00h] 하강 에지 - [01h] 상승 에지
[<<]upValue	인터럽트 발생 설정 여부 (Byte)

Return Values

[0000] AXT_RT_SUCCESS : AXL 라이브러리 초기화 성공

[3051] AXT_RT_DIO_NOT_MODULE : DIO 모듈 없음

[3101] AXT_RT_DIO_INVALID_MODULE_NO : 유효하지않는 DIO 모듈 번호

[3104] AXT_RT_DIO_INVALID_MODE : 유효하지 않는 Edge 를 설정했을 때

* See error code Table for more information on status error codes

Description

사용자가 지정한 입력 접점 모듈, Interrupt Rising / Falling Edge Register 의 Offset 위치에서 byte 단위로 상승 또는 하강 에지 값을 확인한다.

지정한 모듈이 어떤 모듈인지 확인하여 아래의 Table에서 Offset 사용 범위를 참조하여 확인한다.

Offset 범위와 Value 값

Module	Offset	Value
SIO-DI32	0, 1, 2, 3	00h ~ FFh
SIO-DB32x	0, 1	00h ~ FFh

Example

C++

```
// 0번째 입력 접점 모듈, Interrupt Rising Edge Register의 Offset 위치에서 byte 단위로 상승 에지  
// 설정 값을 읽는다.  
DWORD dwEdge  
  
AxdiInterruptEdgeGetByte(0, 0, UP_EDGE, &dwEdge);
```

Visual Basic

```
' 0번째 입력 접점 모듈, Interrupt Rising Edge Register의 Offset 위치에서 byte 단위로 상승 에지  
' 설정 값을 읽는다.  
Dim lEdge As long  
  
AxdiInterruptEdgeGetByte 0, 0, UP_EDGE, lEdge
```

Delphi

```
{ 0번째 입력 접점 모듈, Interrupt Rising Edge Register의 Offset 위치에서 byte 단위로 상승 에지 설정  
값을 읽는다. }  
var  
  dwEdge : DWord;  
  
begin  
  AxdiInterruptEdgeGetByte(0, 0, UP_EDGE, @dwEdge);  
end;
```

See Also

[AxdiInterruptEdgeSetBit](#), [AxdiInterruptEdgeSetByte](#), [AxdiInterruptEdgeSetWord](#),
[AxdiInterruptEdgeSetDword](#), [AxdiInterruptEdgeGetBit](#), [AxdiInterruptEdgeGetWord](#),
[AxdiInterruptEdgeGetDword](#), [AxdiInterruptEdgeSet](#), [AxdiInterruptEdgeGet](#)

AxdiInterruptEdgeGetWord

Purpose

지정한 입력 접점 모듈, Interrupt Rising / Falling Edge Register 의 Offset 위치에서 word 단위로 상승 또는 하강 에지 값을 확인한다.

Format

C++

```
DWORD AxdiInterruptEdgeGetWord(long lModuleNo, long lOffset, DWORD uMode, DWORD *upValue);
```

Visual Basic

```
Function AxdiInterruptEdgeGetWord(ByVal lModuleNo As Long, ByVal lOffset As Long, ByVal uMode As Long, ByRef upValue As Long) As Long
```

Delphi

```
function AxdiInterruptEdgeGetWord(lModuleNo : LongInt; lOffset : LongInt; uMode : DWord; upValue : PDWord) : DWord; stdcall;
```

Parameters

[in/out] Name	[Init Value] Explanation
[>>]lModuleNo	모듈 번호
[>>]lOffset	입력 접점에 대한 Offset 위치
[>>]uMode	인터럽트 발생에지 설정: AXT_DIO_EDGE - [00h] 하강 에지 - [01h] 상승 에지
[<<]upValue	인터럽트 발생 설정 여부 (Word)

Return Values

[0000] AXT_RT_SUCCESS : AXL 라이브러리 초기화 성공

[3051] AXT_RT_DIO_NOT_MODULE : DIO 모듈 없음

[3101] AXT_RT_DIO_INVALID_MODULE_NO : 유효하지않는 DIO 모듈 번호

[3104] AXT_RT_DIO_INVALID_MODE : 유효하지 않는 Edge 를 설정했을 때

* See error code Table for more information on status error codes

Description

사용자가 지정한 입력 접점 모듈, Interrupt Rising / Falling Edge Register 의 Offset 위치에서 word 단위로 상승 또는 하강 에지 값을 확인한다.

지정한 모듈이 어떤 모듈인지 확인하여 아래의 Table에서 Offset 사용 범위를 참조하여 확인한다.

Offset 범위와 Value 값

Module	Offset	Value
SIO-DI32	0, 1	0000h ~ FFFFh
SIO-DB32x	0	0000h ~ FFFFh

Example

C++

```
// 0번째 입력 접점 모듈, Interrupt Rising Edge Register의 Offset 위치에서 word 단위로 상승 에지  
// 설정 값을 읽는다.  
DWORD dwEdge  
  
AxdiInterruptEdgeGetWord(0, 0, UP_EDGE, &dwEdge);
```

Visual Basic

```
' 0번째 입력 접점 모듈, Interrupt Rising Edge Register의 Offset 위치에서 word 단위로 상승 에지  
' 설정 값을 읽는다.  
Dim lEdge As long  
  
AxdiInterruptEdgeGetWord 0, 0, UP_EDGE, lEdge
```

Delphi

```
{ 0번째 입력 접점 모듈, Interrupt Rising Edge Register의 Offset 위치에서 word 단위로 상승 에지 설정  
값을 읽는다. }  
var  
  dwEdge : DWord;  
  
begin  
  AxdiInterruptEdgeGetWord(0, 0, UP_EDGE, @dwEdge);  
end;
```

See Also

[AxdiInterruptEdgeSetBit](#), [AxdiInterruptEdgeSetByte](#), [AxdiInterruptEdgeSetWord](#),
[AxdiInterruptEdgeSetDword](#), [AxdiInterruptEdgeGetBit](#), [AxdiInterruptEdgeGetByte](#)
[AxdiInterruptEdgeGetDword](#), [AxdiInterruptEdgeSet](#), [AxdiInterruptEdgeGet](#)

AxdiInterruptEdgeGetDword

Purpose

지정한 입력 접점 모듈, Interrupt Rising / Falling Edge Register 의 Offset 위치에서 double word 단위로 상승 또는 하강 에지 값을 확인한다.

Format

C++

```
DWORD AxdiInterruptEdgeGetDword(long lModuleNo, long lOffset, DWORD
uMode, DWORD *upValue);
```

Visual Basic

```
Function AxdiInterruptEdgeGetDword(ByVal lModuleNo As Long, ByVal
lOffset As Long, ByVal uMode As Long, ByRef upValue As Long) As Long
```

Delphi

```
function AxdiInterruptEdgeGetDword(lModuleNo : LongInt; lOffset :
LongInt; uMode : DWord; upValue : PDWord) : DWord; stdcall;
```

Parameters

[in/out] Name	[Init Value] Explanation
[>>]lModuleNo	모듈 번호
[>>]lOffset	입력 접점에 대한 Offset 위치
[>>]uMode	인터럽트 발생에지 설정: AXT_DIO_EDGE - [00h] 하강 에지 - [01h] 상승 에지
[<<]upValue	인터럽트 발생 설정 여부 (Double Word)

Return Values

[0000] AXT_RT_SUCCESS : AXL 라이브러리 초기화 성공

[3051] AXT_RT_DIO_NOT_MODULE : DIO 모듈 없음

[3101] AXT_RT_DIO_INVALID_MODULE_NO : 유효하지않는 DIO 모듈 번호

[3104] AXT_RT_DIO_INVALID_MODE : 유효하지 않는 Edge 를 설정했을 때

* See error code Table for more information on status error codes

Description

사용자가 지정한 입력 접점 모듈, Interrupt Rising / Falling Edge Register 의 Offset 위치에서 double word 단위로 상승 또는 하강 에지 값을 확인한다.

지정한 모듈이 어떤 모듈인지 확인하여 아래의 Table에서 Offset 사용 범위를 참조하여 확인한다.

Offset 범위와 Value 값

Module	Offset	Value
SIO-DI32	0	00000000h ~ FFFFFFFFh
SIO-DB32x	0	00000000h ~ 0000FFFFh

Example

C++

```
// 0번째 입력 접점 모듈, Interrupt Rising Edge Register의 Offset 위치에서 double word 단위로  
// 상승 에지 설정 값을 읽는다.  
DWORD dwEdge  
  
AxdiInterruptEdgeGetDword(0, 0, UP_EDGE, &dwEdge);
```

Visual Basic

```
' 0번째 입력 접점 모듈, Interrupt Rising Edge Register의 Offset 위치에서 double word 단위로  
' 상승 에지 설정 값을 읽는다.  
Dim lEdge As long  
  
AxdiInterruptEdgeGetDword 0, 0, UP_EDGE, lEdge
```

Delphi

```
{ 0번째 입력 접점 모듈, Interrupt Rising Edge Register의 Offset 위치에서 double word 단위로 상승  
에지 설정 값을 읽는다. }  
var  
  dwEdge : DWord;  
  
begin  
  AxdiInterruptEdgeGetDword(0, 0, UP_EDGE, @dwEdge);  
end;
```

See Also

[AxdiInterruptEdgeSetBit](#), [AxdiInterruptEdgeSetByte](#), [AxdiInterruptEdgeSetWord](#),
[AxdiInterruptEdgeSetDword](#), [AxdiInterruptEdgeGetBit](#), [AxdiInterruptEdgeGetByte](#)
[AxdiInterruptEdgeGetWord](#), [AxdiInterruptEdgeSet](#), [AxdiInterruptEdgeGet](#)

AxdiInterruptEdgeSet

Purpose

전체 입력 접점 모듈, Interrupt Rising / Falling Edge Register 의 Offset 위치에서 bit 단위로 상승 또는 하강 에지 값을 설정한다.

Format

C++

```
DWORD AxdiInterruptEdgeSet(long lOffset, DWORD uMode, DWORD uValue);
```

Visual Basic

```
Function AxdiInterruptEdgeSet(ByVal lOffset As Long, ByVal uMode As Long, ByVal uValue As Long) As Long
```

Delphi

```
function AxdiInterruptEdgeSet(lOffset : LongInt; uMode : DWord;
uValue : DWord) : DWord; stdcall;
```

Parameters

[in/out] Name	[Init Value] Explanation
[>>]lOffset	입력 접점에 대한 Offset 위치
[>>]uMode	인터럽트 발생에지 설정: AXT_DIO_EDGE - [00h] 하강 에지 - [01h] 상승 에지
[>>]uValue	인터럽트 발생 설정 여부 (Boolean)

Return Values

- [0000] AXT_RT_SUCCESS : AXL 라이브러리 초기화 성공
 - [3051] AXT_RT_DIO_NOT_MODULE : DIO 모듈 없음
 - [3102] AXT_RT_DIO_INVALID_OFFSET_NO : 유효하지않는 DIO OFFSET 번호
 - [3104] AXT_RT_DIO_INVALID_MODE : 유효하지 않는 Edge 를 설정했을 때
- * See error code Table for more information on status error codes

Description

전체 입력 접점 모듈, Interrupt Rising / Falling Edge Register 의 Offset 위치에서 bit 단위로 상승 또는 하강 에지 값을 설정한다.

Offset 은 '0'부터 총 입력 접점 수 - 1 까지 사용할 수 있다.

Example

C++

```
// 전체 입력 접점 모듈, Interrupt Rising Edge Register의 Offset 위치에서 bit 단위로 상승 에지 값을
// 설정한다.
AxdiInterruptEdgeSet(0, UP_EDGE, 1);
```

Visual Basic

‘ 전체 입력 접점 모듈, Interrupt Rising Edge Register의 Offset 위치에서 bit 단위로 상승 에지 값을 설정한다.
AxdiInterruptEdgeSet 0, UP_EDGE, 1

Delphi

```
{ 전체 입력 접점 모듈, Interrupt Rising Edge Register의 Offset 위치에서 bit 단위로 상승 에지 값을 설정한다. }  
AxdiInterruptEdgeSet(0, UP_EDGE, 1);
```

See Also

[AxdiInterruptEdgeSetBit](#), [AxdiInterruptEdgeSetByte](#), [AxdiInterruptEdgeSetWord](#),
[AxdiInterruptEdgeSetDword](#), [AxdiInterruptEdgeGetBit](#), [AxdiInterruptEdgeGetByte](#)
[AxdiInterruptEdgeGetWord](#), [AxdiInterruptEdgeGetDword](#), [AxdiInterruptEdgeGet](#)

AxdiInterruptEdgeGet

Purpose

전체 입력 접점 모듈, Interrupt Rising / Falling Edge Register 의 Offset 위치에서 bit 단위로 상승 또는 하강 에지 값을 확인한다.

Format

C++

```
DWORD AxdiInterruptEdgeGet (long lOffset, DWORD uMode, DWORD *upValue);
```

Visual Basic

```
Function AxdiInterruptEdgeGet (ByVal lOffset As Long, ByVal uMode As Long, ByRef upValue As Long) As Long
```

Delphi

```
function AxdiInterruptEdgeGet(lOffset : LongInt; uMode : DWord;
  upValue : PDWord) : DWord; stdcall;
```

Parameters

[in/out] Name	[Init Value] Explanation
[>>]lOffset	입력 접점에 대한 Offset 위치
[>>]uMode	인터럽트 발생에지 설정: AXT_DIO_EDGE - [00h] 하강 에지 - [01h] 상승 에지
[<<]upValue	인터럽트 발생 설정 여부 (Boolean)

Return Values

- [0000] AXT_RT_SUCCESS : AXL 라이브러리 초기화 성공
- [3051] AXT_RT_DIO_NOT_MODULE : DIO 모듈 없음
- [3102] AXT_RT_DIO_INVALID_OFFSET_NO : 유효하지않는 DIO OFFSET 번호
- [3104] AXT_RT_DIO_INVALID_MODE : 유효하지 않는 Edge 를 설정했을 때
- * See error code Table for more information on status error codes

Description

전체 입력 접점 모듈, Interrupt Rising / Falling Edge Register 의 Offset 위치에서 bit 단위로 상승 또는 하강 에지 값을 확인한다.

Offset 은 '0'부터 총 입력 접점 수 - 1 까지 사용할 수 있다.

Example

C++

```
// 전체 입력 접점 모듈, Interrupt Rising Edge Register의 Offset 위치에서 bit 단위로 상승 에지 설정
// 값을 읽는다.
DWORD dwEdge

AxdiInterruptEdgeGet (0, UP_EDGE, &dwEdge);

AxdiInterruptEdgeGet (0, UP_EDGE, &dwEdge);
```

Visual Basic

```
' 전체 입력 접점 모듈, Interrupt Rising Edge Register의 Offset 위치에서 bit 단위로 상승 에지 설정  
' 값을 읽는다.  
Dim lEdge As long  
  
AxdiInterruptEdgeGet 0, UP_EDGE, lEdge
```

Delphi

```
{ 전체 입력 접점 모듈, Interrupt Rising Edge Register의 Offset 위치에서 bit 단위로 상승 에지 설정 값  
을 읽는다. }  
var  
    dwEdge : DWord;  
  
begin  
    AxdiInterruptEdgeGet(0, UP_EDGE, @dwEdge);  
end;
```

See Also

[AxdiInterruptEdgeSetBit](#), [AxdiInterruptEdgeSetByte](#), [AxdiInterruptEdgeSetWord](#),
[AxdiInterruptEdgeSetDword](#), [AxdiInterruptEdgeGetBit](#), [AxdiInterruptEdgeGetByte](#)
[AxdiInterruptEdgeGetWord](#), [AxdiInterruptEdgeGetDword](#), [AxdiInterruptEdgeSet](#)

입력 레벨 설정 확인

Function	Description
<u>AxdILevelSetInportBit</u>	지정한 입력 접점 모듈의 Offset 위치에서 bit 단위로 레벨을 설정한다.
<u>AxdILevelSetInportByte</u>	지정한 입력 접점 모듈의 Offset 위치에서 byte 단위로 레벨을 설정한다.
<u>AxdILevelSetInportWord</u>	지정한 입력 접점 모듈의 Offset 위치에서 word 단위로 레벨을 설정한다.
<u>AxdILevelSetInportDword</u>	지정한 입력 접점 모듈의 Offset 위치에서 double word 단위로 레벨을 설정한다.
<u>AxdILevelGetInportBit</u>	지정한 입력 접점 모듈의 Offset 위치에서 bit 단위로 레벨을 확인한다.
<u>AxdILevelGetInportByte</u>	지정한 입력 접점 모듈의 Offset 위치에서 byte 단위로 레벨을 확인한다.
<u>AxdILevelGetInportWord</u>	지정한 입력 접점 모듈의 Offset 위치에서 word 단위로 레벨을 확인한다.
<u>AxdILevelGetInportDword</u>	지정한 입력 접점 모듈의 Offset 위치에서 double word 단위로 레벨을 확인한다.
<u>AxdILevelSetInport</u>	전체 입력 접점 모듈의 Offset 위치에서 bit 단위로 레벨을 설정한다.
<u>AxdILevelGetInport</u>	전체 입력 접점 모듈의 Offset 위치에서 bit 단위로 레벨을 확인한다.

AxdiLevelSetImportBit

Purpose

지정한 입력 접점 모듈의 Offset 위치에서 bit 단위로 레벨을 설정한다.

레벨 설정은 신호의 Active 상태를 HIGH(1)로 할지 LOW(0)로 할지 설정한다.

Format

C++

```
DWORD AxdiLevelSetImportBit(long lModuleNo, long lOffset, DWORD uLevel);
```

Visual Basic

```
Function AxdiLevelSetImportBit (ByVal lModuleNo As Long, ByVal lOffset As Long, ByVal uLevel As Long) As Long
```

Delphi

```
function AxdiLevelSetImportBit(lModuleNo : LongInt; lOffset : LongInt; uLevel : DWord) : DWord; stdcall;
```

Parameters

[in/out] Name	[Init Value] Explanation
[>>]lModuleNo	모듈 번호
[>>]lOffset	입력 접점에 대한 Offset 위치
[>>]uLevel	레벨 값 (Boolean)

Return Values

[0000] AXT_RT_SUCCESS : AXL 라이브러리 초기화 성공

[3101] AXT_RT_DIO_INVALID_MODULE_NO : 유효하지않는 DIO 모듈 번호

[3102] AXT_RT_DIO_INVALID_OFFSET_NO : 유효하지않는 DIO OFFSET 번호

* See error code Table for more information on status error codes

Description

사용자가 지정한 입력 접점 모듈의 Offset 위치에서 bit 단위로 레벨을 설정한다.

지정한 모듈이 어떤 모듈인지 확인하여 아래의 Table에서 Offset 사용 범위 참조하여 설정한다.

Offset 범위와 Value 값

Module	Offset	Level
SIO-DI32	0 ~ 31	0(LOW), 1(HIGH)
SIO-DO32P	-	-
SIO-DB32P	0 ~ 15	0(LOW), 1(HIGH)
SIO-DO32T	-	-
SIO-DB32T	0 ~ 15	0(LOW), 1(HIGH)

Example

C++

```
// 0번째 모듈의 Offset 0번지에서 bit 단위로 레벨을 설정한다.  
AxdiLevelSetImportBit(0, 0, 1);
```

Visual Basic

```
' 0번째 모듈의 Offset 0번지에서 bit 단위로 레벨을 설정한다.  
AxdiLevelSetImportBit 0, 0, 1
```

Delphi

```
{ 0번째 모듈의 Offset 0번지에서 bit 단위로 레벨을 설정한다. }  
AxdiLevelSetImportBit(0, 0, 1);
```

See Also

[AxdiLevelSetImportByte](#), [AxdiLevelSetImportWord](#), [AxdiLevelSetImportDword](#), [AxdiLevelSetImport](#)

AxdiLevelSetImportByte

Purpose

지정한 입력 접점 모듈의 Offset 위치에서 byte 단위로 레벨을 설정한다.

레벨 설정은 신호의 Active 상태를 HIGH(1)로 할지 LOW(0)로 할지 설정한다.

Format

C++

```
DWORD AxdiLevelSetImportByte(long lModuleNo, long lOffset, DWORD uLevel);
```

Visual Basic

```
Function AxdiLevelSetImportByte(ByVal lModuleNo As Long, ByVal lOffset As Long, ByVal uLevel As Long) As Long
```

Delphi

```
function AxdiLevelSetImportByte(lModuleNo : LongInt; lOffset : LongInt; uLevel : DWord) : DWord; stdcall;
```

Parameters

[in/out] Name	[Init Value] Explanation
[>>]lModuleNo	모듈 번호
[>>]lOffset	입력 접점에 대한 Offset 위치
[>>]uLevel	레벨 값 (Byte)

Return Values

[0000] AXT_RT_SUCCESS : AXL 라이브러리 초기화 성공

[3101] AXT_RT_DIO_INVALID_MODULE_NO : 유효하지않는 DIO 모듈 번호

[3102] AXT_RT_DIO_INVALID_OFFSET_NO : 유효하지않는 DIO OFFSET 번호

* See error code Table for more information on status error codes

Description

사용자가 지정한 입력 접점 모듈의 Offset 위치에서 byte 단위로 레벨을 설정한다.

지정한 모듈이 어떤 모듈인지 확인하여 아래의 Table에서 Offset 사용 범위 참조하여 설정한다.

Offset 범위와 Value 값

Module	Offset	Level
SIO-DI32	0, 1, 2, 3	00h ~ FFh
SIO-DO32P	-	-
SIO-DB32P	0, 1	00h ~ FFh
SIO-DO32T	-	-
SIO-DB32T	0, 1	00h ~ FFh

Example

C++

```
// 0번째 모듈의 Offset 0번지에서 byte 단위로 레벨을 설정한다.  
AxdiLevelSetImportByte(0, 0, 0xFF);
```

Visual Basic

```
' 0번째 모듈의 Offset 0번지에서 byte 단위로 레벨을 설정한다.  
AxdiLevelSetImportByte 0, 0, &hFF
```

Delphi

```
{ 0번째 모듈의 Offset 0번지에서 byte 단위로 레벨을 설정한다. }  
AxdiLevelSetImportByte(0, 0, $FF);
```

See Also

[AxdiLevelSetImportBit](#), [AxdiLevelSetImportWord](#), [AxdiLevelSetImportDword](#), [AxdiLevelSetImport](#)

AxdiLevelSetImportWord

Purpose

지정한 입력 접점 모듈의 Offset 위치에서 word 단위로 레벨을 설정한다.

레벨 설정은 신호의 Active 상태를 HIGH(1)로 할지 LOW(0)로 할지 설정한다.

Format

C++

```
DWORD AxdiLevelSetImportWord(long lModuleNo, long lOffset, DWORD uLevel);
```

Visual Basic

```
Function AxdiLevelSetImportWord(ByVal lModuleNo As Long, ByVal lOffset As Long, ByVal uLevel As Long) As Long
```

Delphi

```
function AxdiLevelSetImportWord(lModuleNo : LongInt; lOffset : LongInt; uLevel : DWord) : DWord; stdcall;
```

Parameters

[in/out] Name	[Init Value] Explanation
[>>]lModuleNo	모듈 번호
[>>]lOffset	입력 접점에 대한 Offset 위치
[>>]uLevel	레벨 값 (Word)

Return Values

[0000] AXT_RT_SUCCESS : AXL 라이브러리 초기화 성공

[3101] AXT_RT_DIO_INVALID_MODULE_NO : 유효하지않는 DIO 모듈 번호

[3102] AXT_RT_DIO_INVALID_OFFSET_NO : 유효하지않는 DIO OFFSET 번호

* See error code Table for more information on status error codes

Description

사용자가 지정한 입력 접점 모듈의 Offset 위치에서 word 단위로 레벨을 설정한다.

지정한 모듈이 어떤 모듈인지 확인하여 아래의 Table에서 Offset 사용 범위 참조하여 설정한다.

Offset 범위와 Value 값

Module	Offset	Level
SIO-DI32	0, 1	0000h ~ FFFFh
SIO-DO32P	-	-
SIO-DB32P	0	0000h ~ FFFFh
SIO-DO32T	-	-
SIO-DB32T	0	0990h ~ FFFFh

Example

C++

```
// 0번째 모듈의 Offset 0번지에서 word 단위로 레벨을 설정한다.  
AxdiLevelSetImportWord(0, 0, 0xFFFF);
```

Visual Basic

```
' 0번째 모듈의 Offset 0번지에서 word 단위로 레벨을 설정한다.  
AxdiLevelSetImportWord 0, 0, &hFFFF
```

Delphi

```
{ 0번째 모듈의 Offset 0번지에서 word 단위로 레벨을 설정한다. }  
AxdiLevelSetImportWord(0, 0, $FFFF);
```

See Also

[AxdiLevelSetImportBit](#), [AxdiLevelSetImportByte](#), [AxdiLevelSetImportDword](#), [AxdiLevelSetImport](#)

AxdiLevelSetImportDword

Purpose

지정한 입력 접점 모듈의 Offset 위치에서 double word 단위로 레벨을 설정한다.

레벨 설정은 신호의 Active 상태를 HIGH(1)로 할지 LOW(0)로 할지 설정한다.

Format

C++

```
DWORD AxdiLevelSetImportDword(long lModuleNo, long lOffset, DWORD uLevel);
```

Visual Basic

```
Function AxdiLevelSetImportDword(ByVal lModuleNo As Long, ByVal lOffset As Long, ByVal uLevel As Long) As Long
```

Delphi

```
function AxdiLevelSetImportDword(lModuleNo : LongInt; lOffset : LongInt; uLevel : DWord) : DWord; stdcall;
```

Parameters

[in/out] Name	[Init Value] Explanation
[>>]lModuleNo	모듈 번호
[>>]lOffset	입력 접점에 대한 Offset 위치
[>>]uLevel	레벨 값 (double word)

Return Values

[0000] AXT_RT_SUCCESS : AXL 라이브러리 초기화 성공

[3101] AXT_RT_DIO_INVALID_MODULE_NO : 유효하지않는 DIO 모듈 번호

[3102] AXT_RT_DIO_INVALID_OFFSET_NO : 유효하지않는 DIO OFFSET 번호

* See error code Table for more information on status error codes

Description

사용자가 지정한 입력 접점 모듈의 Offset 위치에서 double word 단위로 레벨을 설정한다.

지정한 모듈이 어떤 모듈인지 확인하여 아래의 Table에서 Offset 사용 범위를 참조하여 설정한다.

Offset 범위와 Value 값

Module	Offset	Level
SIO-DI32	0,	00000000h ~ FFFFFFFFh
SIO-DO32P	-	-
SIO-DB32P	0	00000000h ~ FFFFFFFFh
SIO-DO32T	-	-
SIO-DB32T	0	00000000h ~ FFFFFFFFh

Example

C++

```
// 0번째 모듈의 Offset 0번지에서 double word 단위로 레벨을 설정한다.  
AxdiLevelSetImportDword(0, 0, 0xFFFFFFFF);
```

Visual Basic

```
' 0번째 모듈의 Offset 0번지에서 double word 단위로 레벨을 설정한다.  
AxdiLevelSetImportDword 0, 0, &hFFFFFF
```

Delphi

```
{ 0번째 모듈의 Offset 0번지에서 double word 단위로 레벨을 설정한다. }  
AxdiLevelSetImportDword(0, 0, $FFFFFF);
```

See Also

[AxdiLevelSetImportBit](#), [AxdiLevelSetImportByte](#), [AxdiLevelSetImportWord](#), [AxdiLevelSetImport](#)

AxdiLevelGetImportBit

Purpose

지정한 입력 접점 모듈의 Offset 위치에서 bit 단위로 레벨을 확인한다.
레벨 확인은 Active 신호 상태가 HIGH(1) 인지 LOW(0) 인지 확인한다.

Format

C++

```
DWORD AxdiLevelGetImportBit(long lModuleNo, long lOffset, DWORD
*upLevel);
```

Visual Basic

```
Function AxdiLevelGetImportBit(ByVal lModuleNo As Long, ByVal lOffset
As Long, ByRef upLevel As Long) As Long
```

Delphi

```
function AxdiLevelGetImportBit(lModuleNo : LongInt; lOffset : LongInt;
upLevel : PWord) : Word; stdcall;
```

Parameters

[in/out] Name	[Init Value] Explanation
[>>]lModuleNo	모듈 번호
[>>]lOffset	입력 접점에 대한 Offset 위치
[<<]upLevel	레벨 값 (Boolean)

Return Values

- [0000] AXT_RT_SUCCESS : AXL 라이브러리 초기화 성공
- [3101] AXT_RT_DIO_INVALID_MODULE_NO : 유효하지않는 DIO 모듈 번호
- [3102] AXT_RT_DIO_INVALID_OFFSET_NO : 유효하지않는 DIO OFFSET 번호
- * See error code Table for more information on status error codes

Description

사용자가 지정한 입력 접점 모듈의 Offset 위치에서 bit 단위로 레벨을 확인한다.

지정한 모듈이 어떤 모듈인지 확인하여 아래의 Table에서 Offset 사용 범위 참조하여 확인한다.

Offset 범위와 Value 값

Module	Offset	Level
SIO-DI32	0 ~ 31	0(LOW), 1(HIGH)
SIO-DO32P	-	-
SIO-DB32P	0 ~ 15	0(LOW), 1(HIGH)
SIO-DO32T	-	-
SIO-DB32T	0 ~ 15	0(LOW), 1(HIGH)

Example

C++

```
// 0번째 모듈의 Offset 0번지에서 bit 단위로 레벨을 확인한다.  
DWORD dwLevel;  
  
AxdiLevelGetImportBit(0, 0, &dwLevel);
```

Visual Basic

```
' 0번째 모듈의 Offset 0번지에서 bit 단위로 레벨을 확인한다.  
Dim lLevel As long  
  
AxdiLevelGetImportBit 0, 0, lLevel
```

Delphi

```
{ 0번째 모듈의 Offset 0번지에서 bit 단위로 레벨을 확인한다. }  
var  
  dwLevel : DWord;  
  
begin  
  AxdiLevelGetImportBit(0, 0, @dwLevel);  
end;
```

See Also

[AxdiLevelGetImportByte](#), [AxdiLevelGetImportWord](#), [AxdiLevelGetImportDword](#), [AxdiLevelGetImport](#)

AxdiLevelGetImportByte

Purpose

지정한 입력 접점 모듈의 Offset 위치에서 byte 단위로 레벨을 확인한다.

레벨 확인은 Active 신호 상태가 HIGH(1) 인지 LOW(0) 인지 확인한다.

Format

C++

```
DWORD AxdiLevelGetImportByte(long lModuleNo, long lOffset, DWORD
    *upLevel);
```

Visual Basic

```
Function AxdiLevelGetImportByte(ByVal lModuleNo As Long, ByVal
    lOffset As Long, ByRef upLevel As Long) As Long
```

Delphi

```
function AxdiLevelGetImportByte(lModuleNo : LongInt; lOffset :
    LongInt; upLevel : PDWord) : DWord; stdcall;
```

Parameters

[in/out] Name	[Init Value] Explanation
[>>]lModuleNo	모듈 번호
[>>]lOffset	입력 접점에 대한 Offset 위치
[<<]upLevel	레벨 값 (Byte)

Return Values

[0000] AXT_RT_SUCCESS : AXL 라이브러리 초기화 성공

[3101] AXT_RT_DIO_INVALID_MODULE_NO : 유효하지않는 DIO 모듈 번호

[3102] AXT_RT_DIO_INVALID_OFFSET_NO : 유효하지않는 DIO OFFSET 번호

* See error code Table for more information on status error codes

Description

사용자가 지정한 입력 접점 모듈의 Offset 위치에서 byte 단위로 레벨을 확인한다.

지정한 모듈이 어떤 모듈인지 확인하여 아래의 Table에서 Offset 사용 범위를 참조하여 확인한다.

Offset 범위와 Value 값

Module	Offset	Level
SIO-DI32	0, 1, 2, 3	00h ~ FFh
SIO-DO32P	-	-
SIO-DB32P	0, 1	00h ~ FFh
SIO-DO32T	-	-
SIO-DB32T	0, 1	00h ~ FFh

Example

C++

```
// 0번째 모듈의 Offset 0번지에서 byte 단위로 레벨을 확인한다.  
DWORD dwLevel;  
  
AxdiLevelGetImportByte(0, 0, &dwLevel);
```

Visual Basic

```
' 0번째 모듈의 Offset 0번지에서 byte 단위로 레벨을 확인한다.  
Dim lLevel As long  
  
AxdiLevelGetImportByte 0, 0, lLevel
```

Delphi

```
{ 0번째 모듈의 Offset 0번지에서 byte 단위로 레벨을 확인한다. }  
var  
  dwLevel : DWord;  
  
begin  
  AxdiLevelGetImportByte(0, 0, @dwLevel);  
end;
```

See Also

[AxdiLevelGetImportBit](#), [AxdiLevelGetImportWord](#), [AxdiLevelGetImportDword](#), [AxdiLevelGetImport](#)

AxdiLevelGetImportWord

Purpose

지정한 입력 접점 모듈의 Offset 위치에서 word 단위로 레벨을 확인한다.

레벨 확인은 Active 신호 상태가 HIGH(1) 인지 LOW(0) 인지 확인한다.

Format

C++

```
DWORD AxdiLevelGetImportWord(long lModuleNo, long lOffset, DWORD
    *upLevel);
```

Visual Basic

```
Function AxdiLevelGetImportWord(ByVal lModuleNo As Long, ByVal
    lOffset As Long, ByRef upLevel As Long) As Long
```

Delphi

```
function AxdiLevelGetImportWord(lModuleNo : LongInt; lOffset :
    LongInt; upLevel : PDWord) : DWord; stdcall;
```

Parameters

[in/out] Name	[Init Value] Explanation
[>>]lModuleNo	모듈 번호
[>>]lOffset	입력 접점에 대한 Offset 위치
[<<]upLevel	레벨 값 (Word)

Return Values

[0000] AXT_RT_SUCCESS : AXL 라이브러리 초기화 성공

[3101] AXT_RT_DIO_INVALID_MODULE_NO : 유효하지않는 DIO 모듈 번호

[3102] AXT_RT_DIO_INVALID_OFFSET_NO : 유효하지않는 DIO OFFSET 번호

* See error code Table for more information on status error codes

Description

사용자가 지정한 입력 접점 모듈의 Offset 위치에서 word 단위로 레벨을 확인한다.

지정한 모듈이 어떤 모듈인지 확인하여 아래의 Table에서 Offset 사용 범위 참조하여 확인한다.

Offset 범위와 Value 값

Module	Offset	Level
SIO-DI32	0, 1	0000h ~ FFFFh
SIO-DO32P	-	-
SIO-DB32P	0	0000h ~ FFFFh
SIO-DO32T	-	-
SIO-DB32T	0	0990h ~ FFFFh

Example

C++

```
// 0번째 모듈의 Offset 0번지에서 word 단위로 레벨을 확인한다.  
DWORD dwLevel;  
  
AxdiLevelGetImportWord(0, 0, &dwLevel);
```

Visual Basic

```
' 0번째 모듈의 Offset 0번지에서 word 단위로 레벨을 확인한다.  
Dim lLevel As long  
  
AxdiLevelGetImportWord 0, 0, lLevel
```

Delphi

```
{ 0번째 모듈의 Offset 0번지에서 word 단위로 레벨을 확인한다. }  
var  
  dwLevel : DWord;  
  
begin  
  AxdiLevelGetImportWord(0, 0, @dwLevel);  
end;
```

See Also

[AxdiLevelGetImportBit](#), [AxdiLevelGetImportByte](#), [AxdiLevelGetImportDword](#), [AxdiLevelGetImport](#)

AxdiLevelGetImportDword

Purpose

지정한 입력 접점 모듈의 Offset 위치에서 double word 단위로 레벨을 확인한다.

레벨 확인은 Active 신호 상태가 HIGH(1) 인지 LOW(0) 인지 확인한다.

Format

C++

```
DWORD AxdiLevelGetImportDword(long lModuleNo, long lOffset, DWORD *upLevel);
```

Visual Basic

```
Function AxdiLevelGetImportDword(ByVal lModuleNo As Long, ByVal lOffset As Long, ByRef upLevel As Long) As Long
```

Delphi

```
function AxdiLevelGetImportDword(lModuleNo : LongInt; lOffset : LongInt; upLevel : PDWord) : DWord; stdcall;
```

Parameters

[in/out] Name	[Init Value] Explanation
[>>]lModuleNo	모듈 번호
[>>]lOffset	입력 접점에 대한 Offset 위치
[<<]upLevel	레벨 값 (Double Word)

Return Values

[0000] AXT_RT_SUCCESS : AXL 라이브러리 초기화 성공

[3101] AXT_RT_DIO_INVALID_MODULE_NO : 유효하지않는 DIO 모듈 번호

[3102] AXT_RT_DIO_INVALID_OFFSET_NO : 유효하지않는 DIO OFFSET 번호

* See error code Table for more information on status error codes

Description

사용자가 지정한 입력 접점 모듈의 Offset 위치에서 double word 단위로 레벨을 확인한다.

지정한 모듈이 어떤 모듈인지 확인하여 아래의 Table에서 Offset 사용 범위 참조하여 확인한다.

Offset 범위와 Value 값

Module	Offset	Level
SIO-DI32	0,	00000000h ~ FFFFFFFFh
SIO-DO32P	-	-
SIO-DB32P	0	00000000h ~ FFFFFFFFh
SIO-DO32T	-	-
SIO-DB32T	0	00000000h ~ FFFFFFFFh

Example

C++

```
// 0번째 모듈의 Offset 0번지에서 double word 단위로 레벨을 확인한다.  
DWORD dwLevel;  
  
AxdiLevelGetImportDword(0, 0, &dwLevel);
```

Visual Basic

```
' 0번째 모듈의 Offset 0번지에서 double word 단위로 레벨을 확인한다.  
Dim lLevel As long  
  
AxdiLevelGetImportDword 0, 0, lLevel
```

Delphi

```
{ 0번째 모듈의 Offset 0번지에서 double word 단위로 레벨을 확인한다. }  
var  
  dwLevel : DWord;  
  
begin  
  AxdiLevelGetImportDword(0, 0, @dwLevel);  
end;
```

See Also

[AxdiLevelGetImportBit](#), [AxdiLevelGetImportByte](#), [AxdiLevelGetImportWord](#), [AxdiLevelGetImport](#)

AxdiLevelSetInport

Purpose

전체 입력 접점 모듈의 Offset 위치에서 bit 단위로 레벨을 설정한다.

레벨 설정은 신호의 Active 상태를 HIGH(1)로 할지 LOW(0)로 할지 설정한다.

Format

C++

```
DWORD AxdiLevelSetInport(long lOffset, DWORD uLevel);
```

Visual Basic

```
Function AxdiLevelSetInport(ByVal lOffset As Long, ByVal uLevel As Long) As Long
```

Delphi

```
function AxdiLevelSetInport(lOffset : LongInt; uLevel : DWord) : DWord; stdcall;
```

Parameters

[in/out] Name	[Init Value] Explanation
[>>]lOffset	전체 입력 접점에 대한 Offset 위치
[>>]uLevel	레벨 값 (Boolean)

Return Values

[0000] AXT_RT_SUCCESS : AXL 라이브러리 초기화 성공

[3051] AXT_RT_DIO_NOT_MODULE : DIO 모듈 없음

[3102] AXT_RT_DIO_INVALID_OFFSET_NO : 유효하지않는 DIO OFFSET 번호

* See error code Table for more information on status error codes

Description

전체 입력 접점 모듈의 Offset 위치에서 bit 단위로 레벨을 설정한다.

Offset 은 '0'부터 총 입력 접점 수 - 1 까지 사용할 수 있다.

Example

C++

```
// 전체 모듈, Offset 0번지에서 bit 단위로 레벨을 설정한다.  
AxdiLevelSetInport(0, 1);
```

Visual Basic

```
' 전체 모듈, Offset 0번지에서 bit 단위로 레벨을 설정한다.  
AxdiLevelSetInport 0, 1
```

Delphi

```
{ 전체 모듈, Offset 0번지에서 bit 단위로 레벨을 설정한다. }
AxdiLevelSetImport(0, 1);
```

See Also

[AxdiLevelSetImportBit](#), [AxdiLevelSetImportByte](#), [AxdiLevelSetImportWord](#), [AxdiLevelSetImportDword](#),
[AxdiLevelGetImport](#)

AxdiLevelGetInport

Purpose

전체 입력 접점 모듈의 Offset 위치에서 bit 단위로 레벨을 확인한다.

레벨 확인은 Active 신호 상태가 HIGH(1) 인지 LOW(0) 인지 확인한다.

Format

C++

```
DWORD AxdiLevelGetInport(long lOffset, DWORD *upLevel);
```

Visual Basic

```
Function AxdiLevelGetInport(ByVal lOffset As Long, ByRef upLevel As Long) As Long
```

Delphi

```
function AxdiLevelGetInport(lOffset : LongInt; upLevel : PDWord) : DWord; stdcall;
```

Parameters

[in/out] Name	[Init Value] Explanation
[>>]lOffset	전체 입력 접점에 대한 Offset 위치
[<<]upLevel	레벨 값 (Boolean)

Return Values

[0000] AXT_RT_SUCCESS : AXL 라이브러리 초기화 성공

[3101] AXT_RT_DIO_INVALID_MODULE_NO : 유효하지않는 DIO 모듈 번호

[3102] AXT_RT_DIO_INVALID_OFFSET_NO : 유효하지않는 DIO OFFSET 번호

* See error code Table for more information on status error codes

Description

전체 입력 접점 모듈의 Offset 위치에서 bit 단위로 레벨을 확인한다.

Offset 은 '0'부터 총 입력 접점 수 - 1 까지 사용할 수 있다.

Example

C++

```
// 전체 모듈, Offset 0번지에서 bit 단위로 레벨을 확인한다.  
DWORD dwLevel;  
  
AxdiLevelGetInport(0, &dwLevel);
```

Visual Basic

' 전체 모듈, Offset 0번지에서 bit 단위로 레벨을 확인한다.

```
Dim lLevel As long
```

```
AxdiLevelGetInport 0, lLevel
```

Delphi

```
{ 전체 모듈, Offset 0번지에서 bit 단위로 레벨을 확인한다. }
var
    dwLevel : DWord;

begin
    AxdiLevelGetImportBit(0, @dwLevel);
end
```

See Also

[AxdiLevelGetImportBit](#), [AxdiLevelGetImportByte](#), [AxdiLevelGetImportWord](#), [AxdiLevelGetImportDword](#),
[AxdiLevelSetImport](#)

출력 레벨 설정 확인

Function	Description
Axd0LevelSetOutportBit	지정한 출력 접점 모듈의 Offset 위치에서 bit 단위로 레벨을 설정한다.
Axd0LevelSetOutportByte	지정한 출력 접점 모듈의 Offset 위치에서 byte 단위로 레벨을 설정한다.
Axd0LevelSetOutportWord	지정한 출력 접점 모듈의 Offset 위치에서 word 단위로 레벨을 설정한다.
Axd0LevelSetOutportDword	지정한 출력 접점 모듈의 Offset 위치에서 double word 단위로 레벨을 설정한다.
Axd0LevelGetOutportBit	지정한 출력 접점 모듈의 Offset 위치에서 bit 단위로 레벨을 확인한다.
Axd0LevelGetOutportByte	지정한 출력 접점 모듈의 Offset 위치에서 byte 단위로 레벨을 확인한다.
Axd0LevelGetOutportWord	지정한 출력 접점 모듈의 Offset 위치에서 word 단위로 레벨을 확인한다.
Axd0LevelGetOutportDword	지정한 출력 접점 모듈의 Offset 위치에서 double word 단위로 레벨을 확인한다.
Axd0LevelSetOutport	전체 출력 접점 모듈의 Offset 위치에서 bit 단위로 레벨을 설정한다.
Axd0LevelGetOutport	전체 출력 접점 모듈의 Offset 위치에서 bit 단위로 레벨을 확인한다.

AxdoLevelSetOutportBit

Purpose

지정한 출력 접점 모듈의 Offset 위치에서 bit 단위로 레벨을 설정한다.

레벨 설정은 신호의 Active 상태를 HIGH(1)로 할지 LOW(0)로 할지 설정한다.

Format

C++

```
DWORD AxdoLevelSetOutportBit(long lModuleNo, long lOffset, DWORD uLevel);
```

Visual Basic

```
Function AxdoLevelSetOutportBit(ByVal lModuleNo As Long, ByVal lOffset As Long, ByVal uLevel As Long) As Long
```

Delphi

```
function AxdoLevelSetOutportBit(lModuleNo : LongInt; lOffset : LongInt; uLevel : DWord) : DWord; stdcall;
```

Parameters

[in/out] Name	[Init Value] Explanation
[>>]lModuleNo	모듈 번호
[>>]lOffset	출력 접점에 대한 Offset 위치
[>>]uLevel	레벨 값 (Boolean)

Return Values

[0000] AXT_RT_SUCCESS : AXL 라이브러리 초기화 성공

[3101] AXT_RT_DIO_INVALID_MODULE_NO : 유효하지않는 DIO 모듈 번호

[3102] AXT_RT_DIO_INVALID_OFFSET_NO : 유효하지않는 DIO OFFSET 번호

* See error code Table for more information on status error codes

Description

사용자가 지정한 출력 접점 모듈의 Offset 위치에서 bit 단위로 레벨을 설정한다.

지정한 모듈이 어떤 모듈인지 확인하여 아래의 Table에서 Offset 사용 범위를 참조하여 설정한다.

Offset 범위와 Value 값

Module	Offset	Level
SIO-DI32	-	-
SIO-DO32P	0 ~ 31	0(LOW), 1(HIGH)
SIO-DB32P	0 ~ 15	0(LOW), 1(HIGH)
SIO-DO32T	0 ~ 31	0(LOW), 1(HIGH)
SIO-DB32T	0 ~ 15	0(LOW), 1(HIGH)

Example

C++

```
// 0번째 모듈의 Offset 0번지에서 bit 단위로 데이터 레벨을 설정한다.  
AxdoLevelSetOutportBit(0, 0, 1);
```

Visual Basic

```
' 0번째 모듈의 Offset 0번지에서 bit 단위로 데이터 레벨을 설정한다.  
AxdoLevelSetOutportBit 0, 0, 1
```

Delphi

```
{ 0번째 모듈의 Offset 0번지에서 bit 단위로 데이터 레벨을 설정한다. }  
AxdoLevelSetOutportBit(0, 0, 1);
```

See Also

[AxdoLevelSetOutportByte](#), [AxdoLevelSetOutportWord](#), [AxdoLevelSetOutportDword](#),
[AxdoLevelSetOutport](#)

AxdoLevelSetOutportByte

Purpose

지정한 출력 접점 모듈의 Offset 위치에서 byte 단위로 레벨을 설정한다.

레벨 설정은 신호의 Active 상태를 HIGH(1)로 할지 LOW(0)로 할지 설정한다.

Format

C++

```
DWORD AxdoLevelSetOutportByte(long lModuleNo, long lOffset, DWORD uLevel);
```

Visual Basic

```
Function AxdoLevelSetOutportByte(ByVal lModuleNo As Long, ByVal lOffset As Long, ByVal uLevel As Long) As Long
```

Delphi

```
function AxdoLevelSetOutportByte(lModuleNo : LongInt; lOffset : LongInt; uLevel : DWord) : DWord; stdcall;
```

Parameters

[in/out] Name	[Init Value] Explanation
[>>]lModuleNo	모듈 번호
[>>]lOffset	출력 접점에 대한 Offset 위치
[>>]uLevel	레벨 값 (Byte)

Return Values

[0000] AXT_RT_SUCCESS : AXL 라이브러리 초기화 성공

[3101] AXT_RT_DIO_INVALID_MODULE_NO : 유효하지않는 DIO 모듈 번호

[3102] AXT_RT_DIO_INVALID_OFFSET_NO : 유효하지않는 DIO OFFSET 번호

* See error code Table for more information on status error codes

Description

사용자가 지정한 출력 접점 모듈의 Offset 위치에서 byte 단위로 레벨을 설정한다.

지정한 모듈이 어떤 모듈인지 확인하여 아래의 Table에서 Offset 사용 범위를 참조하여 설정한다.

Offset 범위와 Value 값

Module	Offset	Level
SIO-DI32	-	-
SIO-DO32P	0, 1, 2, 3	00h ~ FFh
SIO-DB32P	0, 1	00h ~ FFh
SIO-DO32T	0, 1, 2, 3	00h ~ FFh
SIO-DB32T	0, 1	00h ~ FFh

Example

C++

```
// 0번째 모듈의 Offset 0번지에서 byte 단위로 레벨을 설정한다.  
AxdoLevelSetOutportByte(0, 0, 0xFF);
```

Visual Basic

```
' 0번째 모듈의 Offset 0번지에서 byte 단위로 레벨을 설정한다.  
AxdoLevelSetOutportByte 0, 0, &hFF
```

Delphi

```
{ 0번째 모듈의 Offset 0번지에서 byte 단위로 레벨을 설정한다. }  
AxdoLevelSetOutportByte(0, 0, $FF);
```

See Also

[AxdoLevelSetOutportBit](#), [AxdoLevelSetOutportWord](#), [AxdoLevelSetOutportDword](#),
[AxdoLevelSetOutport](#)

AxdoLevelSetOutportWord

Purpose

지정한 출력 접점 모듈의 Offset 위치에서 word 단위로 레벨을 설정한다.

레벨 설정은 신호의 Active 상태를 HIGH(1)로 할지 LOW(0)로 할지 설정한다.

Format

C++

```
DWORD AxdoLevelSetOutportWord(long lModuleNo, long lOffset, DWORD uLevel);
```

Visual Basic

```
Function AxdoLevelSetOutportWord(ByVal lModuleNo As Long, ByVal lOffset As Long, ByVal uLevel As Long) As Long
```

Delphi

```
function AxdoLevelSetOutportWord(lModuleNo : LongInt; lOffset : LongInt; uLevel : DWord) : DWord; stdcall;
```

Parameters

[in/out] Name	[Init Value] Explanation
[>>]lModuleNo	모듈 번호
[>>]lOffset	출력 접점에 대한 Offset 위치
[>>]uLevel	레벨 값 (Word)

Return Values

[0000] AXT_RT_SUCCESS : AXL 라이브러리 초기화 성공

[3101] AXT_RT_DIO_INVALID_MODULE_NO : 유효하지않는 DIO 모듈 번호

[3102] AXT_RT_DIO_INVALID_OFFSET_NO : 유효하지않는 DIO OFFSET 번호

* See error code Table for more information on status error codes

Description

사용자가 지정한 입력 접점 모듈의 Offset 위치에서 word 단위로 레벨을 설정한다.

지정한 모듈이 어떤 모듈인지 확인하여 아래의 Table에서 Offset 사용 범위를 참조하여 설정한다.

Offset 범위와 Value 값

Module	Offset	Level
SIO-DI32	-	-
SIO-DO32P	0, 1	0000h ~ FFFFh
SIO-DB32P	0	0000h ~ FFFFh
SIO-DO32T	0, 1	0000h ~ FFFFh
SIO-DB32T	0	0990h ~ FFFFh

Example

C++

```
// 0번째 모듈의 Offset 0번지에서 word 단위로 레벨을 설정한다.  
AxdoLevelSetOutportWord(0, 0, 0xFFFF);
```

Visual Basic

```
' 0번째 모듈의 Offset 0번지에서 word 단위로 레벨을 설정한다.  
AxdoLevelSetOutportWord 0, 0, &hFFFF
```

Delphi

```
{ 0번째 모듈의 Offset 0번지에서 word 단위로 레벨을 설정한다. }  
AxdoLevelSetOutportWord(0, 0, $FFFF);
```

See Also

[AxdoLevelSetOutportBit](#), [AxdoLevelSetOutportByte](#), [AxdoLevelSetOutportDword](#),
[AxdoLevelSetOutport](#)

AxdoLevelSetOutportDword

Purpose

지정한 출력 접점 모듈의 Offset 위치에서 double word 단위로 레벨을 설정한다.

레벨 설정은 신호의 Active 상태를 HIGH(1)로 할지 LOW(0)로 할지 설정한다.

Format

C++

```
DWORD AxdoLevelSetOutportDword(long lModuleNo, long lOffset, DWORD uLevel);
```

Visual Basic

```
Function AxdoLevelSetOutportDword(ByVal lModuleNo As Long, ByVal lOffset As Long, ByVal uLevel As Long) As Long
```

Delphi

```
function AxdoLevelSetOutportDword(lModuleNo : LongInt; lOffset : LongInt; uLevel : DWord) : DWord; stdcall;
```

Parameters

[in/out] Name	[Init Value] Explanation
[>>]lModuleNo	모듈 번호
[>>]lOffset	출력 접점에 대한 Offset 위치
[>>]uLevel	레벨 값 (double word)

Return Values

[0000] AXT_RT_SUCCESS : AXL 라이브러리 초기화 성공

[3101] AXT_RT_DIO_INVALID_MODULE_NO : 유효하지않는 DIO 모듈 번호

[3102] AXT_RT_DIO_INVALID_OFFSET_NO : 유효하지않는 DIO OFFSET 번호

* See error code Table for more information on status error codes

Description

사용자가 지정한 출력 접점 모듈의 Offset 위치에서 double word 단위로 레벨을 설정한다.

지정한 모듈이 어떤 모듈인지 확인하여 아래의 Table에서 Offset 사용 범위를 참조하여 설정한다.

Offset 범위와 Value 값

Module	Offset	Level
SIO-DI32	-	-
SIO-DO32P	0	00000000h ~ FFFFFFFFh
SIO-DB32P	0	00000000h ~ FFFFFFFFh
SIO-DO32T	0	00000000h ~ FFFFFFFFh
SIO-DB32T	0	00000000h ~ FFFFFFFFh

Example

C++

```
// 0번째 모듈의 Offset 0번지에서 double word 단위로 레벨을 설정한다.  
AxdoLevelSetOutportDword(0, 0, 0xFFFFFFFF);
```

Visual Basic

```
' 0번째 모듈의 Offset 0번지에서 double word 단위로 레벨을 설정한다.  
AxdoLevelSetOutportDword 0, 0, &hFFFFFF
```

Delphi

```
{ 0번째 모듈의 Offset 0번지에서 double word 단위로 레벨을 설정한다. }  
AxdoLevelSetOutportDword(0, 0, $FFFFFF);
```

See Also

[AxdoLevelSetOutportBit](#), [AxdoLevelSetOutportByte](#), [AxdoLevelSetOutportWord](#),
[AxdoLevelSetOutport](#)

AxdoLevelGetOutportBit

Purpose

지정한 출력 접점 모듈의 Offset 위치에서 bit 단위로 레벨을 확인한다.
레벨 확인은 Active 신호 상태가 HIGH(1) 인지 LOW(0) 인지 확인한다.

Format

C++

```
DWORD AxdoLevelGetOutportBit(long lModuleNo, long lOffset, DWORD
*upLevel);
```

Visual Basic

```
Function AxdoLevelGetOutportBit(ByVal lModuleNo As Long, ByVal
lOffset As Long, ByRef upLevel As Long) As Long
```

Delphi

```
function AxdoLevelGetOutportBit(lModuleNo : LongInt; lOffset :
LongInt; upLevel : PDWord) : DWord; stdcall;
```

Parameters

[in/out] Name	[Init Value] Explanation
[>>]lModuleNo	모듈 번호
[>>]lOffset	출력 접점에 대한 Offset 위치
[<<]upLevel	레벨 값 (Boolean)

Return Values

- [0000] AXT_RT_SUCCESS : AXL 라이브러리 초기화 성공
- [3101] AXT_RT_DIO_INVALID_MODULE_NO : 유효하지않는 DIO 모듈 번호
- [3102] AXT_RT_DIO_INVALID_OFFSET_NO : 유효하지않는 DIO OFFSET 번호
- * See error code Table for more information on status error codes

Description

사용자가 지정한 출력 접점 모듈의 Offset 위치에서 bit 단위로 레벨을 확인한다.

지정한 모듈이 어떤 모듈인지 확인하여 아래의 Table에서 Offset 사용 범위를 참조하여 확인한다.

Offset 범위와 Value 값

Module	Offset	Level
SIO-DI32	-	-
SIO-DO32P	0 ~ 31	0(LOW), 1(HIGH)
SIO-DB32P	0 ~ 15	0(LOW), 1(HIGH)
SIO-DO32T	0 ~ 31	0(LOW), 1(HIGH)
SIO-DB32T	0 ~ 15	0(LOW), 1(HIGH)

Example

C++

```
// 0번째 모듈의 Offset 0번지에서 bit 단위로 레벨을 확인한다.  
DWORD dwLevel;  
  
AxdoLevelGetOutportBit(0, 0, &dwLevel);
```

Visual Basic

```
' 0번째 모듈의 Offset 0번지에서 bit 단위로 레벨을 확인한다.  
Dim lLevel As long  
  
AxdoLevelGetOutportBit 0, 0, lLevel
```

Delphi

```
{ 0번째 모듈의 Offset 0번지에서 bit 단위로 레벨을 확인한다. }  
var  
  dwLevel : DWord;  
  
begin  
  AxdoLevelGetOutportBit(0, 0, @dwLevel);  
end;
```

See Also

[AxdoLevelGetOutportByte](#), [AxdoLevelGetOutportWord](#), [AxdoLevelGetOutportDword](#),
[AxdoLevelGetOutport](#)

AxdoLevelGetOutportByte

Purpose

지정한 출력 접점 모듈의 Offset 위치에서 byte 단위로 레벨을 확인한다.
레벨 확인은 Active 신호 상태가 HIGH(1) 인지 LOW(0) 인지 확인한다.

Format

C++

```
DWORD AxdoLevelGetOutportByte(long lModuleNo, long lOffset, DWORD *upLevel);
```

Visual Basic

```
Function AxdoLevelGetOutportByte(ByVal lModuleNo As Long, ByVal lOffset As Long, ByRef upLevel As Long) As Long
```

Delphi

```
function AxdoLevelGetOutportByte(lModuleNo : LongInt; lOffset : LongInt; upLevel : PDWord) : DWord; stdcall;
```

Parameters

[in/out] Name	[Init Value] Explanation
[>>]lModuleNo	모듈 번호
[>>]lOffset	출력 접점에 대한 Offset 위치
[<<]upLevel	레벨 값 (Byte)

Return Values

- [0000] AXT_RT_SUCCESS : AXL 라이브러리 초기화 성공
- [3101] AXT_RT_DIO_INVALID_MODULE_NO : 유효하지않는 DIO 모듈 번호
- [3102] AXT_RT_DIO_INVALID_OFFSET_NO : 유효하지않는 DIO OFFSET 번호
- * See error code Table for more information on status error codes

Description

사용자가 지정한 출력 접점 모듈의 Offset 위치에서 byte 단위로 레벨을 확인한다.

지정한 모듈이 어떤 모듈인지 확인하여 아래의 Table에서 Offset 사용 범위를 참조하여 확인한다.

Offset 범위와 Value 값

Module	Offset	Level
SIO-DI32	-	-
SIO-DO32P	0, 1, 2, 3	00h ~ FFh
SIO-DB32P	0, 1	00h ~ FFh
SIO-DO32T	0, 1, 2, 3	00h ~ FFh
SIO-DB32T	0, 1	00h ~ FFh

Example

C++

```
// 0번째 모듈의 Offset 0번지에서 byte 단위로 레벨을 확인한다.  
DWORD dwLevel;  
  
AxdoLevelGetOutportByte(0, 0, &dwLevel);
```

Visual Basic

```
' 0번째 모듈의 Offset 0번지에서 byte 단위로 레벨을 확인한다.  
Dim lLevel As long  
  
AxdoLevelGetOutportByte 0, 0, lLevel
```

Delphi

```
{ 0번째 모듈의 Offset 0번지에서 byte 단위로 레벨을 확인한다. }  
var  
  dwLevel : DWord;  
  
begin  
  AxdoLevelGetOutportByte(0, 0, @dwLevel);  
end;
```

See Also

[AxdoLevelGetOutportBit](#), [AxdoLevelGetOutportWord](#), [AxdoLevelGetOutportDword](#),
[AxdoLevelGetOutport](#)

AxdoLevelGetOutportWord

Purpose

지정한 출력 접점 모듈의 Offset 위치에서 word 단위로 레벨을 확인한다.

레벨 확인은 Active 신호 상태가 HIGH(1) 인지 LOW(0) 인지 확인한다.

Format

C++

```
DWORD AxdoLevelGetOutportWord(long lModuleNo, long lOffset, DWORD *upLevel);
```

Visual Basic

```
Function AxdoLevelGetOutportWord(ByVal lModuleNo As Long, ByVal lOffset As Long, ByRef upLevel As Long) As Long
```

Delphi

```
function AxdoLevelGetOutportWord(lModuleNo : LongInt; lOffset : LongInt; upLevel : PDWord) : DWord; stdcall;
```

Parameters

[in/out] Name	[Init Value] Explanation
[>>]lModuleNo	모듈 번호
[>>]lOffset	출력 접점에 대한 Offset 위치
[<<]upLevel	레벨 값 (Word)

Return Values

[0000] AXT_RT_SUCCESS : AXL 라이브러리 초기화 성공

[3101] AXT_RT_DIO_INVALID_MODULE_NO : 유효하지않는 DIO 모듈 번호

[3102] AXT_RT_DIO_INVALID_OFFSET_NO : 유효하지않는 DIO OFFSET 번호

* See error code Table for more information on status error codes

Description

사용자가 지정한 출력 접점 모듈의 Offset 위치에서 word 단위로 레벨을 확인한다.

지정한 모듈이 어떤 모듈인지 확인하여 아래의 Table에서 Offset 사용 범위를 참조하여 확인한다.

Offset 범위와 Value 값

Module	Offset	Level
SIO-DI32	-	-
SIO-DO32P	0, 1	0000h ~ FFFFh
SIO-DB32P	0	0000h ~ FFFFh
SIO-DO32T	0, 1	0000h ~ FFFFh
SIO-DB32T	0	0990h ~ FFFFh

Example

C++

```
// 0번째 모듈의 Offset 0번지에서 word 단위로 레벨을 확인한다.  
DWORD dwLevel;  
  
AxdoLevelGetOutportWord(0, 0, &dwLevel);
```

Visual Basic

```
' 0번째 모듈의 Offset 0번지에서 word 단위로 레벨을 확인한다.  
Dim lLevel As long  
  
AxdoLevelGetOutportWord 0, 0, lLevel
```

Delphi

```
{ 0번째 모듈의 Offset 0번지에서 word 단위로 레벨을 확인한다. }  
var  
  dwLevel : DWord;  
  
begin  
  AxdoLevelGetOutportWord(0, 0, @dwLevel);  
end;
```

See Also

[AxdoLevelGetOutportBit](#), [AxdoLevelGetOutportByte](#), [AxdoLevelGetOutportDword](#),
[AxdoLevelGetOutport](#)

AxdoLevelGetOutportDword

Purpose

지정한 출력 접점 모듈의 Offset 위치에서 double word 단위로 레벨을 확인한다.

레벨 확인은 Active 신호 상태가 HIGH(1) 인지 LOW(0) 인지 확인한다.

Format

C++

```
DWORD AxdoLevelGetOutportDword(long lModuleNo, long lOffset, DWORD *upLevel);
```

Visual Basic

```
Function AxdoLevelGetOutportDword(ByVal lModuleNo As Long, ByVal lOffset As Long, ByRef upLevel As Long) As Long
```

Delphi

```
function AxdoLevelGetOutportDword(lModuleNo : LongInt; lOffset : LongInt; upLevel : PDWord) : DWord; stdcall;
```

Parameters

[in/out] Name	[Init Value] Explanation
[>>]lModuleNo	모듈 번호
[>>]lOffset	출력 접점에 대한 Offset 위치
[<<]upLevel	레벨 값 (Double Word)

Return Values

[0000] AXT_RT_SUCCESS : AXL 라이브러리 초기화 성공

[3101] AXT_RT_DIO_INVALID_MODULE_NO : 유효하지않는 DIO 모듈 번호

[3102] AXT_RT_DIO_INVALID_OFFSET_NO : 유효하지않는 DIO OFFSET 번호

* See error code Table for more information on status error codes

Description

사용자가 지정한 출력 접점 모듈의 Offset 위치에서 double word 단위로 레벨을 확인한다.

지정한 모듈이 어떤 모듈인지 확인하여 아래의 Table에서 Offset 사용 범위를 참조하여 확인한다.

Offset 범위와 Value 값

Module	Offset	Level
SIO-DI32	-	-
SIO-DO32P	0	00000000h ~ FFFFFFFFh
SIO-DB32P	0	00000000h ~ FFFFFFFFh
SIO-DO32T	0	00000000h ~ FFFFFFFFh
SIO-DB32T	0	00000000h ~ FFFFFFFFh

Example

C++

```
// 0번째 모듈의 Offset 0번지에서 double word 단위로 레벨을 확인한다.  
DWORD dwLevel;  
  
AxdoLevelGetOutportDword(0, 0, &dwLevel);
```

Visual Basic

```
' 0번째 모듈의 Offset 0번지에서 double word 단위로 레벨을 확인한다.  
Dim lLevel As long  
  
AxdoLevelGetOutportDword 0, 0, lLevel
```

Delphi

```
{ 0번째 모듈의 Offset 0번지에서 double word 단위로 레벨을 확인한다. }  
var  
  dwLevel : DWord;  
  
begin  
  AxdoLevelGetOutportDword(0, 0, @dwLevel);  
end;
```

See Also

[AxdoLevelGetOutportBit](#), [AxdoLevelGetOutportByte](#), [AxdoLevelGetOutportWord](#),
[AxdoLevelGetOutport](#)

AxdoLevelSetOutport

Purpose

전체 출력 접점 모듈의 Offset 위치에서 bit 단위로 레벨을 설정한다.

레벨 설정은 신호의 Active 상태를 HIGH(1)로 할지 LOW(0)로 할지 설정한다.

Format

C++

```
DWORD AxdoLevelSetOutport(long lOffset, DWORD uLevel);
```

Visual Basic

```
Function AxdoLevelSetOutport(ByVal lOffset As Long, ByVal uLevel As Long) As Long
```

Delphi

```
function AxdoLevelSetOutport(lOffset : LongInt; uLevel : DWord) : DWord; stdcall;
```

Parameters

[in/out] Name	[Init Value] Explanation
[>>]lOffset	전체 출력 접점에 대한 Offset 위치
[>>]uLevel	레벨 값 (Boolean)

Return Values

[0000] AXT_RT_SUCCESS : AXL 라이브러리 초기화 성공

[3101] AXT_RT_DIO_INVALID_MODULE_NO : 유효하지않는 DIO 모듈 번호

[3102] AXT_RT_DIO_INVALID_OFFSET_NO : 유효하지않는 DIO OFFSET 번호

* See error code Table for more information on status error codes

Description

전체 출력 접점 모듈의 Offset 위치에서 bit 단위로 레벨을 설정한다.

Offset 은 '0'부터 총 입력 접점 수 - 1 까지 사용할 수 있다.

Example

C++

```
// 전체 모듈, Offset 0번지에서 bit 단위로 레벨을 설정한다.  
AxdoLevelSetOutport(0, 1);
```

Visual Basic

```
' 전체 모듈, Offset 0번지에서 bit 단위로 레벨을 설정한다.  
AxdoLevelSetOutport 0, 1
```

Delphi

```
{ 전체 모듈, Offset 0번지에서 bit 단위로 레벨을 설정한다. }
AxdoLevelSetOutport(0, 1);
```

See Also

[AxdoLevelSetOutportBit](#), [AxdoLevelSetOutportByte](#), [AxdoLevelSetOutportWord](#),
[AxdoLevelSetOutportDword](#), [AxdoLevelGetOutport](#)

AxdoLevelGetOutport

Purpose

전체 출력 접점 모듈의 Offset 위치에서 bit 단위로 레벨을 확인한다.

레벨 확인은 Active 신호 상태가 HIGH(1) 인지 LOW(0) 인지 확인한다.

Format

C++

```
DWORD AxdoLevelGetOutport(long lOffset, DWORD *upLevel);
```

Visual Basic

```
Function AxdoLevelGetOutport(ByVal lOffset As Long, ByRef upLevel As Long) As Long
```

Delphi

```
function AxdoLevelGetOutport(lOffset : LongInt; upLevel : PDWord) : DWord; stdcall;
```

Parameters

[in/out] Name	[Init Value] Explanation
[>>]lOffset	전체 출력 접점에 대한 Offset 위치
[<<]upLevel	레벨 값 (Boolean)

Return Values

[0000] AXT_RT_SUCCESS : AXL 라이브러리 초기화 성공

[3101] AXT_RT_DIO_INVALID_MODULE_NO : 유효하지않는 DIO 모듈 번호

[3102] AXT_RT_DIO_INVALID_OFFSET_NO : 유효하지않는 DIO OFFSET 번호

* See error code Table for more information on status error codes

Description

전체 입력 접점 모듈의 Offset 위치에서 bit 단위로 레벨을 확인한다.

Offset 은 '0'부터 총 입력 접점 수 - 1 까지 사용할 수 있다.

Example

C++

```
// 전체 모듈, Offset 0번지에서 bit 단위로 레벨을 확인한다.  
DWORD dwLevel;  
  
AxdoLevelGetOutport(0, &dwLevel);
```

Visual Basic

' 전체 모듈, Offset 0번지에서 bit 단위로 레벨을 확인한다.

```
Dim lLevel As long
```

```
AxdoLevelGetOutport 0, lLevel
```

Delphi

```
{ 전체 모듈, Offset 0번지에서 bit 단위로 레벨을 확인한다. }
var
    dwLevel : DWord;

begin
    AxdoLevelGetOutport(0, @dwLevel);
end
```

See Also

[AxdoLevelGetOutportBit](#), [AxdoLevelGetOutportByte](#), [AxdoLevelGetOutportWord](#),
[AxdoLevelGetOutportDword](#), [AxdoLevelSetOutport](#)

입력 포트 읽기

Function	Description
AxdiReadInportBit	지정한 입력 접점 모듈의 Offset 위치에서 bit 단위로 데이터를 읽는다.
AxdiReadInportByte	지정한 입력 접점 모듈의 Offset 위치에서 byte 단위로 데이터를 읽는다.
AxdiReadInportWord	지정한 입력 접점 모듈의 Offset 위치에서 word 단위로 데이터를 읽는다.
AxdiReadInportDword	지정한 입력 접점 모듈의 Offset 위치에서 double word 단위로 데이터를 읽는다.
AxdiReadInport	전체 입력 접점 모듈의 Offset 위치에서 bit 단위로 데이터를 읽는다.

AxdiReadInportBit

Purpose

지정한 입력 접점 모듈의 Offset 위치에서 bit 단위로 데이터를 읽는다.

Format

C++

```
DWORD AxdiReadInportBit(long lModuleNo, long lOffset, DWORD *upValue);
```

Visual Basic

```
Function AxdiReadInportBit(ByVal lModuleNo As Long, ByVal lOffset As Long, ByRef upValue As Long) As Long
```

Delphi

```
function AxdiReadInportBit(lModuleNo : LongInt; lOffset : LongInt;
  upValue : PWord) : Word; stdcall;
```

Parameters

[in/out] Name	[Init Value] Explanation
[>>]lModuleNo	모듈 번호
[>>]lOffset	입력 접점에 대한 Offset 위치
[<<]upValue	입력 접점 값 (Boolean)

Return Values

[0000] AXT_RT_SUCCESS : AXL 라이브러리 초기화 성공

[3051] AXT_RT_DIO_NOT_MODULE : DIO 모듈 없음

[3101] AXT_RT_DIO_INVALID_MODULE_NO : 유효하지않는 DIO 모듈 번호

[3102] AXT_RT_DIO_INVALID_OFFSET_NO : 유효하지않는 DIO OFFSET 번호

* See error code Table for more information on status error codes

Description

사용자가 지정한 입력 접점 모듈의 Offset 위치에서 bit 단위로 데이터를 읽는다.

지정한 모듈이 어떤 모듈인지 확인하여 아래의 Table에서 Offset 사용 범위를 참조하여 읽는다.

Offset 범위와 Return 값

Module	Offset	Value
SIO-DI32	0 ~ 31	0(Off), 1(On)
SIO-DO32x	-	-
SIO-DB32x	0 ~ 15	0(Off), 1(On)

Example

C++

```
// 0번째 모듈에서 offset 0번지에 데이터를 bit 단위로 읽는다.  
DWORD dwValue;  
  
AxdiReadImportBit(0, 0, &dwValue);
```

Visual Basic

```
' 0번째 모듈에서 Offset 0번지에 데이터를 bit 단위로 읽는다.  
Dim lValue As Long  
  
AxdiReadImportBit 0, 0, lValue
```

Delphi

```
{ 0번째 모듈에서 offset 0번지에 데이터를 bit 단위로 읽는다. }  
var  
  dwValue : DWord;  
  
begin  
  AxdiReadImportBit(0, 0, @dwValue);  
end;
```

See Also

[AxdiReadImportByte](#), [AxdiReadImportWord](#), [AxdiReadImportDword](#), [AxdiReadImport](#)

AxdiReadInportByte

Purpose

지정한 입력 접점 모듈의 Offset 위치에서 byte 단위로 데이터를 읽는다.

Format

C++

```
DWORD AxdiReadInportByte(long lModuleNo, long lOffset, DWORD
    *upValue);
```

Visual Basic

```
Function AxdiReadInportByte(ByVal lModuleNo As Long, ByVal lOffset As
    Long, ByRef upValue As Long) As Long
```

Delphi

```
function AxdiReadInportByte(lModuleNo : LongInt; lOffset : LongInt;
    upValue : PWord) : Word; stdcall;
```

Parameters

[in/out] Name	[Init Value] Explanation
[>>]lModuleNo	모듈 번호
[>>]lOffset	입력 접점에 대한 Offset 위치
[<<]upValue	입력 접점 값 (Byte)

Return Values

[0000] AXT_RT_SUCCESS : AXL 라이브러리 초기화 성공

[3051] AXT_RT_DIO_NOT_MODULE : DIO 모듈 없음

[3101] AXT_RT_DIO_INVALID_MODULE_NO : 유효하지않는 DIO 모듈 번호

[3102] AXT_RT_DIO_INVALID_OFFSET_NO : 유효하지않는 DIO OFFSET 번호* [See error code Table for more information on status error codes](#)

Description

사용자가 지정한 입력 접점 모듈의 Offset 위치에서 byte 단위로 데이터를 읽는다.

지정한 모듈이 어떤 모듈인지 확인하여 아래의 Table에서 Offset 사용 범위를 참조하여 읽는다.

Offset 범위와 Return 값

Module	Offset	Value
SIO-DI32	0, 1, 2, 3	00h ~ FFh
SIO-DO32x	-	-
SIO-DB32x	0, 1	00h ~ FFh

Example

C++

```
// 0번째 모듈에서 offset 0번지에 데이터를 byte 단위로 읽는다.  
DWORD dwValue;  
  
AxdiReadInportByte(0, 0, &dwValue);
```

Visual Basic

```
' 0번째 모듈에서 Offset 0번지에 데이터를 byte 단위로 읽는다.  
Dim lValue As Long  
  
AxdiReadInportByte 0, 0, lValue
```

Delphi

```
{ 0번째 모듈에서 offset 0번지에 데이터를 byte 단위로 읽는다. }  
var  
  dwValue : DWord;  
  
begin  
  AxdiReadInportByte(0, 0, @dwValue);  
end;
```

See Also

[AxdiReadInportBit](#), [AxdiReadInportWord](#), [AxdiReadInportDword](#), [AxdiReadInport](#)

AxdiReadInportWord

Purpose

지정한 입력 접점 모듈의 Offset 위치에서 word 단위로 데이터를 읽는다.

Format

C++

```
DWORD AxdiReadInportWord(long lModuleNo, long lOffset, DWORD
    *upValue);
```

Visual Basic

```
Function AxdiReadInportWord(ByVal lModuleNo As Long, ByVal lOffset As
    Long, ByRef upValue As Long) As Long
```

Delphi

```
function AxdiReadInportWord(lModuleNo : LongInt; lOffset : LongInt;
    upValue : PWord) : Word; stdcall;
```

Parameters

[in/out] Name	[Init Value] Explanation
[>>]lModuleNo	모듈 번호
[>>]lOffset	입력 접점에 대한 Offset 위치
[<<]upValue	입력 접점 값 (Word)

Return Values

[0000] AXT_RT_SUCCESS : AXL 라이브러리 초기화 성공

[3051] AXT_RT_DIO_NOT_MODULE : DIO 모듈 없음

[3101] AXT_RT_DIO_INVALID_MODULE_NO : 유효하지않는 DIO 모듈 번호

[3102] AXT_RT_DIO_INVALID_OFFSET_NO : 유효하지않는 DIO OFFSET 번호

* See error code Table for more information on status error codes

Description

사용자가 지정한 입력 접점 모듈의 Offset 위치에서 word 단위로 데이터를 읽는다.

지정한 모듈이 어떤 모듈인지 확인하여 아래의 Table에서 Offset 사용 범위를 참조하여 읽는다.

Offset 범위와 Return 값

Module	Offset	Value
SIO-DI32	0, 1	0000h ~ FFFFh
SIO-DO32x	-	-
SIO-DB32x	0	0000h ~ FFFFh

Example

C++

```
// 0번째 모듈에서 offset 0번지에 데이터를 word 단위로 읽는다.  
DWORD dwValue;  
  
AxdiReadImportWord (0, 0, &dwValue);
```

Visual Basic

```
' 0번째 모듈에서 Offset 0번지에 데이터를 word 단위로 읽는다.  
Dim lValue As Long  
  
AxdiReadImportWord 0, 0, lValue
```

Delphi

```
{ 0번째 모듈에서 offset 0번지에 데이터를 word 단위로 읽는다. }  
var  
    dwValue : DWord;  
  
begin  
    AxdiReadImportWord (0, 0, @dwValue);  
end;
```

See Also

[AxdiReadImportBit](#), [AxdiReadImportByte](#), [AxdiReadImportDword](#), [AxdiReadImport](#)

AxdiReadImportDword

Purpose

지정한 입력 접점 모듈의 Offset 위치에서 double word 단위로 데이터를 읽는다.

Format

C++

```
DWORD AxdiReadImportDword(long lModuleNo, long lOffset, DWORD
    *upValue);
```

Visual Basic

```
Function AxdiReadImportDword(ByVal lModuleNo As Long, ByVal lOffset
    As Long, ByRef upValue As Long) As Long
```

Delphi

```
function AxdiReadImportDword(lModuleNo : LongInt; lOffset : LongInt;
    upValue : PDWord) : DWord; stdcall;
```

Parameters

[in/out] Name	[Init Value] Explanation
[>>]lModuleNo	모듈 번호
[>>]lOffset	입력 접점에 대한 Offset 위치
[<<]upValue	입력 접점 값 (double word)

Return Values

[0000] AXT_RT_SUCCESS : AXL 라이브러리 초기화 성공

[3051] AXT_RT_DIO_NOT_MODULE : DIO 모듈 없음

[3101] AXT_RT_DIO_INVALID_MODULE_NO : 유효하지않는 DIO 모듈 번호

[3102] AXT_RT_DIO_INVALID_OFFSET_NO : 유효하지않는 DIO OFFSET 번호

* See error code Table for more information on status error codes

Description

사용자가 지정한 입력 접점 모듈의 Offset 위치에서 double word 단위로 데이터를 읽는다.

지정한 모듈이 어떤 모듈인지 확인하여 아래의 Table에서 Offset 사용 범위를 참조하여 읽는다.

Offset 범위와 Return 값

Module	Offset	Value
SIO-DI32	0	00000000h ~ FFFFFFFFh
SIO-DO32x	-	-
SIO-DB32x	0	00000000h ~ FFFFFFFFh

Example

C++

```
// 0번째 모듈에서 offset 0번지에 데이터를 double word 단위로 읽는다.  
DWORD dwValue;  
  
AxdiReadImportDword (0, 0, &dwValue);
```

Visual Basic

```
' 0번째 모듈에서 Offset 0번지에 데이터를 double word 단위로 읽는다.  
Dim lValue As Long  
  
AxdiReadImportDword 0, 0, lValue
```

Delphi

```
{ 0번째 모듈에서 offset 0번지에 데이터를 double word 단위로 읽는다. }  
var  
  dwValue : DWord;  
  
begin  
  AxdiReadImportDword (0, 0, @dwValue);  
end;
```

See Also

[AxdiReadImportBit](#), [AxdiReadImportByte](#), [AxdiReadImportWord](#), [AxdiReadImport](#)

AxdiReadInport

Purpose

지정한 입력 접점 모듈의 Offset 위치에서 bit 단위로 데이터를 읽는다.

Format

C++

```
DWORD AxdiReadInport(long lOffset, DWORD *upValue);
```

Visual Basic

```
Function AxdiReadInport(ByIdVal lOffset As Long, ByRef upValue As Long)
As Long
```

Delphi

```
function AxdiReadInport(lOffset : LongInt; upValue : PDWord) : DWord;
stdcall;
```

Parameters

[in/out] Name	[Init Value] Explanation
[>>]lOffset	전체 입력 접점에 대한 Offset 위치
[<<]upValue	입력 접점 값 (Boolean)

Return Values

[0000] AXT_RT_SUCCESS : AXL 라이브러리 초기화 성공

[3051] AXT_RT_DIO_NOT_MODULE : DIO 모듈 없음

[3102] AXT_RT_DIO_INVALID_OFFSET_NO : 유효하지않는 DIO OFFSET 번호

* See error code Table for more information on status error codes

Description

전체 입력 접점 모듈의 Offset 위치에서 bit 단위로 데이터를 읽는다.

Offset은 '0'부터 총 입력 접점 수 - 1 까지 사용할 수 있다.

Example

C++

```
// 전체 출력 모듈의 offset 0번지에서 bit 단위로 데이터를 읽는다.
DWORD dwValue;

AxdiReadInport (0, &dwValue);
```

Visual Basic

' 전체 출력 모듈의 offset 0번지에서 bit 단위로 데이터를 읽는다.

```
Dim lValue As Long
```

```
AxdiReadInport 0, lValue
```

Delphi

```
{ 전체 출력 모듈의 offset 0번지에서 bit 단위로 데이터를 읽는다. }  
var  
    dwValue : DWord;  
  
begin  
    AxdiReadImport(0, @dwValue);  
end;
```

See Also

[AxdiReadImportBit](#), [AxdiReadImportByte](#), [AxdiReadImportWord](#), [AxdiReadImportDword](#)

출력 포트 읽기 쓰기

Function	Description
AxdoWriteOutportBit	지정한 출력 접점 모듈의 Offset 위치에서 bit 단위로 데이터를 출력한다.
AxdoWriteOutportByte	지정한 출력 접점 모듈의 Offset 위치에서 byte 단위로 데이터를 출력한다.
AxdoWriteOutportWord	지정한 출력 접점 모듈의 Offset 위치에서 word 단위로 데이터를 출력한다.
AxdoWriteOutportDword	지정한 출력 접점 모듈의 Offset 위치에서 double word 단위로 데이터를 출력한다.
AxdoReadOutportBit	지정한 출력 접점 모듈의 Offset 위치에서 bit 단위로 데이터를 읽는다.
AxdoReadOutportByte	지정한 출력 접점 모듈의 Offset 위치에서 byte 단위로 데이터를 읽는다.
AxdoReadOutportWord	지정한 출력 접점 모듈의 Offset 위치에서 word 단위로 데이터를 읽는다.
AxdoReadOutportDword	지정한 출력 접점 모듈의 Offset 위치에서 double word 단위로 데이터를 읽는다.
AxdoWriteOutport	전체 출력 접점 모듈의 Offset 위치에서 bit 단위로 데이터를 출력한다.
AxdoReadOutport	전체 출력 접점 모듈의 Offset 위치에서 bit 단위로 데이터를 읽는다.

AxdoWriteOutportBit

Purpose

지정한 출력 접점 모듈의 Offset 위치에서 bit 단위로 데이터를 출력한다.

Format

C++

```
DWORD AxdoWriteOutportBit(long lModuleNo, long lOffset, DWORD uValue);
```

Visual Basic

```
Function AxdoWriteOutportBit(ByVal lModuleNo As Long, ByVal lOffset As Long, ByVal uValue As Long) As Long
```

Delphi

```
function AxdoWriteOutportBit(lModuleNo : LongInt; lOffset : LongInt; uValue : DWord) : DWord; stdcall;
```

Parameters

[in/out] Name	[Init Value] Explanation
[>>]lModuleNo	모듈 번호
[>>]lOffset	출력 접점에 대한 Offset 위치
[>>]uValue	출력 접점 값 (Boolean)

Return Values

[0000] AXT_RT_SUCCESS : AXL 라이브러리 초기화 성공

[3051] AXT_RT_DIO_NOT_MODULE : DIO 모듈 없음

[3101] AXT_RT_DIO_INVALID_MODULE_NO : 유효하지않는 DIO 모듈 번호

[3102] AXT_RT_DIO_INVALID_OFFSET_NO : 유효하지않는 DIO OFFSET 번호

* See error code Table for more information on status error codes

Description

사용자가 지정한 출력 접점 모듈의 Offset 위치에서 bit 단위로 데이터를 출력한다.

지정한 모듈이 어떤 모듈인지 확인하여 아래의 Table에서 Offset 사용 범위를 참조하여 설정한다.

Offset 범위와 Value 값

Module	Offset	bValue
SIO-DO32x	0 ~ 31	0(Off), 1(On)
SIO-DB32x	0 ~ 15	0(Off), 1(On)

Example

C++

```
// 0번째 출력 접점 모듈의 offset 0번지에서 bit 단위로 데이터를 출력한다.  
AxdoWriteOutportBit(0, 0, 1);
```

Visual Basic

' 0번째 출력 접점 모듈의 offset 0번지에서 bit 단위로 데이터를 출력한다.
AxdoWriteOutportBit 0, 0, 1

Delphi

{ 0번째 출력 접점 모듈의 offset 0번지에서 bit 단위로 데이터를 출력한다. }
AxdoWriteOutportBit(0, 0, 1);

See Also

[AxdoWriteOutportByte](#), [AxdoWriteOutportWord](#), [AxdoWriteOutportDword](#), [AxdoReadOutportBit](#),
[AxdoWriteOutport](#)

AxdoWriteOutportByte

Purpose

지정한 출력 접점 모듈의 Offset 위치에서 byte 단위로 데이터를 출력한다.

Format

C++

```
DWORD AxdoWriteOutportByte(long lModuleNo, long lOffset, DWORD uValue);
```

Visual Basic

```
Function AxdoWriteOutportByte(ByVal lModuleNo As Long, ByVal lOffset As Long, ByVal uValue As Long) As Long
```

Delphi

```
function AxdoWriteOutportByte(lModuleNo : LongInt; lOffset : LongInt; uValue : DWord) : DWord; stdcall;
```

Parameters

[in/out] Name	[Init Value] Explanation
[>>]lModuleNo	모듈 번호
[>>]lOffset	출력 접점에 대한 Offset 위치
[>>]uValue	출력 접점 값 (Byte)

Return Values

[0000] AXT_RT_SUCCESS : AXL 라이브러리 초기화 성공

[3051] AXT_RT_DIO_NOT_MODULE : DIO 모듈 없음

[3101] AXT_RT_DIO_INVALID_MODULE_NO : 유효하지않는 DIO 모듈 번호

[3102] AXT_RT_DIO_INVALID_OFFSET_NO : 유효하지않는 DIO OFFSET 번호

* See error code Table for more information on status error codes

Description

사용자가 지정한 출력 접점 모듈의 Offset 위치에서 byte 단위로 데이터를 출력한다.

지정한 모듈이 어떤 모듈인지 확인하여 아래의 Table에서 Offset 사용 범위를 참조하여 설정한다.

Offset 범위와 Value 값

Module	Offset	bValue
SIO-DO32x	0, 1, 2, 3	00h ~ FFh
SIO-DB32x	0, 1	00h ~ FFh

Example

C++

```
// 0번째 출력 접점 모듈의 offset 0번지에서 byte 단위로 데이터를 출력한다.  
AxdoWriteOutportByte(0, 0, 0xFF);
```

Visual Basic

```
' 0번째 출력 접점 모듈의 offset 0번지에서 byte 단위로 데이터를 출력한다.  
AxdoWriteOutportByte 0, 0, &hFF
```

Delphi

```
{ 0번째 출력 접점 모듈의 offset 0번지에서 byte 단위로 데이터를 출력한다. }  
AxdoWriteOutportByte (0, 0, $FF)
```

See Also

[AxdoWriteOutportBit](#), [AxdoWriteOutportWord](#), [AxdoWriteOutportDword](#), [AxdoReadOutportByte](#),
[AxdoWriteOutport](#)

AxdoWriteOutportWord

Purpose

지정한 출력 접점 모듈의 Offset 위치에서 word 단위로 데이터를 출력한다.

Format

C++

```
DWORD AxdoWriteOutportByte(long lModuleNo, long lOffset, DWORD uValue);
```

Visual Basic

```
Function AxdoWriteOutportByte(ByVal lModuleNo As Long, ByVal lOffset As Long, ByVal uValue As Long) As Long
```

Delphi

```
function AxdoWriteOutportByte(lModuleNo : LongInt; lOffset : LongInt; uValue : DWord) : DWord; stdcall;
```

Parameters

[in/out] Name	[Init Value] Explanation
[>>]lModuleNo	모듈 번호
[>>]lOffset	출력 접점에 대한 Offset 위치
[>>]uValue	출력 접점 값 (Word)

Return Values

[0000] AXT_RT_SUCCESS : AXL 라이브러리 초기화 성공

[3051] AXT_RT_DIO_NOT_MODULE : DIO 모듈 없음

[3101] AXT_RT_DIO_INVALID_MODULE_NO : 유효하지않는 DIO 모듈 번호

[3102] AXT_RT_DIO_INVALID_OFFSET_NO : 유효하지않는 DIO OFFSET 번호

* See error code Table for more information on status error codes

Description

사용자가 지정한 출력 접점 모듈의 Offset 위치에서 word 단위로 데이터를 출력한다.

지정한 모듈이 어떤 모듈인지 확인하여 아래의 Table에서 Offset 사용 범위를 참조하여 설정한다.

Offset 범위와 Value 값

Module	Offset	bValue
SIO-DO32x	0,1	0000h ~ FFFFh
SIO-DB32x	0	0000h ~ FFFFh

Example

C++

```
// 0번째 출력 접점 모듈의 Offset 0번지에서 word 단위로 데이터를 출력한다.  
AxdoWriteOutportWord(0, 0, 0xFFFF);
```

Visual Basic

```
' 0번째 출력 접점 모듈의 Offset 0번지에서 word 단위로 데이터를 출력한다.  
AxdoWriteOutportWord 0, 0, &hFFFF
```

Delphi

```
{ 0번째 출력 접점 모듈의 Offset 0번지에서 word 단위로 데이터를 출력한다. }  
AxdoWriteOutportWord(0, 0, $FFFF);
```

See Also

[AxdoWriteOutportBit](#), [AxdoWriteOutportByte](#), [AxdoWriteOutportDword](#), [AxdoReadOutportWord](#),
[AxdoWriteOutport](#)

AxdoWriteOutportDword

Purpose

지정한 출력 접점 모듈의 Offset 위치에서 double word 단위로 데이터를 출력한다.

Format

C++

```
DWORD AxdoWriteOutportDword(long lModuleNo, long lOffset, DWORD uValue);
```

Visual Basic

```
Function AxdoWriteOutportDword(ByVal lModuleNo As Long, ByVal lOffset As Long, ByVal uValue As Long) As Long
```

Delphi

```
function AxdoWriteOutportDword(lModuleNo : LongInt; lOffset : LongInt; uValue : DWord) : DWord; stdcall;
```

Parameters

[in/out] Name	[Init Value] Explanation
[>>]lModuleNo	모듈 번호
[>>]lOffset	출력 접점에 대한 Offset 위치
[>>]uValue	출력 접점 값 (Double Word)

Return Values

[0000] AXT_RT_SUCCESS : AXL 라이브러리 초기화 성공

[3051] AXT_RT_DIO_NOT_MODULE : DIO 모듈 없음

[3101] AXT_RT_DIO_INVALID_MODULE_NO : 유효하지않는 DIO 모듈 번호

[3102] AXT_RT_DIO_INVALID_OFFSET_NO : 유효하지않는 DIO OFFSET 번호

* See error code Table for more information on status error codes

Description

사용자가 지정한 출력 접점 모듈의 Offset 위치에서 double word 단위로 데이터를 출력한다.

지정한 모듈이 어떤 모듈인지 확인하여 아래의 Table에서 Offset 사용 범위를 참조하여 설정한다.

Offset 범위와 Value 값

Module	Offset	bValue
SIO-DO32x	0	00000000h ~ FFFFFFFFh
SIO-DB32x	0	00000000h ~ FFFFFFFFh

Example

C++

```
// 0번째 출력 접점 모듈의 Offset 0번지에서 double word 단위로 데이터를 출력한다.  
AxdoWriteOutportDword(0, 0, 0xFFFFFFFF);
```

Visual Basic

```
' 0번째 출력 접점 모듈의 Offset 0번지에서 double word 단위로 데이터를 출력한다.  
AxdoWriteOutportDword 0, 0, &hFFFFFF
```

Delphi

```
{ 0번째 출력 접점 모듈의 Offset 0번지에서 double word 단위로 데이터를 출력한다. }  
AxdoWriteOutportDword(0, 0, $FFFFFF);
```

See Also

[AxdoWriteOutportBit](#), [AxdoWriteOutportByte](#), [AxdoWriteOutportWord](#), [AxdoReadOutportDword](#),
[AxdoWriteOutport](#)

AxdoReadOutportBit

Purpose

지정한 출력 접점 모듈의 Offset 위치에서 bit 단위로 데이터를 읽는다.

Format

C++

```
DWORD AxdoReadOutportBit(long lModuleNo, long lOffset, DWORD
*upValue);
```

Visual Basic

```
Function AxdoReadOutportBit(ByVal lModuleNo As Long, ByVal lOffset As
Long, ByRef upValue As Long) As Long
```

Delphi

```
function AxdoReadOutportBit(lModuleNo : LongInt; lOffset : LongInt;
upValue : PWord) : Word; stdcall;
```

Parameters

[in/out] Name	[Init Value] Explanation
[>>]lModuleNo	모듈 번호
[>>]lOffset	출력 접점에 대한 Offset 위치
[<<]upValue	출력 접점 값 (Boolean)

Return Values

[0000] AXT_RT_SUCCESS : AXL 라이브러리 초기화 성공

[3051] AXT_RT_DIO_NOT_MODULE : DIO 모듈 없음

[3101] AXT_RT_DIO_INVALID_MODULE_NO : 유효하지않는 DIO 모듈 번호

[3102] AXT_RT_DIO_INVALID_OFFSET_NO : 유효하지않는 DIO OFFSET 번호

* See error code Table for more information on status error codes

Description

사용자가 지정한 출력 접점 모듈의 Offset 위치에서 bit 단위로 데이터를 읽는다.

지정한 모듈이 어떤 모듈인지 확인하여 아래의 Table에서 Offset 사용 범위를 참조하여 읽는다.

Offset 범위와 Return 값

Module	Offset	Value
SIO-DI32	-	-
SIO-DO32x	0 ~ 31	0(Off), 1(On)
SIO-DB32x	0 ~ 15	0(Off), 1(On)

Example

C++

```
// 0번째 모듈에서 offset 0번지에 데이터를 bit 단위로 읽는다.  
DWORD dwValue;  
  
AxdoReadOutportBit(0, 0, &dwValue);
```

Visual Basic

```
' 0번째 모듈에서 Offset 0번지에 데이터를 bit 단위로 읽는다.  
Dim lValue As Long  
  
AxdoReadOutportBit 0, 0, lValue
```

Delphi

```
{ 0번째 모듈에서 offset 0번지에 데이터를 bit 단위로 읽는다. }  
var  
  dwValue : DWord;  
  
begin  
  AxdoReadOutportBit(0, 0, @dwValue);  
end;
```

See Also

[AxdoReadOutportByte](#), [AxdoReadOutportWord](#), [AxdoReadOutportDword](#), [AxdoWriteOutportBit](#),
[AxdoReadOutport](#)

AxdoReadOutportByte

Purpose

지정한 출력 접점 모듈의 Offset 위치에서 byte 단위로 데이터를 읽는다.

Format

C++

```
Function AxdoReadOutportByte(long lModuleNo, long lOffset, DWORD
    *upValue);
```

Visual Basic

```
Function AxdoReadOutportByte(ByVal lModuleNo As Long, ByVal lOffset
    As Long, ByRef upValue As Long) As Long
```

Delphi

```
function AxdoReadOutportByte(lModuleNo : LongInt; lOffset : LongInt;
    upValue : PDWord) : DWord; stdcall;
```

Parameters

[in/out] Name	[Init Value] Explanation
[>>]lModuleNo	모듈 번호
[>>]lOffset	출력 접점에 대한 Offset 위치
[<<]upValue	출력 접점 값 (Byte)

Return Values

[0000] AXT_RT_SUCCESS : AXL 라이브러리 초기화 성공

[3051] AXT_RT_DIO_NOT_MODULE : DIO 모듈 없음

[3101] AXT_RT_DIO_INVALID_MODULE_NO : 유효하지않는 DIO 모듈 번호

[3102] AXT_RT_DIO_INVALID_OFFSET_NO : 유효하지않는 DIO OFFSET 번호

* See error code Table for more information on status error codes

Description

사용자가 지정한 출력 접점 모듈의 Offset 위치에서 byte 단위로 데이터를 읽는다.

지정한 모듈이 어떤 모듈인지 확인하여 아래의 Table에서 Offset 사용 범위를 참조하여 읽는다.

Offset 범위와 Return 값

Module	Offset	Value
SIO-DI32	-	-
SIO-DO32x	0, 1, 2, 3	00h ~ FFh
SIO-DB32x	0, 1	00h ~ FFh

Example

C++

```
// 0번째 모듈에서 offset 0번지에 데이터를 byte 단위로 읽는다.  
DWORD dwValue;  
  
AxdoReadOutportByte(0, 0, &dwValue);
```

Visual Basic

```
' 0번째 모듈에서 Offset 0번지에 데이터를 byte 단위로 읽는다.  
Dim lValue As Long  
  
AxdoReadOutportByte 0, 0, lValue
```

Delphi

```
{ 0번째 모듈에서 offset 0번지에 데이터를 byte 단위로 읽는다. }  
var  
  dwValue : DWord;  
  
begin  
  AxdoReadOutportByte(0, 0, @dwValue);  
end;
```

See Also

[AxdoReadOutportBit](#), [AxdoReadOutportWord](#), [AxdoReadOutportDword](#), [AxdoWriteOutportByte](#),
[AxdoReadOutport](#)

AxdoReadOutportWord

Purpose

지정한 출력 접점 모듈의 Offset 위치에서 word 단위로 데이터를 읽는다.

Format

C++

```
DWORD AxdoReadOutportWord(long lModuleNo, long lOffset, DWORD
    *upValue);
```

Visual Basic

```
Function AxdoReadOutportWord(ByVal lModuleNo As Long, ByVal lOffset
    As Long, ByRef upValue As Long) As Long
```

Delphi

```
function AxdoReadOutportWord(lModuleNo : LongInt; lOffset : LongInt;
    upValue : PWord) : Word; stdcall;
```

Parameters

[in/out] Name	[Init Value] Explanation
[>>]lModuleNo	모듈 번호
[>>]lOffset	출력 접점에 대한 Offset 위치
[<<]upValue	출력 접점 값 (Word)

Return Values

[0000] AXT_RT_SUCCESS : AXL 라이브러리 초기화 성공

[3051] AXT_RT_DIO_NOT_MODULE : DIO 모듈 없음

[3101] AXT_RT_DIO_INVALID_MODULE_NO : 유효하지않는 DIO 모듈 번호

[3102] AXT_RT_DIO_INVALID_OFFSET_NO : 유효하지않는 DIO OFFSET 번호

* See error code Table for more information on status error codes

Description

사용자가 지정한 출력 접점 모듈의 Offset 위치에서 word 단위로 데이터를 읽는다.

지정한 모듈이 어떤 모듈인지 확인하여 아래의 Table에서 Offset 사용 범위를 참조하여 읽는다.

Offset 범위와 Return 값

Module	Offset	Value
SIO-DI32	-	-
SIO-DO32x	0, 1	0000h ~ FFFFh
SIO-DB32x	0	0000h ~ FFFFh

Example

C++

```
// 0번째 모듈에서 offset 0번지에 데이터를 word 단위로 읽는다.  
DWORD dwValue;  
  
AxdoReadOutportWord(0, 0, &dwValue);
```

Visual Basic

```
' 0번째 모듈에서 Offset 0번지에 데이터를 word 단위로 읽는다.  
Dim lValue As Long  
  
AxdoReadOutportWord 0, 0, lValue
```

Delphi

```
{ 0번째 모듈에서 offset 0번지에 데이터를 word 단위로 읽는다. }  
var  
  dwValue : DWord;  
  
begin  
  AxdoReadOutportWord(0, 0, @dwValue);  
end;
```

See Also

[AxdoReadOutportBit](#), [AxdoReadOutportByte](#), [AxdoReadOutportDword](#), [AxdoWriteOutportWord](#),
[AxdoReadOutport](#)

AxdoReadOutportDword

Purpose

지정한 출력 접점 모듈의 Offset 위치에서 double word 단위로 데이터를 읽는다.

Format

C++

```
DWORD AxdoReadOutportDword(long lModuleNo, long lOffset, DWORD
                           *upValue);
```

Visual Basic

```
Function AxdoReadOutportDword(ByVal lModuleNo As Long, ByVal lOffset
                               As Long, ByRef upValue As Long) As Long
```

Delphi

```
function AxdoReadOutportDword(lModuleNo : LongInt; lOffset : LongInt;
                               upValue : PDWord) : DWord; stdcall;
```

Parameters

[in/out] Name	[Init Value] Explanation
[>>]lModuleNo	모듈 번호
[>>]lOffset	출력 접점에 대한 Offset 위치
[<<]upValue	출력 접점 값 (Double Word)

Return Values

[0000] AXT_RT_SUCCESS : AXL 라이브러리 초기화 성공

[3051] AXT_RT_DIO_NOT_MODULE : DIO 모듈 없음

[3101] AXT_RT_DIO_INVALID_MODULE_NO : 유효하지않는 DIO 모듈 번호

[3102] AXT_RT_DIO_INVALID_OFFSET_NO : 유효하지않는 DIO OFFSET 번호

* See error code Table for more information on status error codes

Description

사용자가 지정한 출력 접점 모듈의 Offset 위치에서 double word 단위로 데이터를 읽는다.

지정한 모듈이 어떤 모듈인지 확인하여 아래의 Table에서 Offset 사용 범위를 참조하여 읽는다.

Offset 범위와 Return 값

Module	Offset	Value
SIO-DI32	-	-
SIO-DO32x	0	00000000h ~ FFFFFFFFh
SIO-DB32x	0	00000000h ~ FFFFFFFFh

Example

C++

```
// 0번째 모듈에서 offset 0번지에 데이터를 double word 단위로 읽는다.  
DWORD dwValue;  
  
AxdoReadOutportDword(0, 0, &dwValue);
```

Visual Basic

```
' 0번째 모듈에서 Offset 0번지에 데이터를 double word 단위로 읽는다.  
Dim lValue As Long  
  
AxdoReadOutportDword 0, 0, lValue
```

Delphi

```
{ 0번째 모듈에서 offset 0번지에 데이터를 double word 단위로 읽는다. }  
var  
  dwValue : DWord;  
  
begin  
  AxdoReadOutportDword(0, 0, @dwValue);  
end;
```

See Also

[AxdoReadOutportBit](#), [AxdoReadOutportByte](#), [AxdoReadOutportWord](#), [AxdoWriteOutportDword](#),
[AxdoReadOutport](#)

AxdoWriteOutport

Purpose

전체 출력 접점 모듈의 Offset 위치에서 bit 단위로 데이터를 출력한다.

Format

C++

```
DWORD AxdoWriteOutport (long lOffset, DWORD uValue);
```

Visual Basic

```
Function AxdoWriteOutport (ByVal lOffset As Long, ByVal uValue As Long)
As Long
```

Delphi

```
Function AxdoWriteOutport (lOffset : LongInt; uValue : DWord) : DWord;
stdcall;
```

Parameters

[in/out] Name	[Init Value] Explanation
[>>]lOffset	전체 출력 접점에 대한 Offset 위치
[>>]uValue	출력 접점 값 (Boolean)

Return Values

[0000] AXT_RT_SUCCESS : AXL 라이브러리 초기화 성공

[3051] AXT_RT_DIO_NOT_MODULE : DIO 모듈 없음

[3101] AXT_RT_DIO_INVALID_MODULE_NO : 유효하지않는 DIO 모듈 번호

[3102] AXT_RT_DIO_INVALID_OFFSET_NO : 유효하지않는 DIO OFFSET 번호

* See error code Table for more information on status error codes

Description

전체 출력 접점 모듈의 Offset 위치에서 bit 단위로 데이터를 출력한다.

Offset은 '0'부터 총 출력 접점 수 - 1 까지 사용할 수 있다.

Example

C++

```
// 전체 출력 접점 모듈의 Offset 0번지에서 bit 단위로 데이터를 출력한다.
AxdoWriteOutport (0, 1);
```

Visual Basic

```
' 전체 출력 접점 모듈의 offset 0번지에서 bit 단위로 데이터를 출력한다.
AxdoWriteOutport 0, 1
```

Delphi

```
{ 전체 출력 접점 모듈의 offset 0번지에서 bit 단위로 데이터를 출력한다. }  
AxdoWriteOutport(0, 1);
```

See Also

[AxdoWriteOutportBit](#), [AxdoWriteOutportByte](#), [AxdoWriteOutportWord](#), [AxdoWriteOutportDword](#),
[AxdoReadOutport](#)

AxdoReadOutport

Purpose

전체 출력 접점 모듈의 Offset 위치에서 bit 단위로 데이터를 읽는다.

Format

C++

```
DWORD AxdoReadOutport(long lOffset, DWORD *upValue);
```

Visual Basic

```
Function AxdoReadOutport(ByIdVal lOffset As Long, ByRef upValue As Long)
As Long
```

Delphi

```
function AxdoReadOutport(lOffset : LongInt; upValue : PDWord) : DWord;
stdcall;
```

Parameters

[in/out] Name	[Init Value] Explanation
[>>]lOffset	전체 출력 접점에 대한 Offset 위치
[<<]upValue	출력 접점 값 (Boolean)

Return Values

[0000] AXT_RT_SUCCESS : AXL 라이브러리 초기화 성공

[3051] AXT_RT_DIO_NOT_MODULE : DIO 모듈 없음

[3102] AXT_RT_DIO_INVALID_OFFSET_NO : 유효하지않는 DIO OFFSET 번호

* See error code Table for more information on status error codes

Description

전체 출력 접점 모듈의 Offset 위치에서 bit 단위로 데이터를 읽는다.

Offset은 '0'부터 총 입력 접점 수 - 1 까지 사용할 수 있다.

Example

C++

```
// 전체 출력 모듈의 offset 0번지에서 bit 단위로 데이터를 읽는다.
DWORD dwValue;

AxdoReadOutport(0, &dwValue);
```

Visual Basic

```
' 전체 출력 모듈의 offset 0번지에서 bit 단위로 데이터를 읽는다.
Dim lValue As Long

AxdoReadOutport 0, lValue
```

Delphi

```
{ 전체 출력 모듈의 offset 0번지에서 bit 단위로 데이터를 읽는다. }
var
    dwValue : DWord;

begin
    AxdoReadOutport(0, @dwValue);
end;
```

See Also

[AxdoReadOutportBit](#), [AxdoReadOutportByte](#), [AxdoReadOutportWord](#), [AxdoReadOutportDword](#),
[AxdoWriteOutport](#)

고급함수

Function	Description
AxdilsPulseOn	지정한 입력 접점 모듈의 Offset 위치에서 신호가 Off에서 On으로 바뀌었는지 확인한다.
AxdilsPulseOff	지정한 입력 접점 모듈의 Offset 위치에서 신호가 On에서 Off으로 바뀌었는지 확인한다.
AxdilsOn	지정한 입력 접점 모듈의 Offset 위치에서 신호가 count 만큼 호출될 동안 On 상태로 유지하는지 확인한다.
AxdilsOff	지정한 입력 접점 모듈의 Offset 위치에서 신호가 count 만큼 호출될 동안 Off 상태로 유지하는지 확인한다.
AxdoOutPulseOn	지정한 출력 접점 모듈의 Offset 위치에서 설정한 mSec동안 On을 유지하다가 Off 시킨다.
AxdoOutPulseOff	지정한 출력 접점 모듈의 Offset 위치에서 설정한 mSec동안 Off를 유지하다가 On 시킨다.
AxdoToggleStart	지정한 출력 접점 모듈의 Offset 위치에서 설정한 횟수, 설정한 간격으로 토글한 후 원래의 출력상태를 유지한다.
AxdoToggleStop	지정한 출력 접점 모듈의 Offset 위치에서 토글중인 출력을 설정한 신호 상태로 정지 시킨다.
AxdoSetNetworkErrorAct	지정한 출력 모듈의 Network이 끊어 졌을 경우 출력 상태를 출력 Byte 단위로 설정한다.
AxdSelContactNum	Mechatrolink-II 제어기의 Simple IO Type DIO 모듈의 Input/Output 접점 개수를 설정한다.
AxdGetContactNum	Mechatrolink-II 제어기의 Simple IO Type DIO 모듈의 Input/Output 접점 개수를 반환한다.

AxdiIsPulseOn

Purpose

지정한 입력 접점 모듈의 Offset 위치에서 신호가 Off에서 On으로 바뀌었는지 확인한다.

Format

C++

```
DWORD AxdiIsPulseOn(long lModuleNo, long lOffset, DWORD *upValue);
```

Visual Basic

```
Function AxdiIsPulseOn(ByVal lModuleNo As Long, ByVal lOffset As Long,  
ByRef upValue As Long) As Long
```

Delphi

```
function AxdiIsPulseOn(lModuleNo : LongInt; lOffset : LongInt;  
upValue : PWord) : Word; stdcall;
```

Parameters

[in/out] Name	[Init Value] Explanation
[>>]lModuleNo	모듈 번호
[>>]lOffset	입력 접점에 대한 Offset 위치
[<<]upValue	On 상태 여부 (Boolean)

Return Values

[0000] AXT_RT_SUCCESS : AXL 라이브러리 초기화 성공

[3051] AXT_RT_DIO_NOT_MODULE : DIO 모듈 없음

[3101] AXT_RT_DIO_INVALID_MODULE_NO : 유효하지않는 DIO 모듈 번호

[3102] AXT_RT_DIO_INVALID_OFFSET_NO : 유효하지않는 DIO OFFSET 번호

* See error code Table for more information on status error codes

Description

사용자가 지정한 입력 접점 모듈의 Offset 위치에서 신호가 Off에서 On으로 바뀌었는지 확인한다.

지정한 모듈이 어떤 모듈인지 확인하여 아래의 Table에서 Offset 사용 범위를 참조하여 읽는다.

Offset 범위와 Return 값

Module	Offset	Value
SIO-DI32	0 ~ 31	0(False), 1(True)
SIO-DB32x	0 ~ 15	0(False), 1(True)

Example

C++

```
// 0번째 모듈의 offset 0번지에서 신호가 off에서 on으로 바뀌었는지 확인한다.  
DWORD dwValue;
```

```
AxdiIsPulseOn(0, 0, &dwValue);
```

Visual Basic

' 0번째 모듈의 Offset 0번지에서 신호가 off에서 on으로 바뀌었는지 확인한다.

```
Dim lValue As Long
```

```
AxdiIsPulseOn 0, 0, lValue
```

Delphi

```
{ 0번째 모듈의 Offset 0번지에서 신호가 off에서 on으로 바뀌었는지 확인한다. }
```

```
var
```

```
    dwValue : DWord;
```

```
begin
```

```
    AxdiIsPulseOn(0, 0, @dwValue);
```

```
end;
```

See Also

[AxdiIsPulseOff](#), [AxdiIsOn](#), [AxdiIsOff](#), [AxdoOutPulseOn](#), [AxdoOutPulseOff](#), [AxdoToggleStart](#),
[AxdoToggleStop](#)

AxdiIsPulseOff

Purpose

지정한 입력 접점 모듈의 Offset 위치에서 신호가 On에서 Off로 바뀌었는지 확인한다.

Format

C++

```
DWORD AxdiIsPulseOff(long lModuleNo, long lOffset, DWORD *upValue);
```

Visual Basic

```
Function AxdiIsPulseOff(ByVal lModuleNo As Long, ByVal lOffset As Long, ByRef upValue As Long) As Long
```

Delphi

```
function AxdiIsPulseOff(lModuleNo : LongInt; lOffset : LongInt;
  upValue : PWord) : Word; stdcall;
```

Parameters

[in/out] Name	[Init Value] Explanation
[>>]lModuleNo	모듈 번호
[>>]lOffset	입력 접점에 대한 Offset 위치
[<<]upValue	Off 상태 여부 (Boolean)

Return Values

[0000] AXT_RT_SUCCESS : AXL 라이브러리 초기화 성공

[3051] AXT_RT_DIO_NOT_MODULE : DIO 모듈 없음

[3101] AXT_RT_DIO_INVALID_MODULE_NO : 유효하지않는 DIO 모듈 번호

[3102] AXT_RT_DIO_INVALID_OFFSET_NO : 유효하지않는 DIO OFFSET 번호

* See error code Table for more information on status error codes

Description

사용자가 지정한 입력 접점 모듈의 Offset 위치에서 신호가 On에서 Off로 바뀌었는지 확인한다.

지정한 모듈이 어떤 모듈인지 확인하여 아래의 Table에서 Offset 사용 범위를 참조하여 읽는다.

Offset 범위와 Return 값

Module	Offset	Value
SIO-DI32	0 ~ 31	0(False), 1(True)
SIO-DB32x	0 ~ 15	0(False), 1(True)

Example

C++

```
// 0번째 모듈의 offset 0번지에서 신호가 on에서 off로 바뀌었는지 확인한다.
```

```
DWORD dwValue;
```

```
AxdiIsPulseOff(0, 0, &dwValue);
```

Visual Basic

‘ 0번째 모듈의 Offset 0번지에서 신호가 on에서 off으로 바뀌었는지 확인한다.

```
Dim lValue As Long
```

```
AxdiIsPulseOff 0, 0, lValue
```

Delphi

```
{ 0번째 모듈의 Offset 0번지에서 신호가 on에서 off으로 바뀌었는지 확인한다. }
```

```
var  
  dwValue : DWord;  
  
begin  
  AxdiIsPulseOff(0, 0, @dwValue);  
end;
```

See Also

[AxdiIsPulseOn](#), [AxdiIsOn](#), [AxdiIsOff](#), [AxdoOutPulseOn](#), [AxdoOutPulseOff](#), [AxdoToggleStart](#),
[AxdoToggleStop](#)

AxdiIsOn

Purpose

지정한 입력 접점 모듈의 Offset 위치에서 신호가 count 만큼 호출될 동안 On 상태로 유지하는지 확인한다.

Format

C++

```
DWORD AxdiIsOn(long lModuleNo, long lOffset, long lCount, DWORD
*upValue, long lStart);
```

Visual Basic

```
Function AxdiIsOn(ByVal lModuleNo As Long, ByVal lOffset As Long,
ByVal lCount As Long, ByRef upValue As Long, ByVal lStart As Long) As
Long
```

Delphi

```
function AxdiIsOn(lModuleNo : LongInt; lOffset : LongInt; lCount :
LongInt; upValue : PDWord; lStart : LongInt) : DWord; stdcall;
```

Parameters

[in/out] Name	[Init Value] Explanation
[>>]lModuleNo	모듈 번호
[>>]lOffset	입력 접점에 대한 Offset 위치
[>>]lCount	호출 횟수
[>>]lStart	신호 유지 확인 시작 (최초 호출 : 1, 반복 호출 : 0)
[<<]upValue	On 상태 유지 여부 (Boolean)

Return Values

[0000] AXT_RT_SUCCESS : AXL 라이브러리 초기화 성공

[3051] AXT_RT_DIO_NOT_MODULE : DIO 모듈 없음

[3101] AXT_RT_DIO_INVALID_MODULE_NO : 유효하지않는 DIO 모듈 번호

[3102] AXT_RT_DIO_INVALID_OFFSET_NO : 유효하지않는 DIO OFFSET 번호

* See error code Table for more information on status error codes

Description

사용자가 지정한 입력 접점 모듈의 Offset 위치에서 신호가 count 만큼 호출될 동안 On 상태로 유지하는지 확인한다. 최초 호출에서 lStart 를 1 로 하고 이후 count-1 호출 동안은 0 으로 설정해야 한다. upValue 반환값은 count - 1 동안의 호출에 대해서는 무조건 FALSE 를 반환하며 count 번째 호출부터 신호가 이전 count 동안 On 상태가 유지가 되었다면 TRUE 를 반환하고 아니면 FALSE 를 반환 한다. 즉, count 번째 호출부터의 반환값이 신호유지 여부에 대한 판단을 할 수 있는 반환 값이 된다. 예를 들어, lCount 를 10 으로 설정하고 On 신호가 계속 유지된다면, 10 회 호출 이후 새롭게 count 를 0 으로 초기화 하거나 lStart 를 1 로 설정하여 새롭게 시작하지 않는 이상 On 신호가 계속 유지되어 있으므로 반환값은 TRUE 를 반환하게 된다.

호출 횟수 count 를 0 으로 설정하면 신호가 On 상태인지 확인하는 용도로 쓰일 수 있으며 또한 이전에 설정된 호출 횟수를 초기화하는 용도로 사용 될 수 있다.

지정한 모듈이 어떤 모듈인지 확인하여 아래의 Table 에서 Offset 사용 범위를 참조하여 읽는다.

Offset 범위와 Return 값

Module	Offset	Value
SIO-DI32	0 ~ 31	0(FALSE), 1(TRUE)
SIO-DB32x	0 ~ 15	0(FALSE), 1(TRUE)

Example

C++

```
// 0번지 모듈의 Offset 0번지에서 신호가 10 (Count) 회 만큼 호출될 동안 On 상태로 유지하는지 확인한다.
// 예제 1
short i;
DWORD dwValue;
AxdiIsOn(0, 0, 10, &dwValue, 1); // 최초 호출, 10회 신호 유지 상태 점검 시작 lStart:1
for (i = 0; i < 9;) // 위 최초 호출 포함하여 10번 호출 동안 On 상태 유지하는지 체크
{
    // 실제 프로그램에서는 Timer 나 Thread를 통한 일정 주기를 가진 호출 루틴이 사용
    AxdiIsOn(0, 0, 10, &dwValue, 0); // 이후 count 만큼의 반복 호출
}

if (dwValue != 0)
    AfxMessageBox("ON 상태로 유지 되었습니다.");

// 예제 2(위 코드와 동일하나 아래 코드가 더욱 명확함)
short i;
DWORD dwValue;
AxdiIsOn(0, 0, 0, &dwValue, 1); // lCount 가 0이므로 이전 Count 초기화됨.
if(dwValue !=0) // 현재 신호 상태가 On
{
    for (i = 0; i < 10; i++) // 10번 호출동안 On 상태 유지하는지 체크
    {
        // 실제 프로그램에서는 Timer 나 Thread를 통한 일정 주기를 가진 호출 루틴이 사용
        AxdiIsOn(0, 0, 10, &dwValue, 0); // 새로운 count 적용하여 반복 호출
    }
    if (dwValue != 0)
        AfxMessageBox("ON 상태로 유지 되었습니다.");
}
```

Visual Basic

```
' 0번지 모듈의 Offset 0번지에서 신호가 10 (Count) 회 만큼 호출될 동안 On 상태로 유지하는지 확인한다.
Dim i As Integer
Dim lValue As Long

AxdiIsOn 0, 0, 10, lValue, 1 // 최초 호출 10번 Count 신호 유지 상태 점검 시작

For I = 0 To 9
    AxdiIsOn 0, 0, 10, lValue, 0
```

Next

```
If lValue = 1 Then  
    MsgBox "ON 상태로 유지 되었습니다."  
End If
```

Delphi

```
{ 0번째 모듈의 Offset 0번지에서 신호가 10 (Count) 회 만큼 호출될 동안 On 상태로 유지하는지 확인한다. }  
var  
    i : SmallInt;  
    dwValue : DWord;  
  
begin  
    AxdiIsOn(0, 0, 10, @dwValue, 1);  
    for i := 0 to 9 do  
        begin  
            AxdiIsOn(0, 0, 10, @dwValue, 0);  
        end;  
    end;
```

See Also

[AxdiIsPulseOn](#), [AxdiIsPulseOff](#), [AxdiIsOff](#), [AxdoOutPulseOn](#), [AxdoOutPulseOff](#), [AxdoToggleStart](#),
[AxdoToggleStop](#)

AxdiIsOff

Purpose

지정한 입력 접점 모듈의 Offset 위치에서 신호가 count 만큼 호출될 동안 Off 상태로 유지하는지 확인한다.

Format

C++

```
DWORD AxdiIsOff(long lModuleNo, long lOffset, long lCount, DWORD
    *upValue, long lStart);
```

Visual Basic

```
Function AxdiIsOff(ByVal lModuleNo As Long, ByVal lOffset As Long,
    ByVal lCount As Long, ByRef upValue As Long, ByVal lStart As Long) As
    Long
```

Delphi

```
function AxdiIsOff(lModuleNo : LongInt; lOffset : LongInt; lCount :
    LongInt; upValue : PDWord; lStart : LongInt) : DWord; stdcall;
```

Parameters

[in/out] Name	[Init Value] Explanation
[>>]lModuleNo	모듈 번호
[>>]lOffset	입력 접점에 대한 Offset 위치
[>>]lCount	호출 횟수
[>>]lStart	신호 유지 확인 시작 (최초 호출 : 1, 반복 호출 : 0)
[<<]upValue	Off 상태 유지 여부 (Boolean)

Return Values

[0000] AXT_RT_SUCCESS : AXL 라이브러리 초기화 성공

[3051] AXT_RT_DIO_NOT_MODULE : DIO 모듈 없음

[3101] AXT_RT_DIO_INVALID_MODULE_NO : 유효하지않는 DIO 모듈 번호

[3102] AXT_RT_DIO_INVALID_OFFSET_NO : 유효하지않는 DIO OFFSET 번호

* See error code Table for more information on status error codes

Description

사용자가 지정한 입력 접점 모듈의 Offset 위치에서 신호가 count 만큼 호출될 동안 Off 상태로 유지하는지 확인한다. 최초 호출에서 lStart 를 1 로 하고 이후 count - 1 호출 동안은 '0'으로 셋팅해야 한다. upValue 반환값은 count - 1 동안의 호출에 대해서는 무조건 FALSE 를 반환하며 count 번째 호출부터 신호가 이전 count 동안 Off 상태가 유지가 되었다면 TRUE 를 반환하고 아니면 FALSE 를 계속 반환한다. 즉, count 번째 호출부터의 반환값이 신호유지 여부에 대한 판단을 할 수 있는 반환값이 된다. 예를 들어, lCount 를 '10'으로 셋팅하고 Off 신호가 계속 유지된다면, '10'회 호출 이후 새롭게 count 를 '0'으로 초기화 하거나 lStart 를 '1'로 셋팅하여 새롭게 시작하지 않는 이상 Off 신호가 계속 유지되어 있으므로 반환값은 TRUE 를 반환하게 된다.

호출 횟수 count 를 ‘0’으로 설정하면 신호가 Off 상태인지 확인하는 용도로 쓰일 수 있으며 또한 이전에 셋팅된 호출 횟수를 초기화하는 용도로도 사용 될 수 있다.

지정한 모듈이 어떤 모듈인지 확인하여 아래의 Table에서 Offset 사용 범위를 참조하여 읽는다.

Offset 범위와 Return 값

Module	Offset	Value
SIO-DI32	0 ~ 31	0(FALSE), 1(TRUE)
SIO-DB32x	0 ~ 15	0(FALSE), 1(TRUE)

Example

C++

```
// 0번지 모듈의 Offset 0번지에서 신호가 10 (Count) 회 만큼 호출될 동안 off 상태로 유지하는지 확인한다.
// 예제 1
short i;
DWORD dwValue;
AxdiIsOff(0,0,0,10,&dwValue,1);      // 최초 호출, 10회 신호유지상태 점검 시작 lStart:1
for (i = 0; i < 9;)                // 위 최초 호출 포함하여 10번 호출 동안 off 상태 유지하는지 체크
{
    // 실제 프로그램에서는 Timer 나 Thread를 통한 일정 주기를 가진 호출 루틴이 사용
    AxdiIsOff(0, 0, 10, &dwValue, 0); // 이후 count 만큼의 반복 호출
}

if (dwValue == 1)
    AfxMessageBox("OFF 상태로 유지 되었습니다.");

// 예제 2(위 코드와 동일하나 아래 코드가 더욱 명확함)
short i;
DWORD dwValue;
AxdiIsOff(0, 0, 0, &dwValue, 1);    // lCount 가 0이므로 이전 Count 초기화됨.
if(dwValue == 1)                  // 현재 신호 상태가 off
{
    for (i = 0; i < 10; i++)      // 10번 호출동안 off 상태 유지하는지 체크
    {
        // 실제 프로그램에서는 Timer 나 Thread를 통한 일정 주기를 가진 호출 루틴이 사용
        AxdiIsOff(0, 0, 10, &dwValue, 0); // 새로운 count 적용하여 반복 호출
    }
    if (dwValue == 1)
        AfxMessageBox("OFF 상태로 유지 되었습니다.");
}
```

Visual Basic

```
' 0번지 모듈의 Offset 0번지에서 신호가 10 (Count) 회 만큼 호출될 동안 off 상태로 유지하는지 확인한다.
Dim i As Integer
Dim lValue As Long

AxdiIsOff 0, 0, 10, lValue, 1    ' 최초 호출 10번 Count 신호 유지 상태 점검 시작

For I = 0 To 9
    AxdiIsOff 0, 0, 10, lValue, 0
```

```
Next
```

```
If lValue = 1 Then  
    MsgBox "OFF 상태로 유지 되었습니다."  
End If
```

Delphi

```
{ 0번째 모듈의 Offset 0번지에서 신호가 10 (Count) 회 만큼 호출될 동안 off 상태로 유지하는지 확인한다. }  
var  
    i : SmallInt;  
    dwValue : DWord;  
  
begin  
    AxdiIsOff(0, 0, 10, @dwValue, 1);  
    for i := 0 to 9 do  
        begin  
            AxdiIsOff(0, 0, 10, @dwValue, 0);  
        end;  
  
    if(dwValue = 1) Then  
        Application.MessageBox('OFF 상태로 유지 되었습니다.', 'Ajinextek', MB_OK);  
end;
```

See Also

[AxdiIsPulseOn](#), [AxdiIsPulseOff](#), [AxdiIsOn](#), [AxdoOutPulseOn](#), [AxdoOutPulseOff](#), [AxdoToggleStart](#),
[AxdoToggleStop](#)

AxdoOutPulseOn

Purpose

지정한 출력 접점 모듈의 Offset 위치에서 설정한 mSec 동안 On을 유지하다가 Off 시킨다

Format

C++

```
DWORD AxdoOutPulseOn(long lModuleNo, long lOffset, long lmSec);
```

Visual Basic

```
Function AxdoOutPulseOn(ByVal lModuleNo As Long, ByVal lOffset As Long, ByVal lmSec As Long) As Long
```

Delphi

```
function AxdoOutPulseOn(lModuleNo : LongInt; lOffset : LongInt;
lmSec : LongInt) : DWord; stdcall;
```

Parameters

[in/out] Name	[Init Value] Explanation
[>>]lModuleNo	모듈 번호
[>>]lOffset	입력 접점에 대한 Offset 위치
[>>]lmSec	유지 시간 (1 ~ 30000 밀리초)

Return Values

[0000] AXT_RT_SUCCESS : AXL 라이브러리 초기화 성공

[3051] AXT_RT_DIO_NOT_MODULE : DIO 모듈 없음

[3101] AXT_RT_DIO_INVALID_MODULE_NO : 유효하지않는 DIO 모듈 번호

[3102] AXT_RT_DIO_INVALID_OFFSET_NO : 유효하지않는 DIO OFFSET 번호

* See error code Table for more information on status error codes

Description

사용자가 지정한 출력 접점 모듈의 Offset 위치에서 설정한 mSec 동안 On을 유지하다가 Off 시킨다.

지정한 모듈이 어떤 모듈인지 확인하여 아래의 Table에서 Offset 사용 범위를 참조하여 읽는다.

Offset 범위와 Return 값

Module	Offset	Value
SIO-DI32	0 ~ 31	0(FALSE), 1(TRUE)
SIO-DB32x	0 ~ 15	0(FALSE), 1(TRUE)

Example

C++

```
// 0번재 모듈의 offset 0번지에서 10초 동안 On을 유지하다가 off 시킨다.
```

```
AxdoOutPulseOn(0, 0, 10000);
```

Visual Basic

' 0번째 모듈의 Offset 0번지에서 10초 동안 On을 유지하다가 off 시킨다.
AxdoOutPulseOn 0, 0, 10000

Delphi

{ 0번째 모듈의 Offset 0번지에서 10초 동안 On을 유지하다가 off 시킨다. }
AxdoOutPulseOn(0, 0, 10000);

See Also

[AxdiIsPulseOn](#), [AxdiIsPulseOff](#), [AxdiIsOn](#), [AxdiIsOff](#), [AxdoOutPulseOff](#), [AxdoToggleStart](#),
[AxdoToggleStop](#)

AxdoOutPulseOff

Purpose

지정한 출력 접점 모듈의 Offset 위치에서 설정한 mSec 동안 Off 를 유지하다가 On 시킨다

Format

C++

```
DWORD AxdoOutPulseOff(long lModuleNo, long lOffset, long lmSec);
```

Visual Basic

```
Function AxdoOutPulseOff(ByVal lModuleNo As Long, ByVal lOffset As Long, ByVal lmSec As Long) As Long
```

Delphi

```
function AxdoOutPulseOff(lModuleNo : LongInt; lOffset : LongInt;
lmSec : LongInt) : DWord; stdcall;
```

Parameters

[in/out] Name	[Init Value] Explanation
[>>]lModuleNo	모듈 번호
[>>]lOffset	입력 접점에 대한 Offset 위치
[>>]lmSec	유지 시간 (1 ~ 30000 밀리초)

Return Values

[0000] AXT_RT_SUCCESS : AXL 라이브러리 초기화 성공

[3051] AXT_RT_DIO_NOT_MODULE : DIO 모듈 없음

[3101] AXT_RT_DIO_INVALID_MODULE_NO : 유효하지않는 DIO 모듈 번호

[3102] AXT_RT_DIO_INVALID_OFFSET_NO : 유효하지않는 DIO OFFSET 번호

* See error code Table for more information on status error codes

Description

사용자가 지정한 출력 접점 모듈의 Offset 위치에서 설정한 mSec 동안 Off 를 유지하다가 On 시킨다.

지정한 모듈이 어떤 모듈인지 확인하여 아래의 Table에서 Offset 사용 범위를 참조하여 읽는다.

Offset 범위와 Return 값

Module	Offset	Value
SIO-DI32	0 ~ 31	0(FALSE), 1(TRUE)
SIO-DB32x	0 ~ 15	0(FALSE), 1(TRUE)

Example

C++

```
// 0번재 모듈의 offset 0번지에서 10초 동안 off를 유지하다가 on 시킨다.  
AxdoOutPulseOff(0, 0, 10000);
```

Visual Basic

```
' 0번째 모듈의 Offset 0번지에서 10초 동안 off를 유지하다가 on 시킨다.  
AxdoOutPulseOff 0, 0, 10000
```

Delphi

```
{ 0번째 모듈의 Offset 0번지에서 10초 동안 off를 유지하다가 on 시킨다. }  
AxdoOutPulseOff(0, 0, 10000);
```

See Also

[AxdiIsPulseOn](#), [AxdiIsPulseOff](#), [AxdiIsOn](#), [AxdiIsOff](#), [AxdoOutPulseOn](#), [AxdoToggleStart](#),
[AxdoToggleStop](#)

AxdoToggleStart

Purpose

지정한 출력 접점 모듈의 Offset 위치에서 최초 시작 신호 상태, 설정한 횟수, 설정한 On/Off 시간 간격으로 토글한 후 원래의 출력상태를 유지한다. 설정 횟수를 -1로 지정하면 무한 반복을 한다.

Format

C++

```
DWORD AxdoToggleStart(long lModuleNo, long lOffset, long lInitState,
                      long lmSecOn, long lmSecOff, long lCount);
```

Visual Basic

```
Function AxdoToggleStart(ByVal lModuleNo As Long, ByVal lOffset As
                           Long, ByVal lInitState As Long, ByVal lmSecOn, ByVal lmSecOff As Long,
                           ByVal lCount As Long) As Long
```

Delphi

```
function AxdoToggleStart(lModuleNo : LongInt; lOffset : LongInt;
                        lInitState : LongInt; lmSecOn : LongInt; lmSecOff : LongInt; lCount :
                        LongInt) : DWord; stdcall;
```

Parameters

[in/out] Name	[Init Value] Explanation
[>>]lModuleNo	모듈 번호
[>>]lOffset	입력 접점에 대한 Offset 위치
[>>]lInitState	최초 시작 신호 상태 설정(Off : 0, On: 1)
[>>]lmSecOn	토글 간격 중 On Time 시간 간격 (1 ~ 30000 밀리초)
[>>]lmSecOff	토글 간격 중 Off Time 시간 간격 (1 ~ 30000 밀리초)
[>>]lCount	토글 횟수, -1의 경우 무한 토글

Return Values

[0000] AXT_RT_SUCCESS : AXL 라이브러리 초기화 성공

[3051] AXT_RT_DIO_NOT_MODULE : DIO 모듈 없음

[3101] AXT_RT_DIO_INVALID_MODULE_NO : 유효하지않는 DIO 모듈 번호

[3102] AXT_RT_DIO_INVALID_OFFSET_NO : 유효하지않는 DIO OFFSET 번호

* See error code Table for more information on status error codes

Description

사용자가 지정한 출력 접점 모듈의 Offset 위치에서 초기 신호 상태, 설정한 횟수, 설정한 간격으로 토글한 후 원래의 출력상태를 유지한다.

지정한 모듈이 어떤 모듈인지 확인하여 아래의 Table에서 Offset 사용 범위를 참조하여 읽는다.

Offset 범위와 Return 값

Module	Offset	Value
SIO-DI32	0 ~ 31	0(FALSE), 1(TRUE)
SIO-DB32x	0 ~ 15	0(FALSE), 1(TRUE)

Example

C++

```
// 0번째 모듈의 Offset 0번지에서 초기 off 상태로 시작 10회 동안 0.5/1초 간격으로 토글한다.
AxdoToggleStart(0, 0, 0, 500, 1000, 10);
```

Visual Basic

```
' 0번째 모듈의 Offset 0번지에서 초기 off 상태로 시작 10회 동안 0.5/1초 간격으로 토글한다.
AxdoToggleStart 0, 0, 0, 500, 1000, 10
```

Delphi

```
{ 0번째 모듈의 Offset 0번지에서 초기 off 상태로 시작 10회 동안 0.5/1초 간격으로 토글한다. }
AxdoToggleStart(0, 0, 0, 500, 1000, 10);
```

See Also

[AxdiIsPulseOn](#), [AxdiIsPulseOff](#), [AxdiIsOn](#), [AxdiIsOff](#), [AxdoOutPulseOn](#), [AxdoOutPulseOff](#),
[AxdoToggleStop](#)

AxdoToggleStop

Purpose

지정한 출력 접점 모듈의 Offset 위치에서 토글중인 출력을 설정한 신호 상태로 정지 시킨다.

Format

C++

```
DWORD AxdoToggleStop(long lModuleNo, long lOffset, DWORD uOnOff);
```

Visual Basic

```
Function AxdoToggleStop(ByVal lModuleNo As Long, ByVal lOffset As Long, ByVal uOnOff As Long) As Long
```

Delphi

```
function AxdoToggleStop(lModuleNo : LongInt; lOffset : LongInt;
uOnOff : DWord) : DWord; stdcall;
```

Parameters

[in/out] Name	[Init Value] Explanation
[>>]lModuleNo	모듈 번호
[>>]lOffset	입력 접점에 대한 Offset 위치
[>>]uOnOff	출력 접점 값 (0 ~ 1)

Return Values

[0000] AXT_RT_SUCCESS : AXL 라이브러리 초기화 성공

[3051] AXT_RT_DIO_NOT_MODULE : DIO 모듈 없음

[3101] AXT_RT_DIO_INVALID_MODULE_NO : 유효하지않는 DIO 모듈 번호

[3102] AXT_RT_DIO_INVALID_OFFSET_NO : 유효하지않는 DIO OFFSET 번호

* See error code Table for more information on status error codes

Description

사용자가 지정한 출력 접점 모듈의 Offset 위치에서 토글중인 출력을 설정한 신호 상태로 정지 시킨다.

지정한 모듈이 어떤 모듈인지 확인하여 아래의 Table에서 Offset 사용 범위 참조하여 읽는다.

Offset 범위와 Return 값

Module	Offset	Value
SIO-DI32	0 ~ 31	0(FALSE), 1(TRUE)
SIO-DB32x	0 ~ 15	0(FALSE), 1(TRUE)

Example

C++

```
// 0번재 모듈의 offset 0번지에서 토글을 정지하고 신호 상태를 OFF 시킨다.
```

```
AxdoToggleStop(0, 0, OFF);
```

Visual Basic

‘ 0번째 모듈의 Offset 0번지에서 토글을 정지하고 신호 상태를 OFF 시킨다.
AxdoToggleStop 0, 0, 0t

Delphi

{ 0번째 모듈의 Offset 0번지에서 토글을 정지하고 신호 상태를 OFF 시킨다. }
AxdoToggleStop(0, 0, OFF)

See Also

[AxdiIsPulseOn](#), [AxdiIsPulseOff](#), [AxdiIsOn](#), [AxdiIsOff](#), [AxdoOutPulseOn](#), [AxdoOutPulseOff](#),
[AxdoToggleStart](#)

AxdoSetNetworkErrorAct

Purpose

지정한 출력 모듈의 Network 0이 끊어 졌을 경우 출력 상태를 출력 Byte 단위로 설정한다.

주의사항 : PCI-R1604-RTEX 와 연결된 DO 출력 Slave Node 에서 사용되는 함수입니다.

Format

C++

```
DWORD AxdoSetNetworkErrorAct (long lModuleNo, DWORD dwSize, DWORD
 *dwaSetValue);
```

C#

```
uint AxdoSetNetworkErrorAct (long lModuleNo, uint dwSize, ref uint
 dwaSetValue);
```

Visual Basic

```
Function AxdoSetNetworkErrorAct (ByVal lModuleNo As Long, ByVal
 dwSize As Long, ByRef dwaSetValue As Long) As Long
```

Delphi

```
function AxdoSetNetworkErrorAct (lModuleNo : LongInt; dwSize: LongInt;
 dwaSetValue: PDWord) : DWord; stdcall;
```

Parameters

[in/out] Name	[Init Value] Explanation
[>>]lModuleNo	모듈 번호
[>>]dwSize	설정 할 Byte 수
[>>]dwaSetValue	설정 할 변수 값 (Default는 Network 끊어지기 전 상태 유지) -[00h] Network 끊어지기 전 상태 유지 -[01h] On -[02h] Off

Return Values

[0000] AXT_RT_SUCCESS : AXL 라이브러리 초기화 성공

[3051] AXT_RT_DIO_NOT_MODULE : DIO 모듈 없음

[1181] AXT_RT_3RD_ABOVE_MAX_VALUE : 13 번째 인자값이 최대 값 보다 큼

* See error code Table for more information on status error codes

Description

사용자가 지정한 출력 모듈의 Network 0이 끊어 졌을 경우 출력 상태를 출력 Byte 단위로 설정하는 함수이다. Size 의 경우 Byte 단위로 값을 정해야 하며 예를 들면 RTEX-DB32 의 경우 Size 는 2이며 RTEX-DO32 의 경우 4가 입력 되게 된다.

Example

C++

```
// Network이 끊어지면 0번 Node의 RTEX-DB32의 모든 출력 접점을 Off 시킨다.  
DWORD dwaSetValue[2] = {0x02, 0x02};  
AxdoSetNetworkErrorAct(0, 2, dwaSetValue);
```

C#

```
// Network이 끊어지면 0번 Node의 RTEX-DB32의 모든 출력 접점을 Off 시킨다.  
DWORD dwaSetValue[2] = {0x02, 0x02};  
CAXA.AxdoSetNetworkErrorAct(0, 2, ref dwaSetValue);
```

See Also

[AxdiIsPulseOn](#), [AxdiIsPulseOff](#), [AxdiIsOn](#), [AxdiIsOff](#), [AxdoOutPulseOn](#), [AxdoOutPulseOff](#),
[AxdoToggleStart](#)

AxdSetContactNum

Purpose

Mechatrolink-II 의 Simple IO Type 의 Input/Output 접점 개수를 설정한다.

주의사항 : PCI-R1604-MLI와 연결된 Simple IO Type 출력 Slave Node에서 사용되는 함수입니다.

Format

C++

```
DWORD AxdSetContacNum(long lModuleNo, DWORD dwInputNum, DWORD dwOutputNum);
```

C#

```
uint AxdSetContacNum (long lModuleNo, uint dwInputNum, uint dwOutputNum);
```

Visual Basic

```
Function AxdSetContacNum (ByVal lModuleNo As Long, ByVal dwInputNum As Long, ByVal dwOutputNum As Long) As Long
```

Delphi

```
function AxdSetContacNum (lModuleNo : LongInt; dwInputNum: DWord; dwOutputNum: DWord) : DWord; stdcall;
```

Parameters

[in/out] Name	[Init Value] Explanation
[>>]lModuleNo	모듈 번호
[>>]dwInputNum	설정 할 Input 접점의 개수
[>>]dwOutputNum	설정 할 Output 접점의 개수

Return Values

[0000] AXT_RT_SUCCESS : AXL 라이브러리 초기화 성공

[3051] AXT_RT_DIO_NOT_MODULE : DIO 모듈 없음

[1171] AXT_RT_2ND_ABOVE_MAX_VALUE : 2 번째 인자값이 최대 값 보다 큼

[1181] AXT_RT_3RD_ABOVE_MAX_VALUE : 3 번째 인자값이 최대 값 보다 큼

* See error code Table for more information on status error codes

Description

Mechatrolink-II 제어기의 Simple IO Type 제품에서 사용되는 함수로 Simple IO Type 의 경우 상위 제어기에서 접점의 개수를 알 수 없으므로 사용자가 시스템의 I/O 개수를 설정 해야 한다.

Example

C++

```
// 시스템에 연결된 0번 모듈의 MLII Simple IO 접점의 개수를 In:128, Out:128로 설정 합니다.  
AxdSetContactNum(0, 128, 128);
```

C#

```
// 시스템에 연결된 0번 모듈의 MLII Simple IO 접점의 개수를 In:128, Out:128로 설정 합니다.  
CAXA.AxdSetContactNum (0, 128, 128);
```

Visual Basic

```
'시스템에 연결된 0번 모듈의 MLII Simple IO 접점의 개수를 In:128, Out:128로 설정 합니다.  
AxdSetContactNum 0, 128, 128
```

Delphi

```
{ 시스템에 연결된 0번 모듈의 MLII Simple IO 접점의 개수를 In:128, Out:128로 설정 합니다. }  
AxdSetContactNum (0, 128, 128)
```

See Also

[AxdIsPulseOn](#), [AxdIsPulseOff](#), [AxdIsOn](#), [AxdIsOff](#), [AxdoOutPulseOn](#), [AxdoOutPulseOff](#),
[AxdoToggleStart](#), [AxdGetContactNum](#)

AxdGetContactNum

Purpose

Mechatrolink-II 의 Simple IO Type 의 Input/Output 접점 개수를 반환한다.

주의사항 : PCI-R1604-MLI와 연결된 Simple IO Type 출력 Slave Node에서 사용되는 함수입니다.

Format

C++

```
DWORD AxdGetContacNum(long lModuleNo, DWORD dwpInputNum, DWORD
dwpOutputNum);
```

C#

```
uint AxdGetContacNum (long lModuleNo, ref uint dwpInputNum, ref uint
dwpOutputNum);
```

Visual Basic

```
Function AxdGetContacNum (ByVal lModuleNo As Long, ByRef dwpInputNum
As Long, ByRef dwpOutputNum As Long) As Long
```

Delphi

```
function AxdGetContacNum (lModuleNo : LongInt; dwpInputNum : PDWord;
dwpOutputNum : PDWord) : DWord; stdcall;
```

Parameters

[in/out] Name	[Init Value] Explanation
[>>]lModuleNo	모듈 번호
[>>]dwplnputNum	설정 된 Input 접점의 개수
[>>]dwpOutputNum	설정 된 Output 접점의 개수

Return Values

[0000] AXT_RT_SUCCESS : AXL 라이브러리 초기화 성공

[3051] AXT_RT_DIO_NOT_MODULE : DIO 모듈 없음

* See error code Table for more information on status error codes

Description

Mechatrolink-II 제어기의 Simple IO Type 제품에서 사용되는 함수로 Simple IO Type 의 I/O 접점 개수를 반환 한다.

Example

C++

```
// 시스템에 연결된 0번 모듈의 MLII Simple IO 접점의 개수를 반환 합니다.  
DWORD dwInput, dwOutput;  
AxdGetContactNum(0, &dwInput, &dwOutput);
```

C#

```
// 시스템에 연결된 0번 모듈의 MLII Simple IO 접점의 개수를 반환 합니다.  
uint dwInput, dwOutput;  
AxdGetContactNum (0, ref dwInput, ref dwOutput);
```

Visual Basic

```
'시스템에 연결된 0번 모듈의 MLII Simple IO 접점의 개수를 반환 합니다.  
Dim lInput As Long  
Dim lOutput As Long  
AxdGetContactNum 0, lInput, lOutput
```

Delphi

```
{시스템에 연결된 0번 모듈의 MLII Simple IO 접점의 개수를 반환 합니다.  
Var  
  dwInput : Dword;  
  dwOutPut : Dword;  
AxdGetContactNum (0, @dwInput, @dwOutPut)
```

See Also

[AxdiIsPulseOn](#), [AxdiIsPulseOff](#), [AxdiIsOn](#), [AxdiIsOff](#), [AxdoOutPulseOn](#), [AxdoOutPulseOff](#),
[AxdoToggleStart](#), [AxdSetContactNum](#)

에러코드 테이블 (**Error Code Table**) 확인

라이브러리 사용 중 오동작 또는 동작을 안 하는 경우가 발생 할 수 있다. 이러한 경우 함수의 리턴 값을 체크하므로 라이브러리 사용상에서의 문제점을 알 수 있다. 문제를 빨리 해결하고 손쉬운 디버깅을 위해 많은 에러 코드가 제공 되고 있으니 에러 코드를 잘 활용 하시요.

Error Code	Description
[0000] AXT_RT_SUCCESS	API 함수 수행 성공
[1001] AXT_RT_OPEN_ERROR	라이브러리 오픈 되지 않음
[1002] AXT_RT_OPEN_ALREADY	라이브러리 오픈 되어 있고 사용 중임
[1053] AXT_RT_NOT_OPEN	라이브러리 초기화 실패
[1054] AXT_RT_NOT_SUPPORT_VERSION	지원하지 않는 하드웨어
[1101] AXT_RT_INVALID_BOARD_NO	유효하지 않는 보드 번호
[1102] AXT_RT_INVALID_MODULE_POS	유효하지 않는 모듈 위치
[1103] AXT_RT_INVALID_LEVEL	유효하지 않는 레벨
[1151] AXT_RT_ERROR_VERSION_READ	라이브러리 버전을 읽을 수 없음
[1152] AXT_RT_NETWORK_ERROR	네트워크에 연결된 슬레이브 노드 연결 되지 않았거나 연결상태가 불량함.
[1153] AXT_RT_NETWORK_LOCK_MISMATCH	저장 된 연결정보와 실제 연결 정보가 맞지 않음. EzConfig에서 해당 보드의 Lock 정보와 실제 연결상태가 일치 하지 않음. 슬레이브 재구성 시 기존의 Lock 정보는 Ezconfig에서 Unlock으로 지워줘야 함.
[1160] AXT_RT_1ST_BELOW_MIN_VALUE	첫번째 인자값이 최소값보다 더 작음
[1161] AXT_RT_1ST_ABOVE_MAX_VALUE	첫번째 인자값이 최대값보다 더 큼
[1170] AXT_RT_2ND_BELOW_MIN_VALUE	두번째 인자값이 최소값보다 더 작음
[1171] AXT_RT_2ND_ABOVE_MAX_VALUE	두번째 인자값이 최대값보다 더 큼
[1180] AXT_RT_3RD_BELOW_MIN_VALUE	세번째 인자값이 최소값보다 더 작음
[1181] AXT_RT_3RD_ABOVE_MAX_VALUE	세번째 인자값이 최대값보다 더 큼
[1190] AXT_RT_4TH_BELOW_MIN_VALUE	네번째 인자값이 최소값보다 더 작음
[1191] AXT_RT_4TH_ABOVE_MAX_VALUE	네번째 인자값이 최대값보다 더 큼
[1200] AXT_RT_5TH_BELOW_MIN_VALUE	다섯번째 인자값이 최소값보다 더 작음
[1201] AXT_RT_5TH_ABOVE_MAX_VALUE	다섯번째 인자값이 최대값보다 더 큼
[1210] AXT_RT_6TH_BELOW_MIN_VALUE	여섯번째 인자값이 최소값보다 더 작음
[1211] AXT_RT_6TH_ABOVE_MAX_VALUE	여섯번째 인자값이 최대값보다 더 큼
[1220] AXT_RT_7TH_BELOW_MIN_VALUE	일곱번째 인자값이 최소값보다 더 작음
[1221] AXT_RT_7TH_ABOVE_MAX_VALUE	일곱번째 인자값이 최대값보다 더 큼
[1230] AXT_RT_8TH_BELOW_MIN_VALUE	여덟번째 인자값이 최소값보다 더 작음
[1231] AXT_RT_8TH_ABOVE_MAX_VALUE	여덟번째 인자값이 최대값보다 더 큼

[1240] AXT_RT_9TH_BELOW_MIN_VALUE	아홉번째 인자값이 최소값보다 더 작음
[1241] AXT_RT_9TH_ABOVE_MAX_VALUE	아홉번째 인자값이 최대값보다 더 큼
[1250] AXT_RT_10TH_BELOW_MIN_VALUE	열번째 인자값이 최소값보다 더 작음
[1251] AXT_RT_10TH_ABOVE_MAX_VALUE	열번째 인자값이 최대값보다 더 큼
[2001] AXT_RT_AIO_OPEN_ERROR	AIO 모듈 오픈실패
[2051] AXT_RT_AIO_NOT_MODULE	AIO 모듈 없음
[2052] AXT_RT_AIO_NOT_EVENT	AIO 이벤트 읽지 못함
[2101] AXT_RT_AIO_INVALID_MODULE_NO	유효하지않은 AIO 모듈
[2102] AXT_RT_AIO_INVALID_CHANNEL_NO	유효하지않은 AIO 채널번호
[2106] AXT_RT_AIO_INVALID_USE	AIO 함수 사용못함
[2107] AXT_RT_AIO_INVALID_TRIGGER_MODE	유효하지않는 트리거 모드
[3001] AXT_RT_DIO_OPEN_ERROR	DIO 모듈 오픈실패
[3051] AXT_RT_DIO_NOT_MODULE	DIO 모듈 없음
[3052] AXT_RT_DIO_NOT_INTERRUPT	DIO 인터럽트 설정안됨
[3101] AXT_RT_DIO_INVALID_MODULE_NO	유효하지않는 DIO 모듈 번호
[3102] AXT_RT_DIO_INVALID_OFFSET_NO	유효하지않는 DIO OFFSET 번호
[3103] AXT_RT_DIO_INVALID_LEVEL	유효하지않는 DIO 레벨
[3104] AXT_RT_DIO_INVALID_MODE	유효하지않는 DIO 모드
[3105] AXT_RT_DIO_INVALID_VALUE	유효하지않는 값 설정
[3106] AXT_RT_DIO_INVALID_USE	DIO 함수 사용못함
[4001] AXT_RT_MOTION_OPEN_ERROR	모션 라이브러리 Open 실패
[4051] AXT_RT_MOTION_NOT_MODULE	시스템에 장착된 모션 모듈이 없음
[4052] AXT_RT_MOTION_NOT_INTERRUPT	인터럽트 결과 읽기 실패
[4053] AXT_RT_MOTION_NOT_INITIAL_AXIS_NO	해당 축 모션 초기화 실패
[4054] AXT_RT_MOTION_NOT_IN_CONT_INTERPOL	연속 보간 구동 중이 아닌 상태에서 연속보간 중지 명령을 수행 하였음
[4055] AXT_RT_MOTION_NOT_PARA_READ	원점 구동 설정 파라미터 로드 실패
[4101] AXT_RT_MOTION_INVALID_AXIS_NO	해당 축이 존재하지 않음
[4102] AXT_RT_MOTION_INVALID_METHOD	해당 축 구동에 필요한 설정이 잘못됨
[4103] AXT_RT_MOTION_INVALID_USE	'uUse' 인자값이 잘못 설정됨
[4104] AXT_RT_MOTION_INVALID_LEVEL	'uLevel' 인자값이 잘못 설정됨
[4105] AXT_RT_MOTION_INVALID_BIT_NO	범용 입출력 해당 비트가 잘못 설정됨
[4106] AXT_RT_MOTION_INVALID_STOP_MODE	모션 정지 모드 설정값이 잘못됨
[4107] AXT_RT_MOTION_INVALID_TRIGGER_MODE	트리거 설정 모드가 잘못 설정됨
[4108] AXT_RT_MOTION_INVALID_TRIGGER_LEVEL	트리거 출력 레벨 설정이 잘못됨
[4109] AXT_RT_MOTION_INVALID_SELECTION	'uSelection' 인자가 COMMAND 또는 ACTUAL 이외의 값으로 설정되어 있음

[4110] AXT_RT_MOTION_INVALID_TIME	Trigger 출력 시간값이 잘못 설정되어 있음
[4111] AXT_RT_MOTION_INVALID_FILE_LOAD	모션 설정값이 저장된 파일이 로드가 안됨
[4112] AXT_RT_MOTION_INVALID_FILE_SAVE	모션 설정값을 저장하는 파일 저장에 실패함
[4113] AXT_RT_MOTION_INVALID_VELOCITY	모션 구동 속도값이 0 으로 설정되어 모션 에러 발생
[4114] AXT_RT_MOTION_INVALID_ACCELTIME	모션 구동 가속 시간값이 0 으로 설정되어 모션 에러 발생
[4115] AXT_RT_MOTION_INVALID_PULSE_VALUE	모션 단위 설정 시 입력 펄스값이 0 보다 작은값으로 설정됨
[4116] AXT_RT_MOTION_INVALID_NODE_NUMBER	위치나 속도 오버라이드 함수가 모션 정지 중에 실행됨
[4117] AXT_RT_MOTION_INVALID_TARGET	다축 모션 정지 원인에 관한 플래그를 반환한다.
[4151] AXT_RT_MOTION_ERROR_IN_NONMOTION	모션 구동중이어야 되는데 모션 구동중이 아닐 때
[4152] AXT_RT_MOTION_ERROR_IN_MOTION	모션 구동 중에 다른 모션 구동 함수를 실행함
[4153] AXT_RT_MOTION_ERROR	다축 구동 정지 함수 실행 중 에러 발생함
[4154] AXT_RT_MOTION_ERROR_GANTRY_ENABLE	겐트리 enable 이 되어있어 모션중일 때 또 겐트리 enable 을 눌렀을 때
[4155] AXT_RT_MOTION_ERROR_GANTRY_AXIS	겐트리 측이 마스터채널
[4156] AXT_RT_MOTION_ERROR_MASTER_SERVOON	마스터 측 서보온이 안되어있을 때
[4157] AXT_RT_MOTION_ERROR_SLAVE_SERVOON	슬레이브 측 서보온이 안되어있을 때
[4158] AXT_RT_MOTION_INVALID_POSITION	유효한 위치에 없을 때
[4159] AXT_RT_ERROR_NOT_SAME_MODULE	똑 같은 모듈내에 있지 않을경우
[4160] AXT_RT_ERROR_NOT_SAME_BOARD	똑 같은 보드내에 있지 아닐경우
[4161] AXT_RT_ERROR_NOT_SAME_PRODUCT	제품이 서로 다를경우
[4162] AXT_RT_NOT_CAPTURED	위치가 저장되지 않을 때
[4163] AXT_RT_ERROR_NOT_SAME_IC	같은 칩내에 존재하지않을 때
[4164] AXT_RT_ERROR_NOT_GEARMODE	기어모드로 변환이 안될 때
[4165] AXT_ERROR_CONTI_INVALID_AXIS_NO	연속보간 축맵핑 시 유효한 측이 아닐 때
[4166] AXT_ERROR_CONTI_INVALID_MAP_NO	연속보간 맵핑 시 유효한 맵핑 번호가 아닐 때
[4167] AXT_ERROR_CONTI_EMPTY_MAP_NO	연속보간 맵핑 번호가 비워 있을 때
[4168] AXT_RT_MOTION_ERROR_CACULATION	계산상의 오차가 발생했을 때
[4169] AXT_RT_ERROR_MOVE_SENSOR_CHECK	연속보간 구동전 에러센서가(Alarm, EMG, Limit 등) 감지된 경우
[4170] AXT_ERROR_HELICAL_INVALID_AXIS_NO	헬리컬 축 맵핑 시 유효한 측이 아닐 때
[4171] AXT_ERROR_HELICAL_INVALID_MAP_NO	헬리컬 맵핑 시 유효한 맵핑 번호가 아닐 때
[4172] AXT_ERROR_HELICAL_EMPTY_MAP_NO	헬리컬 맵핑 번호가 비워 있을 때
[4180] AXT_ERROR_SPLINE_INVALID_AXIS_NO	스플라인 축 맵핑 시 유효한 측이 아닐 때
[4181] AXT_ERROR_SPLINE_INVALID_MAP_NO	스플라인 맵핑 시 유효한 맵핑 번호가 아닐 때
[4182] AXT_ERROR_SPLINE_EMPTY_MAP_NO	스플라인 맵핑 번호가 비워있을 때
[4183] AXT_ERROR_SPLINE_NUM_ERROR	스플라인 점숫자가 부적당할 때
[4184] AXT_RT_MOTION_INTERPOL_VALUE	보간할 때 입력 값이 잘못넣어졌을 때

[4185] AXT_RT_ERROR_NOT_CONTIBEGIN	연속보간 할 때 CONTIBEGIN 함수를 호출하지 않을 때
[4186] AXT_RT_ERROR_NOT_CONTIEND	연속보간 할 때 CONTIEND 함수를 호출하지 않을 때
[4201] AXT_RT_MOTION_HOME_SEARCHING	홈을 찾고 있는 중일 때 다른 모션 함수들을 사용할 때
[4202] AXT_RT_MOTION_HOME_ERROR_SEARCHING	홈을 찾고 있는 중일 때 외부에서 사용자나 혹은 어떤것에 의한 강제로 정지당할 때
[4203] AXT_RT_MOTION_HOME_ERROR_START	초기화 문제로 홈시작 불가할 때
[4204] AXT_RT_MOTION_HOME_ERROR_GANTRY	홈을 찾고 있는 중일 때 겐트리 enable 불가할 때
[4251] AXT_RT_MOTION_POS_OUTOFCBOUND	설정한 위치값이 설정 최대값보다 크거나 최소값보다 작은값임
[4252] AXT_RT_MOTION_PROFILE_INVALID	구동 속도 프로파일 설정이 잘못됨
[4253] AXT_RT_MOTION_VELOCITY_OUTOFCBOUND	구동 속도값이 최대값보다 크게 설정됨
[4254] AXT_RT_MOTION_MOVE_UNIT_IS_ZERO	구동 단위값이 0 으로 설정됨
[4255] AXT_RT_MOTION_SETTING_ERROR	속도, 가속도, 저크, 프로파일 설정이 잘못됨
[4256] AXT_RT_MOTION_IN_CONT_INTERPOL	연속 보간 구동 중 구동 시작 또는 재시작 함수를 실행하였음
[4257] AXT_RT_MOTION_DISABLE_TRIGGER	트리거 출력이 Disable 상태임
[4258] AXT_RT_MOTION_INVALID_CONT_INDEX	연속 보간 Index 값 설정이 잘못됨
[4259] AXT_RT_MOTION_CONT_QUEUE_FULL	모션 칩의 연속 보간 큐가 Full 상태임
[4260] AXT_RT_PROTECTED_DURING_SERVOON	서보 온 상태에서 사용할 수 없는 함수 또는 입력값을 사용하였음.
[4261] AXT_RT_HW_ACCESS_ERROR	보드의 H/W 접근 오류가 발생함. 시스템내의 보드 장착 상태를 확인하거나 전원 입력 상태를 확인하여 조치함.
[4300] AXT_RT_COMPENSATION_SET_PARAM_FIRST	Compensation Set 을 하지 않고 Compensation Enable 을 했을 때
[4400] AXT_RT_SEQ_NOT_IN_SERVICE	순차구동함수실행중자원활당실패
[4401] AXT_ERROR_SEQ_INVALID_MAP_NO	순차구동함수실행중맵핑번호이상.
[4402] AXT_ERROR_INVALID_AXIS_NO	함수설정인지종축번호이상.
[4403] AXT_RT_ERROR_NOT_SEQ_NODE_BEGIN	순차구동노드입력시작함수를호출하지않음.
[4404] AXT_RT_ERROR_NOT_SEQ_NODE_END	순차구동노드입력종료함수를호출하지않음.
[4405] AXT_RT_ERROR_NO_NODE	순차구동노드입력이없음.
[4406] AXT_RT_ERROR_SEQ_STOP_TIMEOUT	순차구동함수 종료 시 TimeOut 발생

이 설명서의 내용은 예고 없이 변경될 수 있습니다. 용례에 사용된 회사, 기관, 제품, 인물 및 사건 등은 실제 데이터가 아닙니다. 어떠한 실제 회사, 기관, 제품, 인물 또는 사건과도 연관시킬 의도가 없으며 그렇게 유추해서도 안됩니다. 해당 저작권법을 준수하는 것은 사용자의 책임입니다. 저작권에서의 권리와는 별도로, 이 설명서의 어떠한 부분도 (주)아진엑스텍의 명시적인 서면 승인 없이는 어떠한 형식이나 수단(전기적, 기계적, 복사기에 의한 복사, 디스크 복사 또는 다른 방법) 또는 다른 목적으로도 복제되거나, 검색 시스템에 저장 또는 도입되거나, 전송될 수 없습니다.

(주)아진엑스텍은 이 설명서 본안에 관련된 특허권, 상표권, 저작권 또는 기타 지적 소유권 등을 보유할 수 있습니다. 서면 사용권 계약에 따라 (주)아진엑스텍으로부터 귀하에게 명시적으로 제공된 권리 이외에, 이 설명서의 제공은 귀하에게 이러한 특허권, 저작권 또는 기타 지적 소유권 등에 대한 어떠한 사용권도 허용하지 않습니다.