Library User Manual

AXL library family

AXA

AJINEXTEK AXT

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 theresponsibility 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

ⓒ Copyright 2010. AJINEXTEK co.ltd. All rights reserved.

Content

CONTENT **3**

AIO 라이브러리사용법 **8**

AIO개요 **8**

AIO 함수의기본규칙 9

AIO 함수그룹설명 10

초기화및종료 **11**

라이브러리의초기화 11

라이브러리의종료 13

아날로그입력 **16**

입력모듈의설정및확인 16

입력모듈의 A/D변환 22

이벤트 / 인터럽터 **32**

Event Mask의설정 32

이벤트사용여부의설정 33

이벤트처리방식의설정 33

인터럽터마스크의설정 41

아날로그출력 **43**

출력모듈설정 43

아날로그출력 44

AIO COMMAND 매뉴얼정보 **48**

헤더파일 **48**

함수용어 **49**

본매뉴얼의함수이름 49

본매뉴얼의인자이름 49

AIO COMMAND QUICK LIST 51

Trigger Mode에따른관련함수그룹 51

보드및모듈정보 51

입력모듈정보검색 52

입력모듈인터럽터/채널이벤트설정및확인 52

입력모듈파라미터설정및확인 52

입력모듈 A/D 변환 53

입력모듈버퍼상태체크 53

Extern Trigger 함수 54

출력모듈정보검색 55

출력모듈설정및확인 55

Define문 56

AIO COMMAND FUNCTION LIST 58

보드및모듈정보 58

AxaInfoIsAIOModule 59

AxaInfoGetModuleNo 61

AxaInfoGetInputCount 64

AxaInfoGetOutputCount 66

AxaInfoGetModuleCount 68

AxaInfoGetChannelNoOfModuleNo 70

AxaInfoGetChannelNoAdcOfModuleNo 72

AxaInfoGetChannelNoDacOfModuleNo 74

AxaInfoGetModuleStatus 76

AxaInfoGetModule 78

입력모듈정보검색 81

AxaiInfoGetModuleNoOfChannelNo 82

AxaiInfoGetChannelCount 84

입력모듈인터럽터/채널이벤트설정및확인 86

AxaiEventSetChannel 87
AxaiEventSetChannelEnable 91
AxaiEventGetChannelEnable 93
AxaiEventSetMultiChannelEnable 95
AxaiEventSetChannelMask 97
AxaiEventGetChannelMask 99
AxaiEventSetMultiChannelMask 101
AxaiEventRead 103
AxaiInterruptSetModuleMask 105
AxaiInterruptGetModuleMask 108

입력모듈파라미터설정및확인 110

AxaiSetRange 111
AxaiGetRange 113
AxaiSetRangeModule 115
AxaiGetRangeModule 117
AxaiSetMultiRange 119
AxaiHwSetBufferOverflowMode 121
AxaiHwGetBufferOverflowMode 123
AxaiHwSetMultiBufferOverflowMode 125
AxaiHwSetLimit 127
AxaiHwGetLimit 129
AxaiHwSetMultiLimit 131
AxaiSetTriggerMode 133
AxaiGetTriggerMode 135
AxaiHwSetSampleFreq 138
AxaiHwGetSampleFreq 140
AxaiHwSetSamplePeriod 142
AxaiHwGetSamplePeriod 144
AxaiSetModuleOffsetValue 146
AxaiGetModuleOffsetValue 148

입력모듈 A/D 변환 150

AxaiSwReadVoltage 151
AxaiSwReadDigit 153
AxaiSwReadMultiVoltage 155
AxaiSwReadMultiDigit 157
AxaiHwStartMultiChannel 160
AxaiHwStartMultiFilter 163
AxaiHwStopMultiChannel 166
AxaiHwSetMultiAccess 168
AxaiHwStartMultiAccess 171
AxaiHwReadSampleVoltage 174
AxaiHwReadSampleDigit 177
AxaiHwStartSingleChannelAdc 180
AxaiHwStopSingleChannelAdc 182

입력모듈버퍼상태체크 184

AxaiHwIsBufferEmpty 185
AxaiHwIsBufferUpper 187
AxaiHwIsBufferLower 189
AxaiHwReadDataLength 191

출력모듈정보검색 193

AxaoInfoGetModuleNoOfChannelNo 194
AxaoInfoGetChannelCount 196

출력모듈설정및확인 198

AxaoSetRange 199
AxaoGetRange 201
AxaoSetMultiRange 203
AxaoWriteVoltage 206
AxaoWriteMultiVoltage 208
AxaoReadVoltage 211
AxaoReadMultiVoltage 213

Extern Trigger 함수 215

AxaiExternalStartADC 216
AxaiExternalStopADC 218
AxaiExternalReadFifoStatus 219
AxaiExternalReadVoltage 221

에러코드테이블 (Error Code Table) 확인 223

AIO 라이브러리사용법

AIO개요

아날로그 신호를 입력 받거나, 출력하기 위한 제어모듈이 AIO(Analog Input Output) 모듈이다.

AIO 모듈은 PCI-AI16HR 일체형보드를 기준으로 볼 때 16개의 아날로그 입력 채널로 구성된 AI모듈(SIO-AI16HB)과 2개의 아날로그 출력 채널로 구성된 AO모듈(SIO-AO2HB)로 나눌 수 있는데, 더 많은 채널이 필요하다면, 여러 개의 일체형 보드 및 모듈을 사용해서 확장이 가능하다.

AXLLibrary는 아날로그 입출력 모듈에 대한 API함수들을 풍부하게 제공하고 있다. AIO 모듈에 대한 라이브러리 함수는 총 63개의 함수로 구성되며, 크게 다음과 같이 나누어진다.

- 초기화
- 모듈정보 확인
- 출력모듈의 설정 및 정보확인
- 입력모듈의 설정 및 정보 확인
- 입력모듈의 이벤트 설정
- 입력모듈의 인터럽터 마스크 설정
- 입력모듈 A/D변환
- 입력모듈 버퍼상태 확인

본User Guide에서는 여러 아날로그 입출력 모듈 중 16bit resolution 을 가지는일체형 보드 PCI-AI16HR(SIO-AI16HB/SIO-AO2HB)을기준으로 설명하되 입력 전압의 범위 설정은 SIO-AI4RB를 기준으로 한다.

AXA 라이브러리에서 지원하는 제품 내역은 다음과 같다.

이름	설명	인터럽트지원여부	비고
AXT_SIO_AI4RB	입력 4Ch, 12bit	지원	
AXT_SIO_AO4RB	출력 4Ch, 12bit	미지원	
AXT_SIO_AI16H	입력 4Ch, 16bit	지원	
AXT_SIO_AO8H	출력 4Ch, 16bit	미지원	
AXT_SIO_AI16HB	입력 16Ch, 16bit	지원	
AXT_SIO_AO2HB	출력 2Ch, 16bit	미지원	
AXT_SIO_RAI8RB	입력 8Ch, 16bit	미지원	RTEX
AXT_SIO_RAO4RB	출력 4Ch, 16bit	미지원	RTEX
AXT_SIO_RAI4MLII	입력 4Ch, 16bit	미지원	메카트로링크 II
AXT_SIO_RAO4MLII	출력 2Ch, 16bit	미지원	메카트로링크 II

AIO 함수의기본규칙

모든 AXL함수의 기본적인 규칙이기도 하지만, AIO함수군인 AXA에서도 각각의 함수들은 기본적으로 함수의 수행 결과를 리턴한다. 리턴 값은 아래의 표에서와 같이 해당 함수가 정상적으로 수행되었는지, 아니면 어떠한 이유로 실행되지 못했는지를 알려준다.

함수의 리턴 값을 확인하고자 한다면, 다음과 같이한다.

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

if(Code == AXT_RT_SUCCESS)
  printf ("정상적으로실행되었습니다.");
else
  printf ("AxaInfoIsAIOModule() : 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
[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

또한, 함수들은 매개변수를 통해 필요한 인자들을 입력 받고, 출력 또한 변수의 포인터를 넘겨받아 처리결과를 저장하여 사용자에게 전달한다. 아래에서와 같이 특정 함수 AxaExFunction를 실행시키는데, inPara1, inPara2,의 값이 필요하고, 결과값은 outPara1, outPara2에서 받아온다고 하면, 아래와 같이 한다.

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

AIO 함수그룹설명

AIO 함수는 다음과 같은 함수 접두어 그룹으로 나뉜다.

접두분류	설명
Axai	Analog 입력 관련 대분류
AxaInfo	보드및모듈정보확인함수
AxaiInfo	입력모듈정보검색함수
AxaiEvent	입력채널 이벤트설정및확인함수
AxaiInterrupt	입력모듈인터럽터설정및확인함수
AxaiSw	Software Timer (Normal Trigger Mode) 관련 함수
AxaiHw	Hardware Timer(Timer Trigger + External Trigger Mode) 관련 함수
Axao	Analog 출력 관련 대분류
AxaoInfo	출력모듈정보검색함수

초기화및종료

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

라이브러리의초기화

● 라이브러리의 초기화

AxlOpen: AxlOpen 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.

```
// 라이브러리를초기화한다.
// 7은 IRQ를뜻한다. PCI에서는자동으로 IRQ가설정된다.
if (AxlOpen(7) == AXT_RT_SUCCESS)
  printf("라이브러리가초기화되었습니다.");
```

● 라이브러리 초기화 여부의 확인

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

```
// 라이브러리가초기화되어있는지확인한다.
if (AxlIsOpened())
  printf("라이브러리가초기화되어있습니다.");
else
  printf("라이브러리가초기화되지않았습니다.");
```

● AIO모듈의 존재 여부 확인

AxaInfoIsAIOModule: AxaInfoIsAIOModule 함수는 실제 AIO 모듈이 존재하는가 여부를 확인하는 함수이다. AIO관련 함수를 사용하기 전에 AIO모듈이 있는지를 먼저 확인하기 위해 사용한다. 모듈이 존재하는지를 확인하려면 아래와 같이 한다.

```
// AIO 모듈이있는지확인한다.
DWORD dwStatus;
AxaInfoIsAIOModule(&dwStatus);
if(dwStatus == STATUS_EXIST)
  printf("AIO 모듈이존재합니다.");
else
  printf ("AIO 모듈이존재하지않습니다.");
```

● 입출력 모듈 및 채널의 개수 확인

AxaInfoGetModuleCount: AxaInfoGetModuleCount 함수는 전체 시스템에 장착되어 있는 AIO모듈의 개수를 확인하는 함수이다.

```
// AIO 모듈의개수를확인한다.
long    lCount;
AxaInfoGetModuleCount(&lCount);
printf("AIO 모듈의개수는 %d개입니다.", lCount);
```

AxaInfoGetInputCount: AxaInfoGetInputCount함수는 지정한 AIO모듈에서 입력채널의 개수를 확인하는 함수이다.

```
// 0번째모듈의입력채널개수를확인한다.
long    lCount;
AxaInfoGetInputCount(0, &lCount);
printf("0번째 AIO모듈의입력채널개수는 %d개입니다.", lCount);
```

AxaInfoGetOutputCount: AxaInfoGetOutputCount 함수는 지정한 AIO모듈에서 출력 채널의 개수를 확인하는 함수이다.

```
// 0번째모듈의출력채널개수를확인한다.
long    lCount;
AxaInfoGetOutputCount(0, &lCount);
printf("0번째 AIO모듈의출력채널개수는 %d개입니다.", lCount);
```

AxaiInfoGetChannelCount: AxaiInfoGetChannelCount함수는 전체 입력 채널의 개수를 확인하는 함수이다.

```
//등록되어있는전체아날로그입력채널의개수를반환
long lAdcChannelCounts;
Code = AxaiInfoGetChannelCount(&lAdcChannelCounts);
printf("등록되어있는전체입력채널개수는 %d개입니다.", lAdcChannelCounts);
```

AxaoInfoGetChannelCount: AxaoInfoGetChannelCount함수는 전체 출력채널의 개수를 확인하는 함수이다.

```
//등록되어있는전체아날로그출력채널의개수를반환
long lDacChannelCounts;
Code = AxaoInfoGetChannelCount(&lDacChannelCounts);
printf("등록되어있는전체출력채널개수는 %d개입니다.", lAdcChannelCounts);
```

AxaiInfoGetModuleNoOfChannelNo: AxaiInfoGetModuleNoOfChannelNo함수는 지정 입력채널의 모듈 번호를 확인하는 함수이다.

```
//지정입력채널의모듈번호를반환
long lModuleNo1, lModuleNo2;
Code = AxaiInfoGetModuleNoOfChannelNo(0, &lModuleNo1);
Code = AxaiInfoGetModuleNoOfChannelNo(20, &lModuleNo2);
printf("0번째입력채널모듈번호 : %d\n", lModuleNo1);
printf("20번째입력채널모듈번호 : %d\n", lModuleNo2);
```

AxaoInfoGetModuleNoOfChannelNo: AxaoInfoGetModuleNoOfChannelNo함수는 지정 입력채널의 모듈 번호를 확인하는 함수이다.

```
//지정출력채널의모듈번호를반환
long lModuleNo1, lModuleNo2;
```

```
Code = AxaoInfoGetModuleNoOfChannelNo(0, &lModuleNo1);
Code = AxaoInfoGetModuleNoOfChannelNo(2, &lModuleNo2);
printf("0번째출력채널모듈번호 : %d\n", lModuleNo1);
printf("2번째출력채널모듈번호 : %d\n", lModuleNo2);
```

●모듈 및 채널 번호의 의미

PCI-AI16HR을 기준으로 설명하면 PCI-AI16HR 보드에는 16채널 입력 모듈(SIO-AI16HB)과 2채널 출력 모듈(SIO-AO2HB)이 존재한다. 만약 시스템에 PCI-AI16HR 보드를 두 개 장착 할 경우 모듈과 각 채널의 인덱스의 할당 결과는 다음과 같다.

PCI-AI16HR 보드 두 개 장착 시 모듈 번호와 채널 번호 할당된 결과

모듈번호	입력/출력	채널인덱스
0	입력(SIO-AI16HB)	0~15
1	출력(SIO-AO2HB)	0~1
2	입력(SIO-AI16HB)	16~31
3	출력(SIO-AO2HB)	2~3

모듈이 다른 채널 0 ~ 15 과 16 ~ 31은 Sample rate, Trigger Mode, 인터럽터 마스크 등의 하드웨어적인 셋팅을 달리하여 모듈 별로 각기 다른 데이터 수집 방식을 취할 수 있다. 즉, 예를 들면 위와 같은 경우 모듈 '0'번인 채널 0 ~ 16은 Software Trigger Mode방식으로 데이터 수집을 하고 모듈 2번인 채널 16 ~ 31은 External Trigger Mode방식으로 데이터 수집이 가능하다.

라이브러리의종료

AxlClose: AxlClose 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.

```
// 라이브러리를종료한다.
if (AxlClose())
  printf("라이브러리가종료되었습니다.");
```

```
// Ex1_AXA_InitAndClose.cpp : Defines the entry point for the console application.
// 라이브러리를초기화하고, 입출력정보를확인한후종료한다.

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

void main(void)
{
  // 라이브러리를초기화한다.
```

```
// 7은 IRQ를뜻한다. PCI에서자동으로 IRQ가설정된다.
DWORD Code = AxlOpen(7);
if (Code == AXT_RT_SUCCESS)
{
   printf("라이브러리가초기화되었습니다.\n");

   //AIO 모듈이있는지검사
   DWORD dwStatus;
   Code = AxaInfoIsAIOModule(&dwStatus);
   if(Code == AXT_RT_SUCCESS)
   {
      if(dwStatus == STATUS_EXIST)
      {
         printf("AIO모듈이존재합니다.\n");
         //AIO모듈의개수를확인
         long lModuleCounts;
         Code = AxaInfoGetModuleCount(&lModuleCounts);
         if (Code == AXT_RT_SUCCESS)
            printf("AIO모듈의개수 : %d\n",lModuleCounts);
         else
            printf("AxaInfoGetModuleCounts() : ERRORcode 0x%x\n",Code);

         //입력및출력채널의개수를확인
         long lInputCounts;
         long lOutputCounts;
         for(int ModuleNo=0;ModuleNo<lModuleCounts;ModuleNo++)
         {
            Code = AxaInfoGetInputCount(ModuleNo,&lInputCounts);
            DWORD Code2 = AxaInfoGetOutputCount(ModuleNo,&lOutputCounts);

            if (Code == AXT_RT_SUCCESS || Code2 == AXT_RT_SUCCESS)
               printf ("%d번째모듈 :입력채널 %d개, 출력채널 %d개\n",
                     ModuleNo,lInputCounts,lOutputCounts);
            if (Code != AXT_RT_SUCCESS)
               printf ("AxaInfoGetInputCounts() : ERROR   code 0x%x\n",Code);
            if (Code2 != AXT_RT_SUCCESS)
               printf ("AxaInfoGetOutputCounts() : ERROR   code 0x%x\n",Code2);
         }

         //등록되어있는전체아날로그입력채널의개수를반환
         long lAdcChannelCounts;
         Code = AxaiInfoGetChannelCount(&lAdcChannelCounts);
         printf("등록되어있는전체입력채널개수는 %d개입니다.\n",lAdcChannelCounts);

         //등록되어있는전체아날로그출력채널의개수를반환
         long lDacChannelCounts;
         Code = AxaoInfoGetChannelCount(&lDacChannelCounts);
         printf("등록되어있는전체출력채널개수는 %d개입니다.\n",lAdcChannelCounts);
      }
```

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

   // 라이브러리를종료한다.
   if (AxlClose())
      printf("라이브러리가종료되었습니다.\n");
   else
      printf("라이브러리가정상적으로종료되지않았습니다.\n");
}
```

<PCI-AI16HR 보드 두 개 장착 시 출력 결과>

라이브러리가 초기화 되었습니다.

AIO모듈이 존재 합니다.

AIO모듈의 개수 : 4

0번째 모듈 : 입력채널 16개, 출력채널 0개

1번째 모듈 : 입력채널 0개, 출력채널 2개

2번째 모듈 : 입력채널 16개, 출력채널 0개

아날로그입력

아날로그 입력모듈은 전압 또는 전류 형태의 아날로그 신호를 입력 받아 설정된 샘플링 주파수 혹은 샘플링 간격에 따라 샘플링하여 디지털 데이터로 반환해 주는 모듈이다.

입력모듈의설정및확인

1) 입력 전압의 범위 설정

하기입력전압의범위설정에대한설명은 SIO-AI4RB를기준으로설명한다.

입력모듈을 사용하기 위해서 가장 먼저 해 주어야 할 설정은 입력전압의 범위를 설정하는 일이다. 사용자가 지정한 입력 채널에 허용 입력 전압 범위를 설정하기 위해서는 AxaiSetRange 함수를 이용한다. 적절한 기준 전압 설정은 성능 향상에 영향을 미치므로 신중히 선택 하여야 한다.

모듈 별로 허용 입력 전압이 정해져 있는데, 사용자가 이외의 전압을 입력하면 자동으로 그 값을 포함하는 근사 값으로 자동 설정된다. 최소 입력 전압은 아날로그 입력 모듈의 허용전압 중 최소 전압을 의미한다. 그리고 사용자가 설정한 값은 하드웨어가 지원하는 최소 허용전압을 포함하는 가장 근사치 값으로 설정된다. SIO-AI4R의 경우 최소 기준 전압으로 -10V, -5V, 0V를 제공하므로 사용자가 -8V를 입력하면 -10V로 자동 설정된다. 최대 입력 전압은 아날로그 입력의 허용 전압 중 최대 전압을 의미한다. 그리고 사용자가 설정한 값은 하드웨어가 지원하는 최대 허용전압을 포함하는 가장 근사치 값으로 설정된다. SIO-AI4R의 경우 최대 허용전압으로 10V, 5V를 제공하므로 사용자가 8V를 입력하면 10V로 자동 설정된다.

입력전압 범위는 0번 채널과 1번 채널, 2번 채널과 3번 채널이 서로 묶여있기 때문에 서로 같은 값으로 설정되며, 어느 하나만 해주어도 나머지 하나가 같은 입력전압 범위로 설정된다. 예를 들어 0번과 1번이 묶여 있으므로 0번 입력채널에 설정해준 전압범위 값이 1번 채널에도 적용된다.

Module	최소전압	최대전압	최소측정단위
SIO-AO4R	-10V	10V	4.88mV
	-5V	5V	2.44mV
	0V	10V	2.44mV
	0V	5V	1.22mV
SIO-AO2HB(PCI-AI16HR)	-10V	10V	0.305mV

AxaiSetRange: 입력전압의 범위를 설정하기 위해서는 AxaiSetRange함수를 사용하며, '0'번 채널의 입력 전압의 범위를 -8V에서 8V로 설정하고자 한다면 다음과 같이 한다.

```
//0번채널의입력전압의범위를 -8V에서 8V로설정
DWORD dwReturn = AxaiSetRange(0,-8,8);
//SIO_AI16HB(일체형보드 PCI-AI16HR의입력모듈을지칭) 모듈일경우전압범위고정되어있으므로
// AXT_RT_AIO_INVALID_USE 리턴됨.
```

위와 같이 설정하였을 경우 자동으로 -10V와 10V가 선택되어 설정된다. 만약, 이렇게 설정해 둔 상태에서 9V를 입력 받으면 정상적으로 9V가 입력된다. 왜냐하면, -8V ~ 8V를 설정하는 명령을 내렸지만, 실제 설정된 값은 -10V ~ 10V이기 때문이다.

AxaiSetMultiRange: 위에서와 같이 한번에 하나씩 설정해주는 함수 외에 한꺼번에 여러 개의 입력채널에 전압범위를 설정해 주는 AxaiSetMultiRange함수가 있다.

0~3번 입력채널의 전압범위를 -10V~10V로 설정하고자 한다면 다음과 같이 한다.

```
//여러채널의입력전압범위를설정
long AdcChannelNo[4] = {0,1,2,3};
DWORD dwReturn = AxaiSetMultiRange(4,AdcChannelNo,dMinVolt,dMaxVolt);
//SIO_AI16HB 모듈일경우전압범위고정되어있으므로 AXT_RT_AIO_INVALID_USE 리턴됨.
```

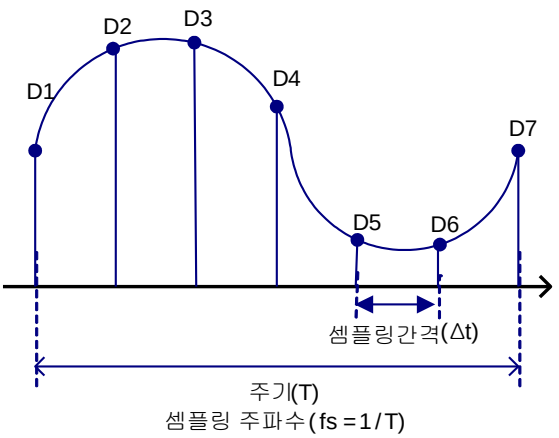
AxaiGetRange: 실제로 하드웨어에서 지원하지 않는 기준전압을 설정했을 때 설정한 값과 실제 설정되는 값이 다를 수 있다. 어떠한 값이 설정 되었는지를 알고자 한다면 AxaiGetRange함수를 사용하여 확인할 수 있다. 만약 '0'번 채널에 실제 설정된 값을 알고자 한다면 다음과 같이 한다.

```
// 0번입력채널의출력전압범위를확인한다.
double dVmin;
double dVmax;
AxaiGetRange(0, &dVmin, &dVmax);
```

2) 샘플링 모드의 설정

입력전압의 범위를 설정하였다면 다음으로 어떻게 샘플링을 할 것인지를 설정하여야 한다.

샘플링이란 입력되는 아날로그 전압을 일정간격(샘플링간격, Δt)으로 추출하여 해당되는 아날로그 값을 디지털 값으로 변환하는 과정으로 변환된 데이터는 모듈 내부의 FIFO메모리에 저장된다.



AxaiSetTriggerMode: 샘플링을 어떻게 할 것인지를 설정하는 과정이 트리거모드의 선택이다. 트리거 모드는 크게 Normal Trigger와 Timer Trigger 그리고 External Trigger가 있다. 트리거 모드의 구분은 샘플링이 이루어지는 시점이 어떻게 결정되는가에 따른 구분이다. 우리가 일반적으로 알고 있는 샘플링 즉, 특정 샘플링 주파수 또는 샘플링 간격을 기준으로 일정하게 샘플링을 하는 방법이 Timer Trigger 모드인 반면에, Normal Trigger 모드는 소프트웨어적으로 특정시간 또는 조건 등 사용자가 원하는 시점에 맞추어 샘플링을 하는 방법이다. External Trigger 모드는 외부의 특정 신호에 따라 샘플링을 하는 방식이다. SIO-AI16HB 모듈의 경우 내부 ADC가 1개이므로 하나의 모듈에 한가지 트리거 모드만을 적용시킬 수 있다. 각각의 모드를 설정하기 위한 설정값은 아래의 표와 같다.

Define	Value	Trigger Mode	Function
NORMAL_MODE	1	Normal Trigger	사용자가 원하는 시점에 A/D변환 하는 Trigger 방식
TIMER_MODE,	2	Timer Trigger	H/W의 내부 클럭을 이용해서 A/D변환 하는 Trigger 방식
EXTERNAL_MODE	3	External Trigger	외부 입력단자의 클럭을 이용해서 A/D변환 하는 Trigger 방식

예를 들어 인장응력 실험을 하고 있는 시편에 부착된 스트레인 게이지에서 나오는 전압값의 변화를 측정하고자 하면, 우리는 Timer Trigger모드를 이용하여 일정간격으로 지속적인 데이터를 획득 할 필요가 있다. 하지만, 특정 시점에 현재 상태만을 측정하여 작업을 수행하는 프로그램에서는 모든 데이터를 측정하고 있을 필요 없이 Normal Trigger 모드를 사용하여 사용자가 원하는 시점에서만 샘플링을 할 수도 있다.

이렇게 각각의 트리거 모드가 서로 다른 성격을 가지므로 각각의 트리거모드로 설정하였을 경우 사용하는 함수가 각각 정해져 있다.

트리거모드	사용하는함수
Normal Trigger Mode	AxaiSwReadVoltage, AxaiSwReadDigit AxaiSwReadMultiVoltage, AxaiSwReadMultiDigit
Timer Trigger Mode	AxaiHwStartMultiChannel, AxaiHwStartMultiFilter AxaiHwStopMultiChannel AxaiHwSetMultiAccess, AxaiHwStartMultiAccess
External Trigger Mode	AxaiHwStartMultiChannel, AxaiHwStartMultiFilter AxaiHwStopMultiChannel

만약 Normal Trigger모드를 사용하고자 한다면 다음과 같이 한다.

```
//0번입력모듈에Normal Trigger 모드를설정한다.
AxaiSetTriggerMode(0,NORMAL_MODE);
```

Normal Trigger Mode를 사용한다면, 바로 AxaiSwReadVoltage등 위의 표에 있는 전용함수를 이용하여 값을 읽어올 수 있지만, 내부 타이머를 이용하는 Timer Trigger모드는 추가적으로 샘플링 주파수를 반드시 셋팅해 주어야 하며 Immediate Mode가 아닌 연속적 신호 샘플링 시에는 Interrupt Mask의 셋팅, 데이터 유실을 방지하기 위한 더블 버퍼링에서의 버퍼 크기와 버퍼의 상한값과 하한값에 도달한 시점을 통지받기 위한 방법을 관련 함수를 통해 설정해주어야 한다.

샘플링 간격을 설정하는 방법에는 두 가지가 있다. 첫 번째는 샘플링 간격을 설정하는 방법이고, 두 번째는 샘플링 주파수를 설정하는 방법이다.

샘플링 주파수란 샘플링 주기의 역수인데, 초당 몇 번의 샘플링을 하는가를 의미한다. 당연히 주파수가 높을수록 샘플링 간격은 좁아져 더욱 정밀한 측정이 가능하고, 주파수가 낮을수록 샘플링 간격이 멀어지게 된다. 이론상으로 샘플링 주파수는 반드시 측정하고자 하는 아날로그 신호의 최고 주파수의 2배 이상이 되어야 하며, 그렇지 않을 경우 제대로 된 측정을 할 수 없다. 하지만, 실제로는 측정하고자 하는 신호의 최고주파수의 5 ~ 10배 정도의 주파수를 사용하는 것이 좋다. 샘플링 간격을 시간 단위로 설정해 주는 함수는 AxaiHwSetSamplePeriod함수이고, 샘플링을 주파수 단위로 설정해 주는 함수는 AxaiHwSetSampleFreq이다. 두 함수 모두 샘플링 주파수를 설정하는 함수이므로 둘 중의 하나만 사용하면 되다. SIO-AI16HB는 내부적으로 16개의 채널이 모두 공통의 ADC(Analog Digital Converter)를 사용하므로 하나의 샘플링 주파수 혹은 주기에 따라 16개의 채널 모두 샘플링 된다.

예로 SIO-AI16HB 의 경우 하나의 채널만 샘플링 할 경우 최대 100Khz의 주파수로 샘플링이 가능하며 16개 채널을 동시에 샘플링 할 경우 AxaiHwSetSampleFreq함수를 이용하여 설정하여야 할 주파수는 (100K/16 = 6250)Hz를 결코 넘어서서는 안된다. 만약 위 계산식에 어긋나게 하드웨어적인 최대 샘플링 한계치를 벗어난 값을 입력할 경우 주파수는 한계치로 자동 조정되어 동작이 수행된다. 반면 하드웨어 특성이 다른 SIO-AI4RB의 경우에는 4채널 ADC를 사용하므로 선택된 채널의 개수로 인한 주파수의 한계치를 정하지 않는다. 그러나, ADC 샘플링 한계치인 100Khz를 넘어설 수 없다.

AxaiHwSetSamplePeriod: 만약'0'번 모듈에 입력되는 아날로그 데이터를 100ms간격으로 샘플링 하고자 한다면 아래와 같이 설정한다. 단위는uSec이다.

```
//0번모듈의샘플링주기를 100us로설정한다.
//100 uSec
AxaiHwSetSamplePeriod(0,100);
```

AxaiHwGetSamplePeriod: 특정 모듈에 현재 설정되어 있는 샘플링 주기를 확인하고자 한다면 아래와 같이 AxaiHwGetSamplePeriod함수를 사용하여 확인하면 된다.단위는 uSec이다.

```
//0번모듈의샘플링주기를확인한다
double lSamplePeriod
AxaiHwGetSamplePeriod(AI_0,&lSamplePeriod);
```

AxaiHwSetSampleFreq: 만약'0'번 모듈에 입력되는 아날로그 데이터를 10Hz의 샘플링 주파수로 샘플링 하고자 한다면 다음과 같이 하면 된다.단위는 Hz이다.

```
//0번모듈의샘플링주파수를 10Hz로설정한다.
AxaiHwSetSampleFreq(0,10);
```

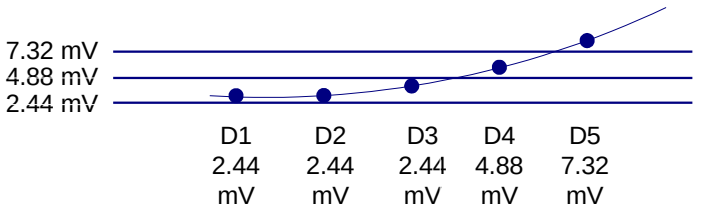
AxaiHwGetSampleFreq: 특정 모듈에 현재 설정되어 있는 샘플링 주파수를 확인하고자 한다면 아래와 같이 AxaiHwGetSampleFreq함수를 이용하면 된다. 단위는 Hz이다.

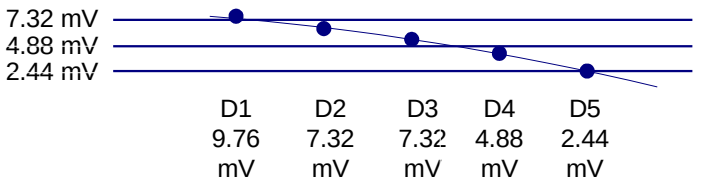
```
double dSampleFreq;
AxaiHwGetSampleFreq(0,&dSampleFreq);
```

3) 입력채널 버퍼의 설정

SIO-AI16HB모듈의 경우 16비트의 Resolution으로 샘플링을 하는데, 최대 전압과 최소 전압을 216 (65536개) 만큼으로 나누어 해당되는 값으로 저장하게 된다.

따라서 전압범위가 -10V ~ 10V로 고정인 SIO-AI16HB모듈의 경우 측정 가능한 최소 단위는 20V / 65536 즉, 약 0.3mV가 된다. 전압 입력 범위를 조절 할 수 있는 12 비트 Resolution인 SIO-AI4RB의 경우 전압범위를 0V ~ 10V로 설정하면 측정 가능한 최소 단위는 10V / 4096 즉, 2.44mV가 된다. 그리고 만약 측정 가능한 최소단위 사이의 입력이 들어온다면, 아래의 그림에서 보는 바와 같이 이전 값을 그대로 출력값으로 가지게 된다.





이렇게 샘플링 된 데이터는 모듈 내부의 FIFO 메모리에 저장된다. 모듈에 따라 FIFO의 크기가 다르나 AIO 라이브러리에서 제공하는 API를 이용하여 데이터의 유실없이 샘플링된 데이터를 획득할 수 있다.

AxaiHwSetBufferOverflowMode: 연속적인 신호 감시를 할 경우 모듈 내에 하드웨어적인 FIFO의 사이즈가 제한되어 있으므로 데이터의 유실을 방지하기 위해 더블 버퍼링을 하게 되는데 만약 더블 버퍼링으로 설정된 버퍼의 크기가 부족하여 버퍼에서 데이터를 적절한 시점에 가져오지 못 할 경우 OverFlow되게 된다. 이렇게 버퍼가 OverFlow되었을 때 어떻게 할 것인가를 설정해 주는 함수가AxaiHwSetBufferOverflowMode 함수이다. 할당되어 있는 메모리 버퍼에 데이터가 가득 찼을 때, 새로운 데이터가 들어오면 최초의 데이터를 버리고 새로운 데이터를 저장하던지 아니면 새로운 데이터를 버리던지 해야 한다.

AxaiHwSetBufferOverflowMode 함수는 NEW_DATA_KEEP을 입력하면 각 채널마다 할당되어있는 메모리 버퍼에 데이터가 가득 찼을 경우 이전 데이터를 버려 새로운 데이터를 유지하고 CURR_DATA_KEEP을 입력하면 최근의 데이터를 버려 최초 데이터를 유지하게 된다.

만약 0번 입력채널의 버퍼가 가득 찼을 때 오래된 데이터를 버리고 새로운 데이터가 유지되도록 설정하고자 한다면 다음과 같이 한다.

```
// 0번입력채널에버퍼가 Full로찼을때새로운데이터가유지되도록설정
AxaiHwSetBufferOverflowMode(0,NEW_DATA_KEEP);
```

AxaiHwSetMultiBufferOverflowMode: 위에서와 같이 한번에 하나씩 설정해주는 함수 외에 한꺼번에 여러 개의 입력채널을 설정해 주는 AxaiHwSetMultiBufferOverflowMode 함수가 있다.

0~3번 입력채널의 버퍼가 가득찼을 때 새로운 데이터를 버리고 기존의 데이터가 유지되도록 설정하고자 한다면 다음과 같이 하면 된다.

```
// 0~3번입력채널에버퍼가 Full로찼을때기존데이터가유지되도록설정
long AdcChannelNo[4] = {0,1,2,3};
AxaiHwSetMultiBufferOverflowMode(4, AdcChannelNo,CURR_DATA_KEEP);
```

AxaiHwSetLimit: 연속적 신호감시, 즉 데이터 변환을 할 경우, 하드웨어 모듈의 FIFO메모리로부터 사용자가 설정한 크기의 PC 메모리 버퍼로 내부 인터럽터를 통해 데이터 이동이 이루어 지는데 현재의 메모리 버퍼의 상태를 이벤트, 윈도우 메시지 또는 콜백함수를 통하여 얻어올 수 있다. 이 경우 이벤트를 얻고자 하는 메모리 버퍼의 상한값과 하한값을 AxaiHwSetLimit 함수로 설정하게 된다. 만약 하드웨어 FIFO로부터 메모리

버퍼로 옮겨진 데이터의 개수가 사용자가 지정한 상한을 넘어서게 되면 사용자는 이벤트를 통지 받게 되어 버퍼를 읽을 적절한 시점을 얻을 수 있다. 만약 '0'번 입력채널의 상한값을 1000, 하한값을 10으로 설정하고자 한다면 다음과 같이 한다.

그러나 이 상한값은 연속적인 데이터 획득을 위해 사용할 함수 AxaiHwStartMultiChannel 에서 지정할 실제 메모리 버퍼의 사이즈 보다 큰 값이어서는 안된다.

```
//0번입력채널의상한값을 1000, 하한값을 10으로설정
AxaiHwSetLimit(0,10,1000);
```

AxaiHwGetLimit: 특정 입력채널의 상한값과 하한값을 확인하고자 한다면 AxaiHwGetLimit 함수를 이용하여 다음과 같이한다.

```
//0번입력채널의상한값과하한값을확인
long lLowLimitGet, lUpLimitGet;
AxaiHwGetLimit(AdcChannelNo,&lLowLimitGet,&lUpLimitGet);
```

AxaiHwSetMultiLimit: 위에서와 같이 한번에 하나씩 설정해주는 함수외에 한꺼번에 여러 개의 입력채널을 설정해 주는 AxaiHwSetMultiLimit함수가 있다.

0 ~ 3번 입력 채널의 상한값과 하한값을 1000과 10으로 설정하고자 한다면 다음과 같이 하면 된다.

```
//여러채널의버퍼의상한값과하한값을설정
long AdcChannelNo[4] = {0,1,2,3};
Code = AxaiHwSetMultiLimit(4,AdcChannelNo,10,1000);
```

AxaiInterruptSetModuleMask: 연속적인 신호감시를 할 때 하드웨어 모듈의 FIFO에서 메모리 버퍼로 복사하는 시점을 AxaiInterruptSetModuleMask 함수를 사용하여 내부 인터럽터 발생 시점을 조정하여 지정 할 수 있다. SIO-AI4RB는 지원되지 않는다.

인터럽터 마스크의 종류는 다음과 같다.

(트리거 모드는 TIMER_MODE와 EXTERNAL_MODE에 해당된다.)

FIFO_HALF_FULL : 모듈내의 FIFO메모리가 HALF이상 찼을 경우 인터럽터가 발생하며 메모리 버퍼로 데이터 복사가 이루어 진다.(고속 샘플링시 적용)

SCAN_END : 셋팅된 채널 모두 ADC 변환이 한번 이루어 질 때 마다 인터럽터가 발생하며 메모리 버퍼로 데이터 복사가 이루어 진다. (저속 샘플링시 - 트리거 신호가 1 ~ 100 Hz 미만의 저속일 경우 사용)

기본 셋팅값은 FIFO_HALF_FULL 이다.

```
//모듈의하드웨어인터럽터 Mask를결정한다.

Code = AxaiInterruptSetModuleMask(0, SCAN_END);
// External 또는 Timer Trigger Mode 에서저속샘플링(1~100Hz)일경우 SCAN_END 사용
```

만약, 고속 샘플링인 상태인데 SCAN_END로 인터럽터 마스크를 지정하게 되면 한번의 ADC 변환이 생길 때마다 인터럽터가 발생하게 됨으로 인해 오버헤드가 심해진다. 반대로 저속 샘플링에서 FIFO_HALF_FULL로 마스크를 지정하게 되면 FIFO의 HALF 까지 데이터가 차야만 메모리 버퍼로 데이터 복사가 이루어지기 때문에 데이터의 시간 지연이 심해진다.

예를 들어 10Hz 의 주파수로 하나의 채널을 샘플링을 할 경우 FIFO_HALF_FULL로 셋팅되어 있다면, SIO_AI16HB의 FIFO 크기는 1024 개이므로 HALF인 512 개의 데이터가 수집되어야만 메모리 버퍼로 복사되므로 약 51초가량의 시간이 지체되어야 메모리 버퍼에서 수집된 데이터를 얻어올 수 있게 된다. 그러나

SCAN_END로 지정하게 되면, 한번의 샘플링이 끝나는 0.1초만에 메모리 버퍼에서 수집된 첫 번째 데이터를 얻어 올 수 있다. 또한, 만약 100Khz의 주파수로 하나의 채널을 샘플링 할 경우에 SCAN_END로 지정되어 있다면, 10 usec 마다 인터럽터가 발생하여 데이터 복사가 이루어 지는 엄청난 오버헤더가 발생하게 된다. 이 경우 FIFO_HALF_FULL로 셋팅하면 약 5msec 마다 한꺼번에 512개씩 데이터 복사가 이루어지므로 오버헤더로 인한 부하가 없어진다.

입력모듈의 A/D변환

1) Normal Trigger Mode에서의 A/D 변환

Normal Trigger 모드에서는 소프트웨어적으로 원하는 시점에 현재 상태를 샘플링하도록 되어 있다. 특정 시점 또는 특정 상황에서 AxaiSwReadVoltage함수나 AxaiSwReadDigit함수를 호출 함으로써 그 시점에 입력된 아날로그 신호를 샘플링하여 A/D변환 한다.

AxaiSwReadVoltage: AxaiSwReadVoltage함수는 지정한 입력채널의 현재 아날로그 입력값을 A/D변환한 후 전압값으로 반환한다.'0'번 채널의 아날로그 입력값을 전압값으로 확인하고자 한다면 다음과 같이 하면 된다.

```
//0번입력채널의아날로그입력값을전압값으로확인한다.
double dVolt;
AxaiSwReadVoltage(0,&dpVolt);
```

AxaiSwReadDigit: AxaiSwReadDigit함수는 지정한 입력채널의 아날로그 입력값을 A/D변환한 후 Digit값으로 반환한다. SIO-AI16HB의 경우 16비트 Resolution을 가지므로 0 ~ 65535의 Digit값을 얻게 된다. '0'번 채널의 아날로그 입력값을 Digit값으로 확인하고자 한다면 다음과 같이 하면 된다.

```
//0번입력채널의아날로그입력값을 Digit값으로확인한다.
DWORD dpDigit;
AxaiSwReadDigit(0,&dpDigit);
short retShort = (short) dpDigit;      // 부호있는 2Byte 로형변환
double dVolt = (((double)retShort )/ 32768.0) * 10;
//SIO-AI16HB Digit값이 Two's complement 이므로위와같이변환하면 Voltage 근사값을얻게된다.
```

```
// Ex2_AXA_NormalTrigger.cpp : Defines the entry point for the console application.
// 라이브러리를초기화하고, 입출력정보를확인한후종료한다.

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

void main(void)
{
   // 라이브러리를초기화한다.
   // 7은 IRQ를뜻한다. PCI는자동으로 IRQ가설정된다.
   DWORD Code = AxlOpen(7);
   if (Code == AXT_RT_SUCCESS)
   {
      printf("라이브러리가초기화되었습니다.\n");
```

```
//AIO 모듈이있는지검사
DWORD dwStatus;
Code = AxaInfoIsAIOModule(&dwStatus);
if(Code == AXT_RT_SUCCESS)
{
   if(dwStatus == STATUS_EXIST)
   {
      //여러채널의입력전압범위를 -10V ~ 10V로설정
      long AdcChannelNo[4] = {0,1,2,3};
      AxaiSetMultiRange(4,AdcChannelNo,-10,10);

      //0번모듈의트리거모드를 NOMAL로설정
      AxaiSetTriggerMode(0,NORMAL_MODE);

      //무한루프를돌면서아날로그입력을받는다.
      printf("[INFORMATION]********************************* \n");

      //설정된입력전압범위를확인한다.
      double dVmin, dVmax;
      for(int nChannalNo=0; nChannalNo<4; nChannalNo++)
      {
         AxaiGetRange(0, &dVmin, &dVmax);
         printf("입력모듈의전압범위 :채널 %d ( %.1f ~ %.1f) \n",nChannalNo,dVmin,dVmax);
      }

      printf("[ESC] : Exit \n");
      printf("[1] : Normal Trigger Mode 에서의셈플링(전압값)\n");
      printf("[2] : Normal Trigger Mode 에서의셈플링(Digit)\n");
      printf("********************************************* \n");

      BOOL fExit = FALSE;
      while (!fExit)          // 무한루프
      {
         if (kbhit())         // 아무키나누르면
         {
            int ch = getch();
            DWORD dpDigit;
            double dpVolt;
            int nChannalNo;

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

               case '1':
                  //현재입력값을전압값으로확인
```

```
                        for(nChannalNo=0; nChannalNo<4; nChannalNo++)
                        {
                           AxaiSwReadVoltage(nChannalNo,&dpVolt);
                           printf("\n채널 %d 의현재입력(전압) : %f V",nChannalNo,dpVolt);
                        }
                     break;

                     case '2':
                        //현재입력값을 DIgit값으로확인
                        for(nChannalNo=0; nChannalNo<4; nChannalNo++)
                        {
                           AxaiSwReadDigit(nChannalNo,&dpDigit);
                           printf("\n채널 %d 의현재입력(Digit) : %d ",nChannalNo,dpDigit);
                        }
                     break;
                  }
               }
            }
         }
         else
            printf ("AxaInfoIsAIOModule() : ERROR ( NOT STATUS_EXIST ) code 0x%x\n",Code);
      }
      else
         printf ("AxaInfoIsAIOModule() : ERROR ( Return FALSE )  code 0x%x\n",Code);
   }
   else
      printf ("AxlOpen() : ERROR  code 0x%x\n",Code);

   // 라이브러리를종료한다.
   if (AxlClose())
      printf("라이브러리가종료되었습니다.\n");
   else
      printf("라이브러리가정상적으로종료되지않았습니다.\n");
}
```

2) Timer Trigger Mode에서의 A/D변환

Timer Trigger Mode는 사용자가 설정해준 샘플링 주파수 또는 샘플링 간격으로 특정 채널에 입력되는 아날로그 신호를 연속적으로 샘플링하는 기능이다. 앞에서도 언급했듯이 아날로그 센서에서 발생하는 연속적인 신호를 측정하는 경우가 많이 있는데, 이러한 경우에 사용할 수 있다.

AxaiHwStartMultiChannel: Timer Trigger Mode/External Trigger Mode를 이용해서 여러 입력채널에 연속적인 샘플링을 시작하기 위해서는 AxaiHwStartMultiChannel함수를 이용한다.

AxaiHwStartMultiChannel함수는 사용할 채널의 개수, 채널번호의 배열, 할당할 버퍼의 크기라는 3가지 인수를 받아서 작동된다. 만약 '0'번과 '1'번 채널만 사용하고자 한다면 사용할 채널의 개수는 2가 되고 채널 번호의 배열의 값은 0, 1이 되는 것이다. 할당할 버퍼의 크기는 사용자가 임의로 정해줄 수 있으나, 셋팅된 Sampling Rate와 버퍼의 상한 이벤트를 받고 버퍼를 비우는 시점 사이의 지연을 고려하여 버퍼의 크기를 충분히 설정하는 것이 좋다. SIO-AI16HB를 예로 들어 만약 100Khz로 하나의 채널을 샘플링 할 경우 512 개의

데이터가 수집될 때 마다 모듈 내부 FIFO에서 PC메모리 버퍼로 데이터가 옮겨지게 되므로 약 5msec 마다 512개의 데이터가 수집되며, 상한값을 2048로 설정하게 되면 이벤트는 약 20msec 마다 통지 받게 된다. 따라서 버퍼의 크기에 대한 설정은 실제 Application에서 데이터를 처리하는 최대 부하를 고려하여 이 부하의 시간 동안 데이터 유실이 발생하는 것을 방지하기 위한 적절한 여유 공간의 버퍼의 크기를 설정하면 된다. 위의 예와 같은 고속 샘플링에 있어 데이터 수집외의 최대 부하의 처리 시간이 40msec 정도 예상된다면 최소 버퍼의 크기는 FIFO의 Half 사이즈의 8배 이상(4096 ~)이 되어야 한다. 만약 N개의 채널을 동시에 수집한다면 동일한 상한값에서 채널당 이벤트 발생 간격은 N배 만큼 시간 간격이 커지게 되므로 상한값을 적절히 조절해야 한다. 당연한 얘기지만 상한값은 결코 버퍼의 크기를 넘어설 수 없다. 상한값의 설정은 AxaiHwStartMultiChannel함수 호출 전에 AxaiHwSetMultiLimit 이나 AxaiHwSetLimit 함수로 지정할 수 있다.

만약'0'번과 '1'번 채널만 사용하고자 하고, 각 채널당 1024개의 버퍼를 할당하고자 한다면 아래와 같이 하면 된다.

```
// 0번과 1번입력채널에각각 1024개씩의버퍼를할당하여 A/D변환을시작한다.
long AdcChannelNo[2] = {0,1};
AxaiInterruptSetModuleMask(0, FIFO_HALF_FULL);
//FIFO가 Half를넘어설경우버퍼로데이터이동
AxaiHwSetMultiLimit(2, AdcChannelNo, 0, 256);
//버퍼의상태이벤트를받을하한/상한값지정
AxaiHwStartMultiChannel (2, AdcChannelNo, 1024);
// 채널당버퍼설정
//상기코드에서 SIO-AI16HB를예로든다면 FIFO 가 512개를넘어설경우버퍼로이동이되며
// 2개채널을감시하므로각버퍼의상한값을 256으로지정한다면매번의 FIFO_HALF_FULL
// 인터럽터마다버퍼의상한에도달하므로이벤트 AIO_EVENT_DATA_UPPER가발생된다.
// 위와같은코드는매번의내부인터럽터마다바로바로데이터를수집하게된다.
// 고속샘플링에서는시스템부하를유발하므로바람직하지않다.
// 고속샘플링에서는버퍼의상한값을 (512/채널) * N 배를한다. 여기서 N은샘플링주파수의크기에비례해서 2~
//수백배까지지정한다.
```

AxaiHwStopMultiChannel: AxaiHwStartMultiChannel를 통해 연속적 신호감시를 하던 채널들이 포함된 각 모듈의 A/D 변환이 종료된다. 파라메터인 IModuleNo가 -1인 경우 연속적 신호감시를 하던 시스템에 장착된 모든 모듈이 모두 A/D변환이 종료된다. 함수 사용은 다음과 같다.

```
//A/D변환을종료한다.
AxaiHwStopMultiChannel (-1);
//모든모듈의 A/D변환종료
```

AxaiHwReadSampleVoltage: A/D변환을 시작하면 설정된 샘플링 간격으로 연속적인 샘플링 결과를 모듈 FIFO에서 인터럽터를 통해 PC 메모리 버퍼에 저장하게 된다. 이렇게 저장된 샘플링 결과를 확인해야 하는데, AxaiHwReadSampleVoltage 함수는 그 값을 전압값으로 확인하고, digit값으로 확인하는 함수가 AxaiHwReadSampleDigit 함수이다. 이들은 AxaiHwReadDataLength 함수를 통해 현재 버퍼에 저장되어 있는 데이터의 개수를 읽어와서 그 범위안에서 원하는 개수만큼의 데이터를 한꺼번에 얻어올 수 있다. 실제 샘플링은 셋팅된 샘플링 Rate 또는 External Trigger의 신호 Rate에 따라 연속적으로 매우 빠르게 진행되므로 버퍼에 데이터가 있는지를 검사하여 데이터가 있다면 이 함수들을 이용하여 버퍼에 입력된 데이터들을 지속적으로 읽어주어야 한다. 데이터를 읽어가면 버퍼는 클리어 된다. 버퍼에 데이터가 있는지 검사하는 바람직한 프로그램밍 방법은 AxaiHwSetMultiLimit와 AxaiEventSetChannel 함수를 통하여 사용자가 버퍼의 상한을 지정하여 버퍼가 상한값을 초과 할 경우 이벤트를 통지 받아 적절한 시점에 해당 채널의 버퍼에 적재되어 있는 데이터의 개수만큼 데이터를 읽어 주게 되면 버퍼에 데이터가 있는지 Polling 하며 검사하지 않아도 된다.

'0'번 채널의 아날로그 입력값을 전압값으로 확인하고자 한다면 다음과 같이 하면 된다.

```
//0번입력채널에해당하는입력버퍼에저장되어있는데이터를전압값으로지정된블록단위로읽어온다.
double dpVolt[1024];
long lCount = 1024;
AxaiHwReadSampleVoltage(0, &lCount, dpVolt);
//lCount에는실제읽어들인데이터갯수가반환된다.
```

AxaiHwReadSampleDigit: AxaiHwReadSampleDigit함수는 지정한 입력채널의 아날로그 입력값을 A/D변환한 후 Digit값으로 반환한다. SIO-AI16HB모듈은 16bit Resolution의 해상도를 가지고 있으므로 0~65535사이의 값을 갖게 된다. 0번 채널의 아날로그 입력값을 Digit값으로 확인하고자 한다면 다음과 같이 하면 된다.

```
//0번입력채널에해당하는입력버퍼에저장되어있는데이터를 Digit값으로하나씩읽어온다.
DWORD lpDigit[1024];
Long lCount = 1024
AxaiHwReadSampleDigit(0,&lCount, lpDigit);  //lCount에는실제읽어들인데이터갯수가반환된다
short retShort = (short) lpDigit[0];        // 부호있는 2Byte로형변환
double dVoltFirst = (((double)retShort ) / 32768.0) * 10;
//Digit값이 Two's complement 이므로위와같이변환하면 Voltage 근사값을얻게된다.
```

AxaiHwReadDataLength: 버퍼에 몇 개의 데이터가 있는지를 검사하기 위해서는 AxaiHwReadDataLength 함수를 사용한다. 이 함수는 각채널에할당되어있는메모리버퍼에현재몇개의데이터가있는지그개수를반환한다. 메모리버퍼는 TimerTrigger/External Trigger를사용하는모드에서할당되므로Normal Trigger방식을사용하는모드에서는이함수는의미가없다. 현재 '0'번 채널의 버퍼에 몇 개의 데이터가 있는지를 확인하기 위해서는 다음과 같이 한다.

```
//0번입력채널의메모리버퍼에데이터가몇개인지검사한다.
long lpDataLength;
AxaiHwReadDataLength(AdcChannelNo,&lpDataLength);
```

```
// Ex3_AXA_TimerTrigger.cpp : Defines the entry point for the console application.
// 타이머트리거모드를이용하여 A/D변환한다.

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

void __stdcall CallBack_ReadChannelData(long lActiveNo, DWORD uFlag)
{
   //지정한버퍼상한값에도달시이벤트를받을콜백함수
   //윈도우메시지및이벤트처리방식도이샘플에서는콜백함수를예제로듬.
   if(uFlag == AIO_EVENT_DATA_UPPER)// 상한값도달이벤트인경우
   {
      GraphData graphData;
      double dVolt[2048];
      long lDataLength = 0;
      longlCount = 0;

      DWORD dwReturn = AxaiHwReadDataLength(lActiveNo, &lDataLength);
```

```
      if(dwReturn!=0)
      {
         return;
      }
      if(lDataLength > 2048)
         lCount = 2048;
      else
         lCount = lDataLength;
      dwReturn = AxaiHwReadSampleVoltage(lActiveNo, &lCount, dVolt);
      if(dwReturn!=0)
      {
         return;
      }
      /* 여기서 lCount 만큼수집된데이터를처리이부분의코드가데이터수집샘플링주기의 512배를넘어서는시간지연이걸
         리는작업을수행하는경우가발생할가능성이있다면데이터유실을방지하기위해콜백방식이아닌윈도우메시지방식이나
         이벤트처리방식을사용할것을권장함.   */
   }
}

void main(void)
{
   // 라이브러리를초기화한다.
   // 7은 IRQ를뜻한다. PCI는자동으로 IRQ가설정된다.
   DWORD Code = AxlOpen(7);
   if (Code == AXT_RT_SUCCESS)
   {
      printf("라이브러리가초기화되었습니다.\n");

      //AIO 모듈이있는지검사
      DWORD dwStatus;
      Code = AxaInfoIsAIOModule(&dwStatus);
      if(Code == AXT_RT_SUCCESS)
      {
         if(dwStatus == STATUS_EXIST)
         {
            long AdcChannelNo[4] = {0,1,2,3};

            //이미입력을받고있다면종료시킨다.
            AxaiHwStopMultiChannel(-1);
            AxaiInterruptSetModuleMask(0, FIFO_HALF_FULL);
            for(int I = 0; I < 4; i++)
            {
               AxaiEventSetChannel(i, NULL, NULL, (AXT_EVENT_PROC)CallBack_ReadChannelData,
                  NULL);                                  //이벤트를받을콜백함수지정
               AxaiEventSetChannelMask(i, DATA_MANY); //상한값에도달시콜백함수호출됨
               AxaiEventSetChannelEnable(i, ENABLE);     //이벤트호출 Enable
            }

            //여러채널의입력전압범위를 -10V ~ 10V로설정
```

```
        AxaiSetMultiRange(4,AdcChannelNo,-10,10);

        //여러채널의버퍼의상한값과하한값을설정
        AxaiHwSetMultiLimit(4,AdcChannelNo,0,1000);

        //0번모듈의트리거모드를 Timer Trigger로설정
        AxaiSetTriggerMode(0,TIMER_MODE);

        //특정모듈의셈플링주파수를 1000Hz로설정한다.
        AxaiHwSetSampleFreq(0,1000);

        //여러입력채널에버퍼가 Full로찼을때새로운데이터가유지되도록설정
        AxaiHwSetMultiBufferOverflowMode(4,AdcChannelNo,NEW_DATA_KEEP);

        // 모든입력채널에각각 2000개씩의버퍼를할당하여 A/D변환을시작한다.
        long lBuffSize = 2000;
        AxaiHwStartMultiChannel(4,AdcChannelNo,lBuffSize);
        printf("Timer Trigger Mode 의시작\n");
        //무한루프를돌면서아날로그입력을받는다.
        printf("[INFORMATION]************************************ \n");

        printf("[ESC] : Exit \n");
        printf("[1] : Timer Trigger Mode 에서의셈플링 (전압값)\n");
        printf("[2] : Timer Trigger Mode 에서의셈플링 (Digit)\n");
        printf("************************************************** \n");

        BOOL fExit = FALSE;
        int ReadMode = 0;
        while (!fExit)          // 무한루프
        {
           if (kbhit())         // 아무키나누르면
           {
              int ch = getch();
              switch (ch)
              {
                 case 27:       // Esc key
                    fExit = TRUE;
                    //Timer Trigger Mode 의정지
                    AxaiHwStopMultiChannel(-1);
                    printf("Timer Trigger Mode 의정지\n");
                    break;
                 default:
                    break;
              }
           }
           Sleep(100);
        } //END of while
     }
     else
```

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

   // 라이브러리를종료한다.
   if (AxlClose())
      printf("\n라이브러리가종료되었습니다.\n");
   else
      printf("\n라이브러리가정상적으로종료되지않았습니다.\n");
}
```

3) Immediate 모드에서의 A/D변환

Immediate모드는 A/D변환의 시작에서부터 특정 개수만큼만 데이터를 입력받는 경우에 사용하게 되는데, A/D변환한 데이터를 하드웨어에 있는 FIFO에 저장한 후 읽어오는 방식이다.

AxaiHwSetMultiAccess: 사용자가 지정한 여러 입력 채널에 Immediate모드를 사용하기 위해 설정 값을 설정한다. WordSize는 채널 당 데이터의 개수인데 SIO-AI16HB의 경우 내부적으로 1024Word의 하드웨어 FIFO를 가진다. 따라서 사용자가 1채널만 사용할 경우 1024개의 데이터를 지정할 수 있으며 여러 채널일 경우 1024 / 사용 채널 수 한 값이 최대 설정할 수 있는 데이터의 개수가 된다. 사용자가 이 값보다 큰 값을 지정하게 되면 자동으로 제한되게 된다. 함수 실행 후 *lpWordSize를 열어 보면 자동으로 제한된 값을 확인할 수 있다. 이 함수를 사용하기 전에 AxaiSetTriggerMode 함수를 사용하여 Timer Trigger Mode로 설정되어 있어야 한다.

```
//지정한입력채널에 Immediate모드를사용하기위한설정값을설정한다.
long AdcChannelNo[4] = {0,1,2,3};
long WordSize[4] = {100,100,100,100};
AxaiHwSetMultiAccess(4, AdcChannelNo, WordSize);
```

AxaiHwStartMultiAccess: AxaiHwStartMultiAccess함수는 내부적으로 해당 채널에 설정한 개수만큼의 데이터가 변환되어야 빠져 나오게 된다. 이 함수는 수행이 끝나게 되면 자동으로 A/D변환 값이 배열에 저장된다. *dpBuffer[]는 A/D변환 데이터 값 이 저장되는데 2차원 배열의 포인터를 저장할 배열이다. 우선 사용할 채널과 그 채널당 A/D변환할 데이터 수만큼 배열을 잡는다. 예를 들면 4개의 채널을 사용하고 채널당 A/D변환할 데이터의 개수가 10개라면 double dBuffer[4][10]라고 선언하고 double *pdBuffer[4]로 선언하면 된다. 이렇게 선언된 dBuffer의 포인터를 pdBuffer에 할당하면 된다. 0~3번 채널에 100개씩의 데이터를 입력받기 위해서는 다음과 같이 한다.

```
//지정한개수(100개)만큼 A/D변환후값을반환한다.
double pdCBuffer[4][100];
double *pdBuffer[4];
for(int i=0; i<4; i++)
  pdBuffer[i] = pdCBuffer[i];
AxaiHwStartMultiAccess(pdBuffer);
```

```
// Ex4_AXA_ImmediateMode.cpp : Defines the entry point for the console application.
// 라이브러리를초기화하고, 입출력정보를확인한후종료한다.

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

void main(void)
{
   // 라이브러리를초기화한다. 7은 IRQ를뜻한다. PCI는자동으로 IRQ가설정된다.
   DWORD Code = AxlOpen(7);
   if (Code == AXT_RT_SUCCESS)
   {
      printf("라이브러리가초기화되었습니다.\n");

      //AIO 모듈이있는지검사.
      DWORD dwStatus;
      Code = AxaInfoIsAIOModule(&dwStatus);
      if(Code == AXT_RT_SUCCESS)
      {
         if(dwStatus == STATUS_EXIST)
         {
            //여러채널의입력전압범위를 -10V ~ 10V로설정.
            long AdcChannelNo[4] = {0,1,2,3};
            AxaiSetMultiRange(4,AdcChannelNo,-10,10);

            //0번모듈의트리거모드를 Timer Trigger로설정.
            AxaiSetTriggerMode(0,TIMER_MODE);

            //특정모듈의샘플링주파수를 10Hz로설정한다.
            AxaiHwSetSampleFreq(0,100);

            //지정한입력채널에 Immediate모드를사용하기위한설정값을설정한다.
            long lWordSize[4] = {100,100,100,100};
            AxaiHwSetMultiAccess(4, AdcChannelNo, lWordSize);

            //지정한개수(100개)만큼 A/D변환후값이저장될배열.
            double pdCBuffer[4][100];
            double *pdBuffer[4];
            for(int i=0; i<4; i++)
               pdBuffer[i] = pdCBuffer[i];

            printf("\n 데이터를입력받고있습니다.\n");
            AxaiHwStartMultiAccess(pdBuffer);

            //데이터출력.
            for (i = 0; i < 100; i++)
               printf("T=%d : Ch0 [%3f], Ch1 [%3f], Ch2 [%3f], Ch3
```

```
                  [%3f]\n",i,pdBuffer[0][i], pdBuffer[1][i],pdBuffer[2][i],pdBuffer[3][i]);
         }
         else
            printf ("AxaInfoIsAIOModule() : ERROR ( NOT STATUS_EXIST ) code 0x%x\n",Code);
      }
      else
         printf ("AxaInfoIsAIOModule() : ERROR ( Return FALSE )  code 0x%x\n",Code);
   }
   else
      printf ("AxlOpen() : ERROR  code 0x%x\n",Code);

   // 라이브러리를종료한다.
   if (AxlClose())
      printf("라이브러리가종료되었습니다.\n");
   else
      printf("라이브러리가정상적으로종료되지않았습니다.\n");
}
```

이벤트 / 인터럽터

AXA에서는 모든 입력채널에 대해서 연속적 신호감시(AxaiHw 함수군중 Immediate 수집 함수 제외)를 할 때 할당해준 입력채널 버퍼가 가득찼는지, 비어있는지, 상한선을 넘었는지 또는 하한선보다 적은 데이터가 저장되어 있는지에 대해서 각각 이벤트를 설정하여 활용할 수 있도록 함수들을 제공한다. 이벤트를 사용하기 위해서는 먼저 해당 채널에 이벤트 종류(Event Mask)를 설정하고, 이벤트를 어떤 방식으로 처리할 것인가를 설정한다. AXA에서는 이벤트를 처리하는 방법으로 콜백함수 방식, 윈도우 메시지 방식, 이벤트 방식의 3가지 방식을 지원하므로, 사용자의 개발환경에 맞추어 선택하면 된다. AxaiEventSetChannelEnable 함수로 이벤트의 사용여부를 결정한다. 각 입력 채널 별로 이벤트를 각기 통지 받을 수 있으나 권장하는 방식은 여러 입력채널을 동시에 데이터 수집을 하더라도 모듈 내에 대표 되는 채널 하나에만 이벤트를 걸어서 그 채널에 대한 이벤트를 통지 받았을 때 나머지 채널도 처리 하는 방식이 효율적이다.

또한 하드웨어 모듈의 FIFO에서 메모리 버퍼로 복사하는 시점을 AxaiInterruptSetModuleMask 함수를 사용하여 내부 인터럽터 발생 시점을 FIFO_HALF_FULL(고속 샘플링시 사용) 이나 SCAN_END(수백 Hz 이하 저속 데이터 수집 샘플링시 사용) 중 선택하여 지정 할 수 있다.

SIO-AI4RB 모듈은AxaiInterruptSetModuleMask 함수를 지원하지 않는다.

Event Mask의 설정

이벤트 마스크는 어떠한 상황에서 이벤트를 발생시킬 것인지를 설정하는 값이다. 이 값을 아래의 표에서 나타낸 값으로 설정하면 해당 이벤트가 걸리게 되는것이다. 아래의 표에서와 같이 4가지 종류의 이벤트 마스크가 있다.

Define	Value	이벤트마스크종류
DATA_EMPTY	0x01	버퍼에 데이터가 없을 때 이벤트 발생
DATA_MANY	0x02	버퍼에 데이터가 상한 설정 값보다 많아질 때 이벤트 발생
DATA_SMALL	0x04	버퍼에 데이터가 하한 설정 값보다 적어질 때 이벤트 발생
DATA_FULL	0x08	버퍼에 데이터가 꽉 찼을 때 이벤트 발생

AxaiEventSetChannelMask: 특정 입력채널에 이벤트 마스크를 설정하기 위해서는 AxaiEventSetChannelMask 함수를 사용한다.

만약'0'번 입력채널에 할당된 버퍼가 설정된 상한값보다 많아지거나 하한값보다 적어질때에 이벤트가 발생하도록 설정하고자 한다면.

```
// 0번입력채널에이벤트종류를상한설정값보다많아질때와하한설정값보다적어질때발생하도록
// 설정한다.
AxaiEventSetChannelMask (0, DATA_MANY | DATA_SMALL);
```

AxaiEventGetChannelMask: 특정 입력채널에 현재 설정되어 있는 이벤트 마스크를 확인하기 위해서는 AxaiEventGetChannelMask 함수를 사용한다.

만약 0번 입력채널에 설정되어 있는 이벤트의 종류를 알고자 한다면 다음과 같이 하면 된다.

```
// 0번입력채널에이벤트종류를확인한다.
DWORD dwMask;
AxaiEventGetChannelMask (0, &dwMask);
```

AxaiEventSetMultiChannelMask: 한꺼번에 여러 개의 채널에 특정 이벤트 마스크를 설정하기 위해서는 AxaiEventSetMultiChannelMask 함수를 사용한다. lSize는 사용할 채널의 개수를 넣어주고, lAdcChannelNo는 채널의 정보를 담을 배열로서 사용할 채널의 번호를 배열의 값으로 넣어주면 된다. 예를 들어 long lAdcChannelNo[4] = {0, 2, 3, 4}; 처럼 하면 0, 2, 3, 4번 채널을 설정한다는 것이다.

만약 0 ~ 3번 입력채널에 할당된 버퍼가 설정된 상한값보다 많아지거나 하한값보다 적어질때에 이벤트가 발생하도록 설정하고 한다면 다음과 같이 한다.

```
// 0~3번입력채널에버퍼의데이터가상한설정값보다많아질때와하한설정값보다적어질때이벤트가
// 발생하도록설정한다.
long lSize = 4;
long lAdcChannelNo[4] = {0, 1, 2, 3};
AxaiEventSetMultiChannelMask(4, lAdcChannelNo, DATA_MANY | DATA_SMALL);
```

이벤트사용여부의설정

이벤트의 사용여부는 Default로 사용함(Enable)으로 설정되어 있다. 그러므로 이벤트 마스크가 설정되어 있고, 마스크의 설정대로 필요한 버퍼의 상한/하한값등이 셋팅되어 있다면 연속적 신호감시가 시작되는 AxaiHwStartMultiChannel/AxaiHwStartMultiFilter 함수가 구동되는 시점부터 이벤트 마스크에서 설정한 대로 이벤트가 발생한다. 사용하지 않으려면 Disable로 설정하면 된다. 다시 사용함으로 설정할 시 버퍼는 그 시점에서 리셋되고, 데이터 수집은 새롭게 이루어진다.

AxaiEventSetChannelEnable: 이벤트의 사용 여부를 설정하는 함수는 AxaiEventSetChannelEnable 함수이다. 만약 '0'번 채널에서 이벤트를 사용함으로 설정하고자 한다면 다음과 같이 한다.

```
//0번입력채널에서이벤트를사용함으로설정.
AxaiEventSetChannelEnable (0,ENABLE);
```

이벤트처리방식의설정

이벤트가 발생했을 때 이를 처리하는 방법으로 AXA에서는 콜백함수 방식, 윈도우 메시지 방식, 이벤트 방식 등 3가지를 제공한다. 콜백 함수 방식은 이벤트 발생 시점에 즉시 콜백 함수가 호출 됨으로 가장 빠르게 이벤트를 통지받을 수 있는 장점이 있으나 콜백 함수 내에 데이터를 처리하는 루틴등의 시스템 부하가 걸리는 작업이 있을 경우에는 데이터 유실의 위험이 있다. 즉, SIO-AI16HB의 경우 상기의 시스템 부하로 인한 시간 지연이 샘플링 주기의 512배를 넘어설 경우에 모듈내 FIFO에서 버퍼로 데이터 이동이 정체되어 데이터 유실이 발생할 수 있다. 수치적인 시간계산을 한다면 샘플링 주파수 100Khz일 경우 약 5 msec 이상의 시스템 부하시에 데이터유실이 발생할 수 있다. 따라서 데이터 처리에 있어서 부하가 걸리는 작업이 있을 경우 이 부분을 따로 동작 할 수 있게 하는 윈도우 메시지 방식이나 이벤트 방식을 사용할 것을 권장한다. 고속의 샘플링에서는 윈도우 메시지 방식도 윈도우의 메시지 큐의 Dispatch루틴 또한 단일 쓰레드이므로 메시지를 적절히 빠른 시간에 처리 하지 않을 경우 메시지가 적체되어 Application 동작에 영향을 미치게 된다. 그러나, 이벤트 방식은 이벤트의 발생 여부를 감시하는 특정 쓰레드를 사용하며 그 쓰레드 내에서 수집된 데이터를 처리 하게 되므로 MultiProcessor 시스템등에서 자원을 가장 효율적으로 사용할 수 있게 되어 특히 권장하는 방식이다.

이벤트 처리 방식을 설정하기 위해서는 AxaiEventSetChannel함수를 사용하여 다음과 같이 설정해주면 된다.

● 콜백함수 방식

콜백함수 방식을 사용하기 위해서는 아래와 같이 설정한다.

```
//0번채널의이벤트처리방식을콜백함수방식으로설정한다.
AxaiEventSetChannel(0,NULL,NULL,OnAIOEventCallback,NULL);
```

OnAIOEventCallback은 이벤트를 처리할 콜백함수의 포인터이다. 콜백함수는 전역함수로 다음과 같이 설정하면 된다. 이벤트가 발생하면 자동으로 이벤트를 처리할 콜백 함수를 호출하게 된다.

```
//이벤트를처리할콜백함수.
void OnAIOEventCallback(long lChannelNo, DWORD dwStatus)
{
  //lChannelNo :이벤트발생채널번호 ( 0, 1, 2, 3).
  //dwStatus :이벤트발생 Flag.
  long lpDataLength;
  DWORD dwDigit;

  switch(dwStatus)
  {
     case AIO_EVENT_DATA_UPPER:
        //  상한값을넘었을경우.
        printf("\nEVENT : %d번입력채널의데이터가상한값을넘었습니다.\n",lChannelNo);
     break;

     case AIO_EVENT_DATA_LOWER:
        //하한값보다적을경우.
        printf("\nEVENT : %d번입력채널의데이터가하한값보다적습니다.\n",lChannelNo);
     break;

     case AIO_EVENT_DATA_FULL:
        //데이터가가득찼을경우.
        printf("\nEVENT : %d번입력채널의데이터가가득찼습니다.\n",lChannelNo);
     break;

     case AIO_EVENT_DATA_EMPTY:
        //데이터가텅비었을경우.
        printf("\nEVENT : %d번입력채널의데이터가없습니다.\n",lChannelNo);
     break;
  }
}
```

● 윈도우 메시지 방식

윈도우 메시지 방식을 사용하기 위해서는 아래와 같이 설정한다. C에서는 윈도우 메시지방식을 사용하지 못하므로 이 부분은 VC++을 사용하여 설명한다. 윈도우 메시지 방식을 사용하기 위해서는 메시지를 받을 윈도우 헨들과 윈도우 메시지를 입력한다. 윈도우 메시지를 사용하지 않거나 디폴트 값을 사용하고자 한다면 '0'

을 입력한다.

```
//0번채널의이벤트처리방식을윈도우메시지방식으로설정한다.
AxaiEventSetChannel(CHANNEL_NO,this->m_hWnd,WM_AIO_EVENT,NULL,NULL);
```

사용방법을 전체적으로 살펴보면 아래와 같으며, OnAIOEventMessage함수에서 이벤트를 처리한다.

```
#define WM_AIO_EVENT 1011     //이벤트메시지를받아오기위한메세지정의.
#define CHANNEL_NO 0          //입력채널번호.

…(중략)…

//메시지와함수를연결
BEGIN_MESSAGE_MAP(CExDIOEventDlg, CDialog)
   //{{AFX_MSG_MAP(CExDIOEventDlg)

ON_MESSAGE(WM_AIO_EVENT,OnAIOEventMessage)

   //}}AFX_MSG_MAP
END_MESSAGE_MAP()

…(중략)…

void CExAIOEventDlg::OnBtnStartEventMessage()
{
   CString Message;

   //이벤트처리방식설정 ,윈도우메시지방식의예제.
   DWORD Code = AxaiEventSetChannel(CHANNEL_NO,this->m_hWnd,WM_AIO_EVENT,NULL,NULL);
   if (Code == AXT_RT_SUCCESS)
   Message.Format("%d번모듈의이벤트처리방식을콜백함수방식으로설정 ... OK",CHANNEL_NO);
   else
      Message.Format("AxdSetlModuleEvent() : ERROR   code 0x%d",Code);

   m_LIST_Message.InsertItem(CHANNEL_NO,Message);

   //특정채널의사용할이벤트종류를설정.
   AxaiEventSetChannelMask(CHANNEL_NO,DATA_EMPTY|DATA_MANY|DATA_SMALL|DATA_FULL);

   //특정채널의이벤트허용여부설정.
   AxaiEventSetChannelEnable(CHANNEL_NO,ENABLE);

   //TimerTriggerMode에서 A/D 변환을시작한다.
   StartAnalogInput();
}

void CExAIOEventDlg::OnAIOEventMessage(long lChannelNo, DWORD dwStatus)
{
   CString IntMessage;
   long lpDataLength;
```

```
    DWORD dwDigit;

    switch(dwStatus)
    {
        case AIO_EVENT_DATA_UPPER:
            //  상한값이넘었을경우
            IntMessage.Format("EVENT : %d번입력채널의데이터가상한값을넘었습니다.",lChannelNo);
            m_LIST_Message.InsertItem(0,IntMessage);
        break;
        case AIO_EVENT _DATA_LOWER:
            //하한값보다적을경우
            IntMessage.Format("EVENT : %d번입력채널의데이터가하한값보다적습니다.",lChannelNo);
            m_LIST_Message.InsertItem(0,IntMessage);
        break;
        case AIO_EVENT_DATA_FULL:
            //데이터가가득찼을경우
            g_bSaveBuffer = FALSE;
            IntMessage.Format("EVENT : %d번입력채널의데이터가가득찼습니다.",lChannelNo);
            m_LIST_Message.InsertItem(0,IntMessage);

            //타이머트리거의 A/D변환을종료시킨다.
            AxaiHwStopMultiChannel();
            IntMessage.Format("EVENT : %d번입력채널의 A/D 변환을종료합니다.",lChannelNo);
            m_LIST_Message.InsertItem(0,IntMessage);

            //버퍼에저장된데이터를읽는다.
            g_bReadBuffer = TRUE;
        break;
        case AIO_EVENT_DATA_EMPTY:
            //데이터가비었을경우
            g_bReadBuffer = FALSE;
            IntMessage.Format("EVENT : %d번입력채널의데이터가없습니다.",lChannelNo);
            m_LIST_Message.InsertItem(0,IntMessage);
        break;
    }
}
```

● 이벤트 방식

이벤트 방식을 사용하기 위해서는 아래와 같이 설정한다. 이벤트 방식은 이벤트의 발생 여부를 감시하는 특정 쓰레드를 사용하며 그 쓰레드 내에서 수집된 데이터를 처리 하게 되므로 MultiProcessor 시스템등에서 자원을 가장 효율적으로 사용할 수 있으므로 특히 권장하는 방식이다. 이 부분 또한 VC++을 사용하여 설명한다. 이벤트 방식을 사용하기 위해서는 이벤트 ID를 받을 HANDLE형 멤버변수 m_hEvent를 미리 선언해두고 사용하여야 한다.

```
//0번채널의이벤트처리방식을윈도우이벤트방식으로설정한다.
AxaiEventSetChannel (0,NULL,NULL,NULL,&m_hEvent);
```

사용방법을 전체적으로 살펴보면 아래와 같으며, ThreadProc함수에서 이벤트를 처리한다. 여기서 주의할 점

은 반드시 AxaiEventSetChannel함수를 쓰레드가 실행되기 전에 호출하여야 한다는 것이다.

설정된 이벤트가 발생하면 쓰레드는 이 이벤트를 감지하여 이벤트가 발생했다는 것을 알게 된다. 그리고 쓰레드 내부에서 AxaiReadEvent 함수를 호출하여 현재 이벤트가 발생한 채널 번호와 이벤트 플래그 정보를 확인하여 어떤 채널에 무슨 이벤트가 발생하였는지를 확인한다.

```
//이벤트방식이벤트사용에필요한변수.
HANDLE m_hThreadHandle;
BOOL m_bThread;
HANDLE m_hEventEvent;
```

```
void CExAIOEventDlg::OnBtnStartEventEvent()
{
   CString Message;

   //이벤트처리방식설정.
   DWORD Code =   AxaiEventSetChannel(0,NULL,NULL,NULL,&m_hEvent);  //이벤트방식.
   if (Code == AXT_RT_SUCCESS)
      Message.Format("%d번모듈의이벤트처리방식을이벤트방식으로설정 ... OK",CHANNEL_NO);
   else
      Message.Format("AxdSetlModuleEvent() : ERROR   code 0x%d",Code);

   m_LIST_Message.InsertItem(CHANNEL_NO,Message);

   //이벤트방식을사용하기위해서는 HANDLE pEvent 멤버변수로선언후초기화하고
   //쓰레드를만든다.
   m_bThread = TRUE;
   CreateThread(NULL, 0, (LPTHREAD_START_ROUTINE)ThreadProc, this, NULL, NULL);

   //특정채널의사용할이벤트종류를설정.
   AxaiEventSetChannelMask(CHANNEL_NO,DATA_EMPTY | DATA_MANY | DATA_SMALL| DATA_FULL);

   //특정채널의이벤트허용여부설정.
   AxaiEventSetChannelEnable(CHANNEL_NO,ENABLE);

   //TimerTriggerMode에서 A/D 변환을시작한다.
   StartAnalogInput();
}

void ThreadProc(LPVOID lpData)
{
   CExAIOEventDlg *pDlg = (CExAIOEventDlg *)lpData;

   while(pDlg->m_bThread)
   {
      if (WaitForSingleObject(pDlg->m_hEventEvent, INFINITE) == WAIT_OBJECT_0)
      {
         //lChannelNo :이벤트발생입력채널번호 ( 0, 1, 2, 3).
```

```
        //dwStatus :이벤트발생 Flag.
        CString IntMessage;
        long lpDataLength;
        DWORD dwDigit;

        long lChannelNo;
        DWORD dwStatus;
        DWORD Code = AxaiEventRead(&lChannelNo, &dwStatus);

        switch(dwStatus)
        {
           case AIO_EVENT_DATA_UPPER:       //  상한값을넘었을경우.
              IntMessage.Format("EVENT : %d번입력채널의데이터가상한값을넘었습니다.",lChannelNo);
              pDlg->m_LIST_Message.InsertItem(0,IntMessage);
           break;
           case AIO_EVENT_DATA_LOWER:       //하한값보다적을경우.
              IntMessage.Format("EVENT : %d번입력채널의데이터가하한값보다적습니다.",ChannelNo);
              pDlg->m_LIST_Message.InsertItem(0,IntMessage);
           break;
           case AIO_EVENT_DATA_FULL:        //데이터가가득찼을경우.
              g_bSaveBuffer = FALSE;
              IntMessage.Format("EVENT : %d번입력채널의데이터가가득찼습니다.",lChannelNo);
              pDlg->m_LIST_Message.InsertItem(0,IntMessage);

              //타이머트리거의 A/D변환을종료시킨다.
              AxaiHwStopMultiChannel();
              IntMessage.Format("EVENT : %d번입력채널의 A/D 변환을종료합니다.",lChannelNo);
              pDlg->m_LIST_Message.InsertItem(0,IntMessage);

              //버퍼에저장된데이터를읽는다.
              g_bReadBuffer = TRUE;
           break;
           case AIO_EVENT_DATA_EMPTY:       //데이터가비었을경우.
              g_bReadBuffer = FALSE;
              IntMessage.Format("EVENT : %d번입력채널의데이터가없습니다.",lChannelNo);
              pDlg->m_LIST_Message.InsertItem(0,IntMessage);
           break;
        }
     }
   }
}
```

```
// Ex5_AXA_Event.cpp :
// 콜백방식예제.

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

```
#include "stdio.h"
#include "time.h"

void __stdcall CallBack_ReadChannelData(long lActiveNo, DWORD uFlag)
{
   //지정한버퍼상한값에도달시이벤트를받을콜백함수.
   long AdcChannelNo[4] = {0,1,2,3};
   if(uFlag == AIO_EVENT_DATA_UPPER)// 상한값도달이벤트인경우.
   {
      GraphData graphData;
      double dVolt[4][2048];
      long lDataLength = 0;
      longlCount = 2048;
      int i;
      for(i = 0; i<4; i++)
      {
         DWORD dwReturn = AxaiHwReadDataLength(AdcChannelNo[i], &lDataLength);
         if(dwReturn!=0)
         {
            return ;
         }
         long lTemp;
         if(lDataLength > 2048)
            lTemp = 2048;
         else
            lTemp = lDataLength;
         if(lCount > lTemp)
            lCount = lTemp;
      }
      for(i = 0; i<4; i++)
      {
         dwReturn = AxaiHwReadSampleVoltage(AdcChannelNo[i], &lCount, dVolt[i]);
         if(dwReturn!=0)
         {
            return ;
         }
      }
      /* 여기서 lCount 만큼채널별로수집된데이터를처리  이부분의코드가샘플링주기의 512배를넘어서는시간지연이걸
         리는작업을수행시에는  콜백방식이아닌윈도우메시지방식이나이벤트처리방식을반드시사용해야함.*/
   }
}

void main(void)
{
   // 라이브러리를초기화한다.
   // 7은 IRQ를뜻한다. PCI는자동으로 IRQ가설정된다.
   DWORD Code = AxlOpen(7);
   if (Code == AXT_RT_SUCCESS)
   {
```

```
    printf("라이브러리가초기화되었습니다.\n");

    //AIO 모듈이있는지검사.
    DWORD dwStatus;
    Code = AxaInfoIsAIOModule(&dwStatus);
    if(Code == AXT_RT_SUCCESS)
    {
       if(dwStatus == STATUS_EXIST)
       {
          long AdcChannelNo[4] = {0,1,2,3};

          //이미입력을받고있다면종료시킨다.
          AxaiHwStopMultiChannel(-1);
          AxaiInterruptSetModuleMask(0, FIFO_HALF_FULL);
          AxaiEventSetChannel(AdcChannelNo[0], NULL, NULL,
             (AXT_EVENT_PROC)CallBack_ReadChannelData, NULL);                //이벤트를받을
             콜백함수지정.
          AxaiEventSetChannelMask(AdcChannelNo[0], DATA_MANY);//상한값에도달시.
          AxaiEventSetChannelEnable(AdcChannelNo[0], ENABLE);
          //이벤트호출 Enable.
          //여러채널의입력전압범위를 -10V ~ 10V로설정.
          AxaiSetMultiRange(4,AdcChannelNo,-10,10);

          //여러채널의버퍼의상한값과하한값을설정.
          AxaiHwSetMultiLimit(4,AdcChannelNo,0,1000);

          //0번모듈의트리거모드를 Timer Trigger로설정.
          AxaiSetTriggerMode(0,TIMER_MODE);

          //특정모듈의셈플링주파수를 1000Hz로설정한다.
          AxaiHwSetSampleFreq(0,1000);

          //여러입력채널에버퍼가 Full로찼을때새로운데이터가유지되도록설정.
          AxaiHwSetMultiBufferOverflowMode(4,AdcChannelNo,NEW_DATA_KEEP);

          // 모든입력채널에각각 2000개씩의버퍼를할당하여 A/D변환을시작한다.
          long lBuffSize = 2000;
          AxaiHwStartMultiChannel(4,AdcChannelNo,lBuffSize);
          printf("Timer Trigger Mode 의시작\n");

          //무한루프를돌면서아날로그입력을받는다.
          printf("[INFORMATION]************************************ \n");
          printf("[ESC] : Exit \n");
          printf("*************************************************** \n");

          BOOL fExit = FALSE;
          int ReadMode = 0;
          while (!fExit)          // 무한루프.
          {
```

```
            if (kbhit())          // 아무키나누르면.
            {
               int ch = getch();
               switch (ch)
               {
                  case 27:        // Esc key.
                     fExit = TRUE;
                     //Timer Trigger Mode 의정지.
                     AxaiHwStopMultiChannel(-1);
                     printf("Timer Trigger Mode 의정지\n");
                     break;
                  default:
                     break;
               }
            }
            Sleep(100);
         } // end of while
      }
      else
         printf ("AxaInfoIsAIOModule() : ERROR ( NOT STATUS_EXIST ) code 0x%x\n",Code);
   }
   else
      printf ("AxaInfoIsAIOModule() : ERROR ( Return FALSE )  code 0x%x\n",Code);
}
else
   printf ("AxlOpen() : ERROR  code 0x%x\n",Code);

// 라이브러리를종료한다.
if (AxlClose())
   printf("\n라이브러리가종료되었습니다.\n");
else
   printf("\n라이브러리가정상적으로종료되지않았습니다.\n");
}
```

인터럽터마스크의설정

연속적으로변화하는 아날로그 신호감시를 할 때 하드웨어 모듈의 FIFO에서 메모리 버퍼로 복사하는 시점을 AxaiInterruptSetModuleMask 함수를 사용하여 내부에서 인터럽터 발생 시점을 조정하여 지정 할 수 있다. SIO-AI4RB는 하드웨어 저장 메모리가 RAM 방식이므로 하드웨어 인터럽트를 사용하지않는다.

인터럽터 마스크의 종류는 다음과 같다. 기본 셋팅값은 FIFO_HALF_FULL 이다.

종류	설명
FIFO_HALF_FULL	모듈내의 FIFO가 HALF이상 찼을 경우 내부 인터럽터 발생하며 메모리 버퍼로 데이터 복사가 이루어 진다.(고속 샘플링시 적용)
SCAN_END	셋팅된 채널 모두 ADC 변환이 한번 이루어 질 때 마다 인터럽터가 발생하며 메모리 버퍼로 데이터 복사가 이루어 진다. (저속 샘플링시 - 트리거 신호가 100~200Hz 이하의 저속일 경우 사용 추천)

```
//모듈의하드웨어인터럽터 Mask를결정한다.

Code = AxaiInterruptSetModuleMask(0, SCAN_END);
// External 또는 Timer Trigger Mode 에서저속샘플링(100~200Hz이하)일경우 SCAN_END 사용.
```

만약, 고속 샘플링인 상태인데 SCAN_END로 인터럽터 마스크를 지정하게 되면 한번의 ADC 변환이 생길 때마다 인터럽터가 발생하게 됨으로 인해 오버헤드가 심해진다. 반대로 저속 샘플링에서 FIFO_HALF_FULL로 마스크를 지정하게 되면 FIFO의 HALF 까지 데이터가 차야만 메모리 버퍼로 데이터 복사가 이루어지기 때문에 데이터의 시간 지연이 심해진다.

예를 들어 10Hz 의 주파수로 하나의 채널을 샘플링을 할 경우 FIFO_HALF_FULL로 셋팅되어 있다면, SIO_AI16HB의 FIFO 크기는 1024 개이므로 HALF인 512 개의 데이터가 수집되어야만 메모리 버퍼로 복사되므로 약 51초가량의 시간이 지체되어야 메모리 버퍼에서 수집된 데이터를 얻어올 수 있게 된다. 그러나 SCAN_END로 지정하게 되면, 한번의 샘플링이 끝나는 0.1초만에 메모리 버퍼에서 수집된 첫 번째 데이터를 얻어 올 수 있다. 또한, 만약 100Khz의 주파수로 하나의 채널을 샘플링 할 경우에 SCAN_END로 지정되어 있다면, 10 usec 마다 데이터 복사가 이루어 지는 무리한 오버헤더가 발생하게 된다. 이 경우 FIFO_HALF_FULL로 셋팅하면 약 5msec 마다 한꺼번에 512개씩 데이터 복사가 이루어지므로 오버헤더로 인한 부하가 없어진다.

아날로그출력

아날로그 출력 모듈은 디지털 데이터를 바탕으로 연속적인 전압 형태의 아날로그 데이터로 출력해주는 모듈이다. 출력 모듈은 출력 전압값의 범위를 정해주고, 해당 모듈의 해상도에따라 알맞은 전압을 출력해 주게 된다.

출력모듈설정

사용자가 지정한 출력 채널의 출력 전압 범위를 설정하는데, 이 설정에 따라 출력 전압의 정밀도가 변하므로 적절히 설정해야 한다. 예를 들어 12Bit 아날로그 출력모듈인 SIO-AO4R 사용한다고 할 때, 기준 전압을 0V-5V로 설정하게 되면 이론적으로 사용자는 5V / 4096 해서 1.22mV 단위로 제어할 수 있는 반면에 기준 전압을 -10V - 10V로 설정하게 되면 20V / 4096 해서 4.8mV 단위로 밖에 제어할 수 없다는 것이다. 따라서 용도에 맞게 기준 전압을 적절히 설정해야 정밀한 제어를 할 수 있다.

최소 출력 전압은 아날로그 출력의 기준 전압 중 최소 전압을 의미한다. 즉 사용자는 이 전압보다 작은 전압을 출력할 수 없다. 그리고 사용자가 설정한 값은 하드웨어가 지원하는 최소 기준 전압을 포함하는 가장 근사치 값으로 설정된다. SIO-AO4R의 경우 최소 기준 전압으로 -10V, -5V, 0V를 제공하므로 사용자가 -8V를 입력하면 -10V로 자동 설정된다.

최대 출력 전압은 아날로그 출력의 기준 전압 중 최대전압을 의미한다. 즉 사용자는 이 전압보다 높은 전압을 출력할 수 없다. 그리고 사용자가 설정한 값은 하드웨어가 지원하는 최대 기준 전압을 포함하는가장근사치값으로설정된다. 16Bit Resolution을 가지는 일체형 보드 PCI-AI16HR 의 2채널 Output 모듈 SIO-AO2HB는 -10V~10V의 고정 출력 전압 범위를 가진다. 최소 측정단위는 20V / 65536 으로 약 0.305mV 이다.

Module	최소전압	최대전압	최소측정단위
SIO-AO4R	-10V	10V	4.88mV
	-5V	5V	2.44mV
	0V	10V	2.44mV
	0V	5V	1.22mV
SIO-AO2HB(PCI-AI16HR)	-10V	10V	0.305mV

AxaoSetRange: 기준전압을 설정하기 위해서는 AxaoSetRange 함수를 사용하며, '0'번 채널의 출력 전압의 범위를 -8V에서 8V로 설정하고자 한다면 다음과 같이 한다. PCI-AI16HR 보드의 Output 2채널(SIO-AO2HB)은 출력 전압 범위가 고정되어 있으므로 함수 수행 시 AXT_RT_AIO_INVALID_USE 에러가 리턴된다.

```
//0번채널의출력전압의범위를 -8V에서 8V로설정.
AxaoSetRange(0,-8,8);
```

AxaoGetRange: 위와 같이 설정하였을 경우 자동으로 -10V와 10V가 선택되어 설정된다. 이와 같이 실제로 하드웨어에서 지원하지 않는 기준전압을 설정했을 때 설정한 값과 실제 설정되는 값이 다를 수 있다. 어떠한 값이 설정되었는지를 알고자 한다면 AxaoGetRange 함수를 사용하여 확인할 수 있다. 만약 0번 채널에 실제 설정된 값을 알고자 한다면 다음과 같이 한다.

```
// 0번출력채널의출력전압범위를확인한다.
double dVmin;
double dVmax;
AxaoGetRange (0, &dVmin, &dVmax);
```

아날로그출력

AxaoWriteVoltage: 출력전압의 범위를 설정하고 나면, 사용자가 지정한 출력 채널에 특정 전압을 출력할 수 있다. 특정 전압을 출력하기 위해서는 AxaoWriteVoltage 라는 함수를 사용하며, 사용자가 원하는 전압을 입력하게 되면 그 전압이 출력되며 새로운 값이 설정되기 전에는 그 전압이 계속 유지된다. 이 출력 전압은 AxaoGetRange에서 설정한 전압 폭을 넘을 수 없다. 만약 0번 채널에 5V를 출력고자 한다면 다음과 같이 한다.

```
//0번채널에 5V를출력한다.
AxaoWriteVoltage (0,5);
```

AxaoReadVoltage: 현재 출력되고 있는 값을 알고자 할 때는 AxaoReadVoltage 함수를 사용하여 아래와 같이 한다.

```
//0번채널에출력되고있는값을확인한다.
Double dVolt;
AxaoReadVoltage(0, dVolt);
```

만약 사용자가 특정한 파형의 아날로그 출력을 원할 경우 위에서 언급한 AxaoWriteVoltage함수를 이용하여 생성하여야 한다. 예를 들어 sin파형의 신호를 출력하고자 한다면 다음과 같이 한다.

```
// 주기가 1 이고크기가 10인 Sin파형을생성한다.
double OutputValue;
double SampleCount = 1000;   //Sampling 1000으로설정.
double Magnitude = 10;    //크기 10V로설정.
while (1)                 // 무한루프.
{
   //주기가 1이고, 크기가 10V인 Sin파형을출력한다.
   for(int i=0;i< SampleCount;i++)
   {
      OutputValue = sin((2*3.141592/ SampleCount)*(double) i ) * Magnitude;
      AxaoWriteVoltage(0,OutputValue);
      Sleep(1);               //1msec Sleep * SampleCount(1000) = 1초주기파형.
   }
}
/*상기코드에서 Sleep(1)이실제시스템에서정확히 1mSec를 Sleep하지않기때문에좀더정확한파형출력을위해서는시스템및
   OS에따라정확한 Sleep Time을구현할수있어야한다. 윈도우시스템에서는
   QueryPerformanceFrequency/QueryPerformanceCounter 등의 Windows API를사용하여 CPU 주파수와클럭수
   를측정하여상당히정확한시간지연을구현할수있다.*/
```

```
// Ex6_AXA_AnalogOut.cpp : Defines the entry point for the console application.
// 아날로그출력모듈을설정하고출력을내보낸다.

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

```
void main(void)
{
   // 라이브러리를초기화한다.
   // 7은 IRQ를뜻한다. PCI는자동으로 IRQ가설정된다.
   DWORD Code = AxlOpen(7);
   if (Code == AXT_RT_SUCCESS)
   {
      printf("라이브러리가초기화되었습니다.\n");

      //AIO 모듈이있는지검사.
      DWORD dwStatus;
      Code = AxaInfoIsAIOModule(&dwStatus);
      if(Code == AXT_RT_SUCCESS)
      {
         if(dwStatus == STATUS_EXIST)
         {
            //출력모듈을초기화한다.
            //0번채널의출력전압의범위를 -8V에서 8V로설정.
            AxaoSetRange(0,-8,8);
            //설정된출력전압범위를확인한다.
            // 0번출력채널의출력전압범위를확인한다.
            double dVmin;
            double dVmax;
            AxaoGetRange(0, &dVmin, &dVmax);

            //무한루프를돌면서아날로그출력을내보낸다.
            printf("[INFORMATION]*************** \n");
            printf("출력모듈의전압범위 : %.1f ~ %.1f\n",dVmin,dVmax);
            printf("[ESC] : Exit \n");
            printf("[1] : -10V 출력\n");
            printf("[2] :  -8V 출력\n");
            printf("[3] :  -4V 출력\n");
            printf("[4] :   0V 출력\n");
            printf("[5] :   4V 출력\n");
            printf("[6] :   8V 출력\n");
            printf("[7] :  10V 출력\n");
            printf("**************************** \n");

            BOOL fExit = FALSE;
            while (!fExit)          // 무한루프.
            {
               if (kbhit())         // 아무키나누르면.
               {
                  int ch = getch();
                  switch (ch)
                  {
                     case 27:       // Esc key.
                        fExit = TRUE;
                     break;
```

```
                    //키에해당하는전압을 0번채널에서출력.
                    case '1':   AxaoWriteVoltage(0,-10);
                    break;  //-10V 출력.
                    case '2':   AxaoWriteVoltage(0,-8);
                    break;  // -8V 출력.
                    case '3':   AxaoWriteVoltage(0,-4);
                    break;  // -4V 출력.
                    case '4':   AxaoWriteVoltage(0,0);
                    break;  //  0V출력.
                    case '5':   AxaoWriteVoltage(0,4);
                    break;  //  4V출력.
                    case '6':   AxaoWriteVoltage(0,8);
                    break;  //  8V출력.
                    case '7':   AxaoWriteVoltage(0,10);
                    break;  // 10V 출력.
                 }
              }
           }
        }
        else
           printf ("AxaInfoIsAIOModule() : ERROR ( NOT STATUS_EXIST )  code 0x%x\n",Code);
     }
     else
        printf ("AxaInfoIsAIOModule() : ERROR ( Return FALSE )  code 0x%x\n",Code);
  }
  else
     printf ("AxlOpen() : ERROR  code 0x%x\n",Code);

  // 라이브러리를종료한다.
  if (AxlClose())
     printf("\n라이브러리가종료되었습니다.\n");
  else
     printf("\n라이브러리가정상적으로종료되지않았습니다.\n");
}
```

```
// Ex7_AXA_AnalogOutSin.cpp : Defines the entry point for the console application.
// Sin파형을내보낸다.

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

void main(void)
{
  // 라이브러리를초기화한다.
```

```
   // 7은 IRQ를뜻한다. PCI는자동으로 IRQ가설정된다.
   DWORD Code = AxlOpen(7);
   if (Code == AXT_RT_SUCCESS)
   {
      printf("라이브러리가초기화되었습니다.\n");
      //AIO 모듈이있는지검사.
      DWORD dwStatus;
      Code = AxaInfoIsAIOModule(&dwStatus);
      if(Code == AXT_RT_SUCCESS)
      {
         if(dwStatus == STATUS_EXIST)
         {
            //출력모듈을초기화한다.
            //0번채널의출력전압의범위를 -10V에서 10V로설정.
            AxaoSetRange(0,-10,10);
            //무한루프를돌면서아날로그출력을내보낸다.
            double OutputValue;
            double SampleCount = 1000;    //Sampling 1000으로설정.
            double Magnitude = 10;        //크기 10V로설정.
            while (1)                     // 무한루프.
            {
               //주기가 1초이고, 크기가 10V인 Sin파형을출력한다.
               for(int i=0;i< SampleCount;i++)
               {
                  OutputValue = sin((2*3.141592/ SampleCount)*(double) i ) * Magnitude;
                  AxaoWriteVoltage(0,OutputValue);
                  Sleep(1);               //1msec Sleep * SampleCount(1000) = 1초.
               }
            }
         }
         else
            printf ("AxaInfoIsAIOModule() : ERROR ( NOT STATUS_EXIST ) code 0x%x\n",Code);
      }
      else
         printf ("AxaInfoIsAIOModule() : ERROR ( Return FALSE )  code 0x%x\n",Code);
   }
   else
      printf ("AxlOpen() : ERROR  code 0x%x\n",Code);
   // 라이브러리를종료한다.
   if (AxlClose())
      printf("\n라이브러리가종료되었습니다.\n");
   else
      printf("\n라이브러리가정상적으로종료되지않았습니다.\n");
}
```

AIO Command 매뉴얼정보

본 매뉴얼은 아진엑스텍의 AIO 보드/모듈을 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.

헤더파일

C++

AXA.h

Visual Basic

AXA.bas

Delphi

AXA.pas

함수용어

본매뉴얼의함수이름

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

라이브러리 함수 Prefix

접두분류	설명
Axa	Axa 전용 함수임을 나타낸다. Axa로 시작되는 함수들은 모두 Axa.h에 정의 되어 있다.
AxaInfo	보드및모듈정보확인함수
Axai	입력 관련 함수
AxaiInfo	입력모듈정보검색함수
AxaiEvent	입력채널 이벤트설정및확인함수
AxaiInterrupt	입력모듈인터럽터설정및확인함수
AxaiSw	Software Timer (Normal Trigger Mode) 방식 관련 함수
AxaiHw	Hardware Timer(Timer Trigger + External Trigger Mode) 방식 관련 함수
Axao	출력 관련 함수
AxaoInfo	출력모듈정보검색함수
Set	칩의 레지스터나 변수들을 설정한다.
Get	Set과 한 쌍으로 Set함수에 의해 설정된 변수값을 확인하거나 칩 레지스터의 상태를 읽는다.
Multi	여러 채널을 동시에 처리한다.
Read	상태 및 값을 읽는다.
Write	값을 쓴다.

본매뉴얼의인자이름

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

long lChannelNo :채널 번호는 입력 채널 번호와 출력 채널 번호로 나뉘어 진다.
입력 채널 번호는 첫 번째 보드의 첫 번째 입력 모듈부터 오름차순으로 자동 정렬된다.
출력 채널 번호도 첫 번째 보드의 첫 번째 출력 모듈부터 오름차순으로 자동 정렬된다.
입출력 채널 번호는 '0'부터 시작한다.

PCI-AI16HR 보드 두 개 장착 시 모듈 번호와 채널 번호 할당된 결과

모듈번호	입력/출력	채널인덱스
0	입력(SIO-AI16HB)	0~15
1	출력(SIO-AO2HB)	0~1
2	입력(SIO-AI16HB)	16~31
3	출력(SIO-AO2HB)	2~3

AIO Command Quick List

Trigger Mode에따른관련함수그룹

Trigger Mode 설정 (AxaiSetTriggerMode 참조)

Enum	Value	Explanation
NORMAL_MODE	1	사용자가 원하는 시점에 A/D변환하는 Software Trigger 방식 (기본 값)
TIMER_MODE	2	H/W의 내부 클럭을 이용해서 A/D변환하는 Trigger 방식
EXTERNAL_MODE	3	외부 입력단자의 클럭을 이용해서 A/D변환하는 Trigger 방식

Trigger Mode 에 따른 관련 함수

Trigger Mode	중요관련함수
Normal Trigger Mode	AxaiSwReadVoltage, AxaiSwReadMultiVoltage
Timer Trigger Mode	(함수 그룹 : AxaiHw.., AxaiEvent..,AxaiInterrupt..) AxaiEventSetChannel AxaiHwSetSampleFreq, AxaiHwSetSamplePeriod AxaiHwStartMultiChannel, AxaiHwStartMultiFilter, AxaiHwStopMultiChannel, AxaiHwSetLimit, AxaiHwReadDataLength AxaiHwReadSampleVoltage AxaiHwSetMultiAccess(Immediate Mode), AxaiHwStartMultiAccess(Immediate Mode)Etc..
External Trigger Mode	Timer Trriger Mode와 동일하나 Immediate Mode 함수 와 AxaiHwSetSampleFreq / AxaiHwSetSamplePeriod 함수는 해당 사항 없음.

보드및모듈정보

Function	Description
AxaInfoIsAIOModule	시스템에 AIO 제품 군이 있는지 확인한다.
AxaInfoGetModuleNo	AIO 모듈번호를 확인한다.
AxaInfoGetModuleCount	시스템에 장착된 모든AIO 모듈의 개수를 확인한다.
AxaInfoGetInputCount	지정한 AIO 제품의 입력 채널 수를 확인한다.
AxaInfoGetOutputCount	지정한 AIO 제품의 출력 채널 수를 확인한다.
AxaInfoGetChannelNoOfModuleNo	지정한 모듈의 첫 번째 채널 번호를 확인한다.
AxaInfoGetModule	지정한 모듈 번호로 베이스 보드 번호, 모듈 위치, 모듈 ID를 확인한다.

입력모듈정보검색

Function	Description
AxaiInfoGetModuleNoOfChannelNo	지정한 아날로그 입력 채널 번호로 모듈 번호를 확인한다.
AxaiInfoGetChannelCount	시스템에 장착된 아날로그 입력 모듈의 전체 채널 개수를 확인한다.

입력모듈인터럽터/채널이벤트설정및확인

Function	Description
AxaiEventSetChannel	지정한 입력 채널에 수집된 데이터 처리를 위한 이벤트 수신 방법을 설정한다.
AxaiEventSetChannelEnable	지정한 입력 채널에 수집된 데이터 처리를 위한 이벤트 사용 여부를 설정한다.
AxaiEventGetChannelEnable	지정한 입력 채널의 수집된 데이터 처리를 위한 이벤트 사용 여부를 확인한다.
AxaiEventSetMultiChannelEnable	지정한 여러 입력 채널에 수집된 데이터 처리를 위한 이벤트 사용 여부를 설정한다.
AxaiEventSetChannelMask	지정한 입력 채널에 수집된 데이터 처리를 위한 이벤트 종류를 설정한다.
AxaiEventGetChannelMask	지정한 입력 채널에 수집된 데이터 처리를 위해 설정한 이벤트 종류를 확인한다.
AxaiEventSetMultiChannelMask	지정한 여러 입력 채널에 이벤트 종류를 설정한다.
AxaiEventRead	수집된 데이터 처리를 위한 이벤트 발생 시 이벤트 발생 입력 채널 번호를 확인한다.
AxaiInterruptSetModuleMask	지정한 모듈에인터럽트발생 소스를 설정한다.
AxaiInterruptGetModuleMask	지정한 모듈에 설정한 인터럽트소스를 확인한다.

입력모듈파라미터설정및확인

Function	Description
AxaiSetRange	지정한 입력 채널에 입력 전압 범위를 설정한다.
AxaiGetRange	지정한 입력 채널의 입력 전압 범위를 확인한다.
AxaiSetRangeModule	지정한 입력 모듈의 입력 전압 범위를 설정한다. (RTEX-AI16F 전용)
AxaiGetRangeModule	지정한 입력 모듈의 입력 전압 범위를 확인한다. (RTEX-AI16F 전용)
AxaiSetMultiRange	지정한 여러 입력 채널에 허용 입력 전압 범위를 설정한다.
AxaiHwSetBufferOverflowMode	지정한 입력 채널에 버퍼가 Full로 찼을 때 데이터 관리 방식을 설정한다.
AxaiHwGetBufferOverflowMode	지정한 입력 채널이 버퍼가 Full로 찼을 때 데이터 관리 방식을 확인한다.
AxaiHwSetMultiBufferOverflowMode	지정한 여러 입력 채널에 버퍼가 Full로 찼을 때 데이터 관리 방식을 설정한다.
AxaiHwSetLimit	지정한 입력 채널에 버퍼의 상한 값과 하한 값을 설정한다.
AxaiHwGetLimit	지정한 입력 채널에 버퍼의 상한 값과 하한 값을 확인한다.
AxaiHwSetMultiLimit	지정한 여러 입력 채널에 버퍼의 상한 값과 하한 값을 설정한다.

AxaiSetTriggerMode	지정한 입력 모듈에 A/D변환 방식을 설정한다.
AxaiGetTriggerMode	지정한 입력 모듈에 설정한 A/D변환 방식을 확인한다.
AxaiHwSetSampleFreq	지정한 입력 모듈에 샘플링 주기 단위를 주파수로 설정한다.
AxaiHwGetSampleFreq	지정한 입력 모듈에 설정한 샘플링 주파수를 확인한다.
AxaiHwSetSamplePeriod	지정한 모듈에 샘플링 주기 단위를 시간(uSec)으로 설정한다.
AxaiHwGetSamplePeriod	지정한 모듈에 샘플링 시간(uSec)을 확인한다.
AxaiSetModuleOffsetValue	지정한 입력모듈의 Offset을 mV단위로 설정한다.
AxaiGetModuleOffsetValue	지정한 입력모듈에 설정한 Offset을 확인한다

입력모듈 A/D 변환

Function	Description
AxaiSwReadVoltage	지정한 입력 채널에 아날로그 입력 값을 전압으로 반환한다.
AxaiSwReadDigit	지정한 입력 채널에 아날로그 입력 값을 Digit 값으로 반환한다.
AxaiSwReadMultiVoltage	지정한 여러 입력 채널에 아날로그 입력 값을 전압 값으로 반환한다.
AxaiSwReadMultiDigit	지정한 여러 입력 채널에 아날로그 입력 값을 Digit 값으로 반환한다.
AxaiHwStartMultiChannel	지정한 여러 입력 채널에 H/W타이머를 이용한 A/D변환을 시작한다.
AxaiHwStartMultiFilter	지정한 여러 입력 채널에 A/D변환 시작 후 지정한 개수만큼의 평균값을버퍼에 저장한다.
AxaiHwStopMultiChannel	지정한 여러 입력 채널에 H/W타이머를 이용한 연속 A/D변환을 중지한다.
AxaiHwSetMultiAccess	지정한 여러 입력 채널에 Immediate모드를 사용하기 위한 설정한다.
AxaiHwStartMultiAccess	Immediate모드로 설정되어 수집된 데이터를 확인한다.
AxaiHwReadSampleVoltage	지정한 입력 채널에 H/W타이머를 이용하여 A/D변환된 데이터를 전압 값으로 확인한다.
AxaiHwReadSampleDigit	지정한 입력 채널에 H/W타이머를 이용하여 A/D변환된 데이터를Digit 값으로 확인한다.
AxaiHwStartSingleChannelAdc	지정한 입력 채널에 H/W타이머를 이용한 A/D변환을 시작한다
AxaiHwStopSingleChannelAdc	H/W타이머를 이용한 연속 신호 A/D변환을 중지한다

입력모듈버퍼상태체크

Function	Description
AxaiHwIsBufferEmpty	지정한 입력 채널의 버퍼에 데이터 존재 유무를 확인한다.
AxaiHwIsBufferUpper	지정한 입력 채널의 버퍼에 설정되어 있는 상한 값보다 데이터가 많은지 확인한다.
AxaiHwIsBufferLower	지정한 입력 채널의 버퍼에 설정되어 있는 하한 값보다 데이터가 적은지 확인한다.

AxaiHwReadDataLength	지정한 입력 채널의 버퍼에 저장되어 있는 데이터 개수를 확인한다.

Extern Trigger 함수

Function	Description
AxaiExternalStartADC	지정한 입력모듈의 선택된 채널들의 외부 트리거 모드를 시작한다.
AxaiExternalStopADC	지정한 입력모듈의 외부트리거 모드를 정지한다.
AxaiExternalReadFifoStatus	지정한 입력모듈의 Fifo상태를 반환한다.
AxaiExternalReadVoltage	지정한 입력모듈의 외부신호에 의해 변환된 A/D값을 읽어온다.

출력모듈정보검색

Function	Description
AxaoInfoGetModuleNoOfChannelNo	지정한 출력 채널 번호로 모듈 번호를 확인한다.
AxaoInfoGetChannelCount	시스템에 장착된 아날로그 출력 모듈의 전체 채널 개수를 확인한다.

출력모듈설정및확인

Function	Description
AxaoSetRange	지정한 출력 채널에 출력 전압 범위를 설정한다.
AxaoGetRange	지정한 출력 채널에 설정한 출력 전압 범위를 확인한다.
AxaoSetMultiRange	지정한 여러 출력 채널에 출력 전압 범위를 설정한다.
AxaoWriteVoltage	지정한 출력 채널에 입력한값을 전압으로 출력한다.
AxaoWriteMultiVoltage	지정한 여러 출력 채널에 입력한 값을 전압으로 출력한다.
AxaoReadVoltage	지정한 출력 채널에 출력하고 있는 전압 값을 확인한다.
AxaoReadMultiVoltage	지정한 여러 출력 채널에 출력하고 있는 전압 값을 확인한다.

Define문

AXAAnalog Input/OutputLibrary에서 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_AIO_TRIGGER_MODE

Definition	Value	Explanation
DISABLE_MODE	00	트리거 사용하지 않음
NORMAL_MODE	01	사용자 이벤트에 의한 트리거모드
TIMER_MODE	02	지정한 샘플링 타임에 의한 자동 트리거모드
EXTERNAL_MODE	03	외부 신호에 의한 트리거모드

AXT_AIO_FULL_MODE

Definition	Value	Explanation
NEW_DATA_KEEP	00h	데이터 버퍼 Full시 새로운데이터로 갱신
CURR_DATA_KEEP	01h	데이터 버퍼 Full시 기존데이타 유지

AXT_AIO_EVENT_MASK

Definition	Value	Explanation
DATA_EMPTY	01h	데이터 버퍼가 비워질 때

DATA_MANY	02h	데이터 버퍼가 지정한 개수보다 많아질 때
DATA_SMALL	04h	데이터 버퍼가 지정한 개수보다 적어질 때
DATA_FULL	08h	데이터 버퍼가 가득찰 때

AXT_AIO_INTERRUPT_MASK

Definition	Value	Explanation
ADC_DONE	00	A/D Conversion이 완료되었을 때
SCAN_END	01	지정된 채널이 모두 A/D Conversion이 완료되었을 때
FIFO_HALF_FULL	02	H/W 버퍼에 데이터가 반이상 찼을 때
NO_SIGNAL	03	이벤트 발생없음

AIO Command Function List

보드및모듈정보

Function	Description
AxaInfoIsAIOModule	AIO 모듈이 있는지 확인한다.
AxaInfoGetModuleNo	AIO 모듈 Number를 확인한다
AxaInfoGetInputCount	지정한 모듈의 입력 채널 수를 확인한다.
AxaInfoGetOutputCount	지정한 모듈의 출력 채널 수를 확인한다.
AxaInfoGetModuleCount	AIO 모듈의 개수를 확인한다.
AxaInfoGetChannelNoOfModuleNo	지정한 모듈의 첫 번째 채널 번호를 확인한다.
AxaInfoGetChannelNoAdcOfModuleNo	지정한 모듈의 첫 번째 입력 채널 번호를 확인 한다.
AxaInfoGetChannelNoDacOfModuleNo	지정한 모듈의 첫 번째 출력 채널 번호를 확인 한다.
AxaInfoGetModuleStatus	해당 모듈이 제어가 가능한 상태인지 반환한다.
AxaInfoGetModule	지정한 모듈 번호로 베이스 보드 번호, 모듈 위치, 모듈 ID를 확인한다.

AxaInfoIsAIOModule

Purpose

시스템에 AIO 제품 군이 있는지 확인합니다.

Format

C++

```
DWORD AxaInfoIsAIOModule(DWORD *upStatus);
```

Visual Basic

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

Delphi

```
Function AxaInfoIsAIOModule(upStatus : PDWord) : DWord; stdcall;
```

C#

```
uint AxaInfoIsAIOModule(ref uint upStatus);
```

Parameters

[in/out] Name	[Init Value] Explanation
[out]upStatus	AIO모듈의 존재여부: AXT_EXISTENCE - [00h] AIO모듈이 없음 - [01h] AIO모듈이 존재함

Return Values

[0000] AXT_RT_SUCCESS :함수실행성공
[1053] AXT_RT_NOT_OPEN :라이브러리초기화실패
* See error code Table for more information on status error codes

Description

현재 시스템에 장착된 AIO제품군 존재 여부 확인이 가능합니다.

※참고사항

AIO 모듈 제품뿐 아니라 싱글 보드 타입의 존재 여부도 확인 가능합니다.

Example

C++

```
#include "Axl.h"        // 라이브러리추가
DWORD dwStatus;

// AIO모듈이있는지확인합니다.
AxaInfoIsAIOModule(&dwStatus);
if(dwStatus == STATUS_EXIST)
  AfxMessageBox("AIO 모듈이존재합니다.");
else
  AfxMessageBox("AIO 모듈이존재하지않습니다.");
```

Visual Basic

```
'라이브러리사용을위해 AXA.bas와 AXL.bas, AXHS.bas 모듈을추가합니다.
Dim lStatus As Long

' AIO 모듈이있는지확인합니다.
AxaInfoIsAIOModule lStatus
If lStatus = STATUS_EXIST Then
  MsgBox "AIO 모듈이존재합니다.", vbOKCancel
Else
  MsgBox "AIO 모듈이존재하지않습니다.", vbOKCancel
End If
```

Delphi

```
Uses
Axl, AXHS, AXA   { 라이브러리추가 }
var
  dwStatus : DWord;

begin
  { AIO모듈이있는지확인합니다. }
  AxaInfoIsAIOModule(@dwStatus);
  if(dwStatus = STATUS_EXIST) then
    Application.MessageBox ('AIO 모듈이존재합니다.', 'Ajinextek', MB_OK)
  else
    Application.MessageBox ('AIO 모듈이존재하지않습니다.', 'Ajinextek', MB_OK)
end;
```

C#

```
uintuStatus;

// AIO모듈이있는지확인합니다.
CAXA.AxaInfoIsAIOModule(ref uStatus);
if(uStatus == STATUS_EXIST)
  MessageBox.Show("AIO 모듈이존재합니다.");
else
  MessageBox.Show("AIO 모듈이존재하지않습니다.");
```

See Also

AxaInfoIsAIOModule, AxaInfoGetModuleNo, AxaInfoGetModuleCount, AxaInfoGetInputCount, AxaInfoGetOutputCount, AxaInfoGetChannelNoDacOfModuleNo, AxaInfoGetChannelNoAdcOfModuleNo, AxaInfoGetModule, AxaInfoGetModuleStatus

AxaInfoGetModuleNo

Purpose

AIO 모듈 번호를 확인합니다.

Format

C++

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

Visual Basic

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

Delphi

```
Function AxInfoGetModuleNo(lBoardNo: LongInt; lModulePos:
LongInt;lpModuleNo: PLongInt) : DWord; stdcall;
```

C#

```
uint AxaInfoGetModuleNo(int lBoardNo, int lModulePos, ref int
lpModuleNo);
```

Parameters

[in/out] Name	[Init Value] Explanation
[in]lBoardNo	보드 번호
[in]lModulePos	모듈 위치
[out]lpModuleNo	AIO 모듈 번호

Return Values

[0000]AXT_RT_SUCCESS :함수실행성공
[1101]AXT_RT_INVALID_BOARD_NO :라이브러리초기화실패
[1102]AXT_RT_INVALID_MODULE_POS :유효하지않은모듈번호
* See error code Table for more information on status error codes

Description

현재 시스템에 장착되어 사용 가능한 AIO모듈의 번호를 확인합니다. AIO함수들은 모듈 번호로 입/출력 접점을 제어합니다. 이때, 해당 함수로 사용하고자 하는AIO 모듈이 장착된 베이스 보드의 번호와 모듈의 위치 값을 입력하여 AIO모듈의 번호 확인이 가능합니다.

※참고사항

AIO모듈 번호는 입·출력 모듈에 관계없이 시스템에서 PCI Slot을 인식한 차례로 보드 번호가 할당되며 보드에 장착된 모듈은 보드 번호가 빠른 순으로 차례로 모듈 번호가 할당됩니다. 일체형 AIO 보드인 PCI-AI16HR의 경우 입력 모듈과 출력 모듈이 혼합 조합되어 있는 것과 같습니다.

예) BPHR + SIO-AI4RB + SIO-AO4RB ,PCI-AI16HR

위와 같은 제품의 조합에서 BPHR 의 보드 번호가 0, PCI-AI16HR 의보드 번호가 1 일 때 모듈 번호는 다음의 표와 같다.

Board	Module	Module Number
BPHR	SIO-AI4RB	0
	SIO-AO4RB	1
PCI-AI16HR	SIO-AI16HB	2
	SIO-AO2HB	3

Example

C++

```
// AIO 모듈Number를확인합니다.
Long lModuleNo;
CString strData;

AxaInfoGetModuleNo (0, 0, &lModuleNo);
strData.Format("0번보드의 0번 Pos에있는 AIO 모듈번호는 %d 입니다.", lModuleNo);
AfxMessageBox(strData);
```

Visual Basic

```
' AIO 모듈 Number를확인합니다.
Dim lModuleNo As Long
Dim  strData As String

AxaInfoGetModuleNo 0 0 lCount
strData    = "0번보드의 0번 Pos에있는 AIO 모듈번호는" + CStr(lModuleNo) + "입니다."

MsgBox strData
```

Delphi

```
{ AIO모듈Number를확인합니다. }
var
  lModuleNo: LongInt;
  strData : String;

begin
  AxaInfoGetModuleNo (0, 0, @lModuleNo);
  strData :='0번보드의 0번 Pos에있는 AIO 모듈번호는' + IntToStr(lModuleNo) + '입니다.';

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

C#

```
// AIO 모듈Number를확인합니다.
intnModuleNo;
String strData;
```

```
CAXD.AxaInfoGetModuleNo (0, 0, ref nModuleNo);
strData.Format("0번보드의 0번 Pos에있는 AIO 모듈번호는 %d 입니다.", nModuleNo);
MessageBox.Show(strData);
```

See Also

AxaInfoIsAIOModule, AxaInfoGetModuleNo, AxaInfoGetModuleCount, AxaInfoGetInputCount, AxaInfoGetOutputCount, AxaInfoGetChannelNoDacOfModuleNo, AxaInfoGetChannelNoAdcOfModuleNo, AxaInfoGetModule, AxaInfoGetModuleStatus

AxaInfoGetInputCount

Purpose

지정한 AIO 제품의 입력 채널 수를 확인합니다.

Format

C++

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

Visual Basic

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

Delphi

```
Function AxaInfoGetInputCount(lModuleNo : LongInt; lpCount :
PLongInt) : DWord; stdcall;
```

Parameters

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

Return Values

[0000] AXT_RT_SUCCESS :함수실행성공
[1053] AXT_RT_NOT_OPEN :라이브러리초기화실패
[2101]AXT_RT_AIO_INVALID_MODULE_NO :유효하지않은 AIO 모듈번호
* See error code Table for more information on status error codes

Description

사용자가 지정한 모듈의 입력 채널 수를 확인합니다.

※참고사항

일체형 AIO보드인 PCI-AI16HR은 일체형 싱글 보드 타입이지만 입력 모듈 SIO-AI16HB와 출력 모듈 SIO-AO2HB가 혼합 조합되어 있는 것과 같습니다.

AIO 제품 입출력 접점 수

Board	Module	Input	Output
-	SIO-AI4RB	4	0
	SIO-AO4RB	0	4
PCI-AI16HR	SIO-AI16HB	16	0
	SIO-AO2HB	0	2

Example

C++

```
// 0번째모듈의입력채널개수를확인합니다.
long lCount;
CString   strData;

AxaInfoGetInputCount(0, &lCount);
strData.Format("0번모듈의입력채널개수는 %d개입니다.", lCount);

AfxMessageBox(strData);
```

Visual Basic

```
' 0번째모듈의입력채널개수를확인합니다.
Dim lCount As Long
Dim strData As String

AxaInfoGetInputCount 0, lCount
strData   = "0번모듈의입력채널개수는" + CStr(lCount) + "개입니다."

MsgBox strData
```

Delphi

```
{ 0번째모듈의입력채널개수를확인합니다. }
var
  lCount : LongInt;
  strData : String;

begin
  AxaInfoGetInputCount(0, @lCount);
  strData :='0번모듈의입력채널개수는' + IntToStr(lCount) + '개입니다.';

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

C#

```
// 0번째모듈의입력채널개수를확인합니다.
int  nCount;
String strData;

CAXA.AxaInfoGetInputCount(0, ref nCount);
strData.Format("0번모듈의입력채널개수는 %d개입니다.", nCount);

MessageBox.Show(strData);
```

See Also

AxaInfoIsAIOModule, AxaInfoGetModuleNo, AxaInfoGetModuleCount, AxaInfoGetInputCount, AxaInfoGetOutputCount, AxaInfoGetChannelNoDacOfModuleNo, AxaInfoGetChannelNoAdcOfModuleNo, AxaInfoGetModule, AxaInfoGetModuleStatus

AxaInfoGetOutputCount

Purpose

지정한 AIO 제품의 출력 채널 수를 확인합니다.

Format

C++

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

Visual Basic

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

Delphi

```
Function AxaInfoGetOutputCount(lModuleNo : LongInt; lpCount :
PLongInt) : DWord; stdcall;
```

C#

```
uint AxaInfoGetOutputCount(int lModuleNo, ref int lpCount);
```

Parameters

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

Return Values

[0000] AXT_RT_SUCCESS :함수실행성공
[1053] AXT_RT_NOT_OPEN :라이브러리초기화실패
[2101] AXT_RT_AIO_INVALID_MODULE_NO :유효하지않은 AIO 모듈번호
* See error code Table for more information on status error codes

Description

사용자가 지정한 모듈의 출력 채널 수를 확인합니다.

※참고사항

일체형 AIO 보드인 PCI-AI16HR은 일체형 싱글 보드 타입이지만 입력 모듈 SIO-AI16HB와 출력 모듈 SIO-AO2HB가 혼합 조합되어 있는 것과 같습니다.

AIO 제품 입출력 접점 수

Board	Module	Input	Output
-	SIO-AI4RB	4	0
	SIO-AO4RB	0	4
PCI-AI16HR	SIO-AI16HB	16	0
	SIO-AO2HB	0	2

Example

C++

```
// 0번째모듈의출력채널개수를확인합니다.
long lCount;
CString strData;

AxaInfoGetOutputCount(0, &lCount);
strData.Format("0번모듈의출력채널개수는 %d개입니다.", lCount);

AfxMessageBox(strData);
```

Visual Basic

```
' 0번째모듈의출력채널개수를확인합니다.
Dim lCount As Long
Dim strData As String

AxaInfoGetOutputCount 0, lCount
strData = "0번모듈의출력채널개수는" + CStr(lCount) + "개입니다."

MsgBox strData
```

Delphi

```
{ 0번째모듈의출력채널개수를확인합니다. }
var
  lCount : LongInt;
  strData : String;

begin
  AxaInfoGetOutputCount(0, @lCount);
  strData  :='0번모듈의출력채널개수는' + IntToStr(lCount) + '개입니다.';

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

C#

```
// 0번째모듈의출력채널개수를확인합니다.
int  nCount;
String  strData;

CAXA.AxaInfoGetOutputCount(0, ref nCount);
strData.Format("0번모듈의출력채널개수는 %d개입니다.", nCount);

MessageBox.Show(strData);
```

See Also

AxaInfoIsAIOModule, AxaInfoGetModuleNo, AxaInfoGetModuleCount, AxaInfoGetInputCount, AxaInfoGetOutputCount, AxaInfoGetChannelNoDacOfModuleNo, AxaInfoGetChannelNoAdcOfModuleNo, AxaInfoGetModule, AxaInfoGetModuleStatus

AxaInfoGetModuleCount

Purpose

시스템에 장착된 모든 AIO 모듈의 개수를 확인합니다.

Format

C++

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

Visual Basic

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

Delphi

```
Function AxaInfoGetModuleCount(lpModuleCount : PLongInt) : DWord;
stdcall;
```

C#

```
uint AxaInfoGetModuleCount(ref int lpModuleCount);
```

Parameters

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

Return Values

[0000] AXT_RT_SUCCESS :함수실행성공
[1053] AXT_RT_NOT_OPEN :라이브러리초기화실패
[2051]AXT_RT_AIO_NOT_MODULE : AIO 모듈없음
* See error code Table for more information on status error codes

Description

현재 시스템에 장착된 모든 AIO 모듈의 개수를 확인합니다. 일체형 AIO 보드 제품 사용 시에는 모듈 2 개로 확인됩니다.

※참고사항

일체형 AIO 보드인 PCI-AI16HR 의 경우 입력 모듈과 출력 모듈이 혼합 조합되어 있는 것과 같습니다.

Example

C++

```
// AIO 모듈의개수를확인합니다.
long lCount;
CString strData;

AxaInfoGetModuleCount(&lCount);
strData.Format("AIO 모듈의개수는 %d개입니다.", lCount);

AfxMessageBox(strData);
```

Visual Basic

```
' AIO 모듈의개수를확인합니다.
Dim lCount As Long
Dim  strData As String

AxaInfoGetModuleCount lCount
strData    = "AIO 모듈의개수는" + CStr(lCount) + "개입니다."

MsgBox strData
```

Delphi

```
{ AIO모듈의개수를확인합니다. }
var
  lCount : LongInt;
  strData : String;

begin
  AxaInfoGetModuleCount(@lCount);
  strData  :='AIO 모듈의개수는' + IntToStr(lCount) + '개입니다.';

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

C#

```
// AIO 모듈의개수를확인합니다.
int  nCount;
StringstrData;

CAXA.AxaInfoGetModuleCount(ref nCount);
strData.Format("AIO 모듈의개수는 %d개입니다.", lCount);

MessageBox.Show(strData);
```

See Also

AxaInfoIsAIOModule, AxaInfoGetModuleNo, AxaInfoGetModuleCount, AxaInfoGetInputCount, AxaInfoGetOutputCount, AxaInfoGetChannelNoDacOfModuleNo, AxaInfoGetChannelNoAdcOfModuleNo, AxaInfoGetModule, AxaInfoGetModuleStatus

AxaInfoGetChannelNoOfModuleNo

Purpose

지정한 모듈의 첫 번째 채널 번호를 확인합니다.

Format

C++

```
DWORD AxaInfoGetChannelNoOfModuleNo(long lModuleNo, long
*lpChannelNo);
```

Visual Basic

```
Function AxaInfoGetChannelNoOfModuleNo(ByVal lModuleNo As Long, ByRef
lpChannelNo As Long) As Long
```

Delphi

```
Function AxaInfoGetChannelNoOfModuleNo(lModuleNo : LongInt;
lpChannelNo : PLongInt) : DWord; stdcall;
```

C#

```
uint AxaInfoGetChannelNoOfModuleNo(int lModuleNo, ref int
lpChannelNo);
```

Parameters

[in/out] Name	[Init Value] Explanation
[in]lModuleNo	모듈 번호 (0 ~ 모듈 개수 -1)
[out]lpChannelNo	해당 모듈의 첫 번째 채널 번호

Return Values

[0000] AXT_RT_SUCCESS :함수실행성공
[1053] AXT_RT_NOT_OPEN :라이브러리초기화실패
[2101] AXT_RT_AIO_INVALID_MODULE_NO :유효하지않은 AIO 모듈번호
* See error code Table for more information on status error codes

Description

하나 이상의 제품 사용 시 여러 개의 모듈이 존재하게 되므로 제어하고자 하는 모듈의 첫 번째 채널 번호를 확인할 수 있다.

PCI-AI16HR 보드 두개 장착시 모듈 번호와 채널 번호 할당된 결과

Module No.	Module	Type	Channel
0	SIO-AI16HB	입력	0~15
1	SIO-AO2HB	출력	0~1
2	SIO-AI16HB	입력	16~31
3	SIO-AO2HB	출력	2~3

Example

C++

```
// 0번째모듈의첫번째채널번호를확인합니다.
long lChannelNo;
CString strData;

AxaInfoGetChannelNoOfModuleNo(0, &lChannelNo);
strData.Format("0번째모듈의첫번째채널번호는 %d 입니다.", lChannelNo);

AfxMessageBox(strData);
```

Visual Basic

```
' 0번째모듈의첫번째채널번호를확인합니다.
Dim lChannelNo As Long
Dim strData As String

AxaInfoGetChannelNoOfModuleNo 0, lChannelNo
strData = "0번째모듈의첫번째채널번호는" + CStr(lChannelNo) + "입니다."

MsgBox strData
```

Delphi

```
{ 0번째모듈의첫번째채널번호를확인합니다. }
var
  lChannelNo : LongInt;
  strData : String;

begin
  AxaInfoGetChannelNoOfModuleNo(0, @lChannelNo);
  strData  :='0번째모듈의첫번째채널번호는' + IntToStr(lChannelNo) + '입니다.';

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

C#

```
// 0번째모듈의첫번째채널번호를확인합니다.
int  nChannelNo;
String  strData;

CAXA.AxaInfoGetChannelNoOfModuleNo(0, ref nChannelNo);
strData.Format("0번째모듈의첫번째채널번호는 %d 입니다.", nChannelNo);

MessageBox.Show(strData);
```

See Also

AxaInfoIsAIOModule, AxaInfoGetModuleNo, AxaInfoGetModuleCount, AxaInfoGetInputCount, AxaInfoGetOutputCount, AxaInfoGetChannelNoDacOfModuleNo, AxaInfoGetChannelNoAdcOfModuleNo, AxaInfoGetModule, AxaInfoGetModuleStatus

AxaInfoGetChannelNoAdcOfModuleNo

Purpose

지정한 모듈의 첫 번째 입력 채널 번호를 확인합니다. (입력 모듈, 입력/출력 통합 모듈용)

Format

C++

```
DWORD AxaInfoGetChannelNoAdcOfModuleNo (long lModuleNo, long
*lpChannelNo);
```

Visual Basic

```
Function AxaInfoGetChannelNoAdcOfModuleNo (ByVal lModuleNo As Long,
ByRef lpChannelNo As Long) As Long
```

Delphi

```
Function AxaInfoGetChannelNoAdcOfModuleNo (lModuleNo : LongInt;
lpChannelNo : PLongInt) : DWord; stdcall;
```

C#

```
uint AxaInfoGetChannelNoAdcOfModuleNo (int lModuleNo, ref int
lpChannelNo);
```

Parameters

[in/out] Name	[Init Value] Explanation
[in]lModuleNo	모듈 번호 (0 ~ 모듈 개수 -1)
[out]lpChannelNo	해당 모듈의 첫 입력 번째 채널 번호

Return Values

[0000] AXT_RT_SUCCESS :함수실행성공
[1053] AXT_RT_NOT_OPEN :라이브러리초기화실패
[2101] AXT_RT_AIO_INVALID_MODULE_NO :유효하지않은 AIO 모듈번호
* See error code Table for more information on status error codes

Description

하나 이상의 제품 사용 시 여러 개의 모듈이 존재하게 되므로 제어하고자 하는 모듈의 첫 번째 입력 채널 번호를 확인할 수 있다.

PCI-AI16HR 보드 두개 장착시 모듈 번호와 채널 번호 할당된 결과

Module No.	Module	Type	Channel
0	SIO-AI16HB	입력	0~15
1	SIO-AO2HB	출력	0~1
2	SIO-AI16HB	입력	16~31
3	SIO-AO2HB	출력	2~3

Example

C++

```
// 0번째 모듈의 첫 번째 입력  채널 번호를 확인 합니다.
long lChannelNo;
CString strData;

AxaInfoGetChannelNoAdcOfModuleNo (0, &lChannelNo);
strData.Format("0번째 모듈의 첫 번째 입력 채널번호는 %d 입니다.", lChannelNo);

AfxMessageBox(strData);
```

Visual Basic

```
' 0번째 모듈의 첫 번째 입력 채널 번호를 확인 합니다.
Dim lChannelNo As Long
Dim strData As String

AxaInfoGetChannelNoAdcOfModuleNo 0, lChannelNo
strData = "0번째 모듈의 첫 번째 입력 채널번호는" + CStr(lChannelNo) + "입니다."

MsgBox strData
```

Delphi

```
{ 0번째 모듈의 첫 번째 입력 채널 번호를 확인 합니다. }
var
  lChannelNo : LongInt;
  strData : String;

begin
  AxaInfoGetChannelNoAdcOfModuleNo (0, @lChannelNo);
  strData  :='0번째 모듈의 첫 번째 입력 채널번호는' + IntToStr(lChannelNo) + '입니다.';

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

C#

```
// 0번째 모듈의 첫 번째 입력 채널 번호를 확인 합니다.
int  nChannelNo;
String  strData;

CAXA. AxaInfoGetChannelNoAdcOfModuleNo (0, ref nChannelNo);
strData.Format("0번째 모듈의 첫 번째 입력 채널번호는 %d 입니다.", nChannelNo);

MessageBox.Show(strData);
```

See Also

AxaInfoIsAIOModule, AxaInfoGetModuleNo, AxaInfoGetModuleCount, AxaInfoGetInputCount, AxaInfoGetOutputCount, AxaInfoGetChannelNoDacOfModuleNo, AxaInfoGetChannelNoAdcOfModuleNo, AxaInfoGetModule, AxaInfoGetModuleStatus

AxaInfoGetChannelNoDacOfModuleNo

Purpose

지정한 모듈의 첫 번째 출력 채널 번호를 확인합니다. (입력 모듈, 입력/출력 통합 모듈용)

Format

C++

```
DWORD AxaInfoGetChannelNoDacOfModuleNo (long lModuleNo, long
*lpChannelNo);
```

Visual Basic

```
Function AxaInfoGetChannelNoDacOfModuleNo (ByVal lModuleNo As Long,
ByRef lpChannelNo As Long) As Long
```

Delphi

```
Function AxaInfoGetChannelNoDacOfModuleNo (lModuleNo : LongInt;
lpChannelNo : PLongInt) : DWord; stdcall;
```

C#

```
uint AxaInfoGetChannelNoDacOfModuleNo (int lModuleNo, ref int
lpChannelNo);
```

Parameters

[in/out] Name	[Init Value] Explanation
[in]lModuleNo	모듈 번호 (0 ~ 모듈 개수 -1)
[out]lpChannelNo	해당 모듈의 첫 입력 번째 채널 번호

Return Values

[0000] AXT_RT_SUCCESS :함수실행성공
[1053] AXT_RT_NOT_OPEN :라이브러리초기화실패
[2101] AXT_RT_AIO_INVALID_MODULE_NO :유효하지않은 AIO 모듈번호
* See error code Table for more information on status error codes

Description

하나 이상의 제품 사용 시 여러 개의 모듈이 존재하게 되므로 제어하고자 하는 모듈의 첫 번째 출력 채널 번호를 확인할 수 있다.

PCI-AI16HR 보드 두개 장착시 모듈 번호와 채널 번호 할당된 결과

Module No.	Module	Type	Channel
0	SIO-AI16HB	입력	0~15
1	SIO-AO2HB	출력	0~1
2	SIO-AI16HB	입력	16~31
3	SIO-AO2HB	출력	2~3

Example

C++

```
// 0번째 모듈의 첫 번째 출력 채널 번호를 확인 합니다.
long lChannelNo;
CString strData;

AxaInfoGetChannelNoDacOfModuleNo (0, &lChannelNo);
strData.Format("0번째 모듈의 첫 번째 입력 채널번호는 %d 입니다.", lChannelNo);

AfxMessageBox(strData);
```

Visual Basic

```
' 0번째 모듈의 첫 번째 출력 채널 번호를 확인 합니다.
Dim lChannelNo As Long
Dim strData As String

AxaInfoGetChannelNoDacOfModuleNo 0, lChannelNo
strData = "0번째 모듈의 첫 번째 입력 채널번호는" + CStr(lChannelNo) + "입니다."

MsgBox strData
```

Delphi

```
{ 0번째 모듈의 첫 번째 출력 채널 번호를 확인 합니다. }
var
  lChannelNo : LongInt;
  strData : String;

begin
  AxaInfoGetChannelNoDacOfModuleNo (0, @lChannelNo);
  strData  :='0번째 모듈의 첫 번째 입력 채널번호는' + IntToStr(lChannelNo) + '입니다.';

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

C#

```
// 0번째 모듈의 첫 번째 출력 채널 번호를 확인 합니다.
int  nChannelNo;
String  strData;

CAXA. AxaInfoGetChannelNoDacOfModuleNo (0, ref nChannelNo);
strData.Format("0번째 모듈의 첫 번째 입력 채널번호는 %d 입니다.", nChannelNo);

MessageBox.Show(strData);
```

See Also

AxaInfoIsAIOModule, AxaInfoGetModuleNo, AxaInfoGetModuleCount, AxaInfoGetInputCount, AxaInfoGetOutputCount, AxaInfoGetChannelNoDacOfModuleNo, AxaInfoGetChannelNoAdcOfModuleNo, AxaInfoGetModule, AxaInfoGetModuleStatus

AxaInfoGetModuleStatus

Purpose

해당 모듈이 제어가 가능한 상태인지 반환 합니다.

Format

C++

```
DWORD AxaInfoGetModuleStatus (long lModuleNo);
```

Visual Basic

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

Delphi

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

C#

```
uint AxaInfoGetModuleStatus (int lModuleNo);
```

Parameters

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

Return Values

[0000] AXT_RT_SUCCESS :함수실행성공
[2051] AXT_RT_AIO_NOT_MODULE : AIO 모듈 없음
[2101] AXT_RT_AIO_INVALID_MODULE_NO :유효하지않은 AIO 모듈번호
* See error code Table for more information on status error codes

Description

해당 모듈이 제어가 가능한 상태인지 반환 한다. 반환 값은 Return Value 로 확인 할 수 있습니다.

Example

C++

```
// 해당 모듈이 제어가 가능한 상태인지 반환 합니다.
long lChannelNo;
Dword dwReturn;

dwReturn = AxaInfoGetModuleStatus (0);
strData.Format("%d", dwReturn);

AfxMessageBox(strData);
```

See Also

AxaInfoIsAIOModule, AxaInfoGetModuleNo, AxaInfoGetModuleCount, AxaInfoGetInputCount,

AxaInfoGetOutputCount, AxaInfoGetChannelNoDacOfModuleNo, AxaInfoGetChannelNoAdcOfModuleNo, AxaInfoGetModule, AxaInfoGetModuleStatus

AxaInfoGetModule

Purpose

지정한 모듈 번호로 베이스 보드 번호, 모듈 위치, 모듈 ID 를 확인합니다.

Format

C++

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

Visual Basic

```
Function AxaInfoGetModule(ByVal lModuleNo As Long, ByRef lpBoardNo As
Long, ByRef lpModulePos As Long, ByRef upModuleID As Long) As Long
```

Delphi

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

C#

```
uint AxaInfoGetModule(int lModuleNo, ref int lpBoardNo, ref int
lpModulePos, ref uint upModuleID);
```

Parameters

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

Return Values

[0000] AXT_RT_SUCCESS :함수실행성공
[1053] AXT_RT_NOT_OPEN :라이브러리초기화실패
[2101] AXT_RT_AIO_INVALID_MODULE_NO :유효하지않은 AIO 모듈번호
* See error code Table for more information on status error codes

Description

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

AIO모듈의 ID는 다음과 같습니다.

※참고사항

일체형 AIO 보드인 PCI-AI16HR 은 일체형 싱글 보드 타입이지만 입력 모듈 SIO-AI16HB 와 출력 모듈 SIO-AO2HB 가 혼합 조합되어 있는 것과 같습니다.

Module Define	Module ID	Module
AXT_SIO_AI4RB	0xA1 (161)	SIO-AI4RB Input 4ch

AXT_SIO_AO4RB	0xA2 (162)	SIO-AO4RB Output 4ch
AXT_SIO_AI16HB	0xA5 (165)	PCI-AI16HR 일체형보드의 Input 16ch 모듈
AXT_SIO_AO2HB	0xA6 (166)	PCI-AI16HR 일체형보드의 Output 2ch 모듈

Example

C++

```
// 0번째모듈의베이스보드번호, 모듈번호,모듈 ID를확인합니다.
long lBoardNo;
long lModulePos;
DWORD dwModuleID;
CString strData;

AxaInfoGetModule(0, &lBoardNo, &lModulePos, &dwModuleID);
strData.Format("BoardNo=%d, ModulePos=%d, dwModuleID=%02Xh", lBoardNo, lModulePos,
  dwModuleID);
AfxMessageBox(strData);
```

Visual Basic

```
' 0번째모듈의베이스보드번호, 모듈번호, 모듈 ID를확인합니다.
Dim lBoardNo As Long
Dim lModulePos As Long
Dim dwModuleID As DWord
Dim strData As String

AxaInfoGetModule 0, lBoardNo, lModulePos, dwModuleID
strData = "BoardNo=" + CStr(lBoardNo) + ", ModulePos=" + CStr(lModulePos) + ",
  ModuleID=" + CStr(dwModuleID)
MsgBox strData
```

Delphi

```
{ 0번째모듈의베이스보드번호, 모듈번호, 모듈 ID를확인합니다. }
var
  BoardNo : LongInt;
  lModulePos : LongInt;
  dwModuleID : DWord;
  strData : String;;

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

C#

```
// 0번째모듈의베이스보드번호, 모듈번호,모듈 ID를확인합니다.
intnBoardNo;
```

```
intnModulePos;
uintuModuleID;
String strData;

CAXA.AxaInfoGetModule(0, ref nBoardNo, ref nModulePos, ref uModuleID);
strData.Format("BoardNo=%d, ModulePos=%d, dwModuleID=%02Xh", nBoardNo, nModulePos,
  uModuleID);
MessageBox.Show(strData);
```

See Also

AxaInfoIsAIOModule, AxaInfoGetModuleNo, AxaInfoGetModuleCount, AxaInfoGetInputCount, AxaInfoGetOutputCount, AxaInfoGetChannelNoDacOfModuleNo, AxaInfoGetChannelNoAdcOfModuleNo, AxaInfoGetModule, AxaInfoGetModuleStatus

입력모듈정보검색

Function	Description
AxaiInfoGetModuleNoOfChannelNo	지정한 입력 채널 번호로 모듈 번호를 확인한다.
AxaiInfoGetChannelCount	아날로그 입력 모듈의 전체 채널 개수를 확인한다.

AxaiInfoGetModuleNoOfChannelNo

Purpose

지정한 아날로그 입력 채널 번호로 모듈 번호를 확인합니다.

Format

C++

```
DWORD AxaiInfoGetModuleNoOfChannelNo(long lAdcChannelNo, long
*lpModuleNo);
```

Visual Basic

```
Function AxaiInfoGetModuleNoOfChannelNo(ByVal lAdcChannelNo As Long,
ByRef lpModuleNo As Long) As Long
```

Delphi

```
functionAxaiInfoGetModuleNoOfChannelNo(lAdcChannelNo : LongInt;
lpModuleNo : PLongInt) : DWord; stdcall;
```

C#

```
uint AxaiInfoGetModuleNoOfChannelNo(int lChannelNo, ref int
lpModuleNo);
```

Parameters

[in/out] Name	[Init Value] Explanation
[in]lAdcChannelNo	입력 채널 번호 (0 ~ 채널 개수 -1)
[out]lpModuleNo	모듈 번호

Return Values

[0000] AXT_RT_SUCCESS :함수실행성공
[1053] AXT_RT_NOT_OPEN :라이브러리초기화실패
[2102]AXT_RT_AIO_INVALID_CHANNEL_NO :유효하지않은 AIO 채널번호
* See error code Table for more information on status error codes

Description

사용자가 지정한 아날로그 입력 채널 번호로 모듈 번호를 확인합니다. 모듈 번호를 파라메터로 갖는 함수들을 이용하여 각기 다른 설정을 함으로써 데이터 수집방식을 달리할 수 있습니다.

Example

C++

```
// 0번째입력채널의모듈번호를확인합니다.
long lModuleNo;
CString strData;

AxaiInfoGetModuleNoOfChannelNo(0, &lModuleNo);
strData.Format("0번째입력채널의모듈번호는 %d 입니다.", lModuleNo);
```

```
AfxMessageBox(strData);
```

Visual Basic

```
' 0번째입력채널의모듈번호를확인합니다.
Dim lModuleNo As Long
Dim strData As String

AxaiInfoGetModuleNoOfChannelNo 0, lModuleNo
strData = "0번째입력채널의모듈번호는" + CStr(lCount) + "입니다."

MsgBox strData
```

Delphi

```
{ 0번째입력채널의모듈번호를확인합니다. }
var
  lModuleNo : LongInt;
  strData : String;

begin
  AxaiInfoGetModuleNoOfChannelNo(0, @lModuleNo);
  strData  :='0번째입력채널의모듈번호는' + IntToStr(lModuleNo) + '입니다.';

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

C#

```
// 0번째입력채널의모듈번호를확인합니다.
int  nModuleNo;
String  strData;

CAXA.AxaiInfoGetModuleNoOfChannelNo(0, ref nModuleNo);
strData.Format("0번째입력채널의모듈번호는 %d 입니다.", nModuleNo);

MessageBox.Show(strData);
```

See Also

AxaiInfoGetChannelCount

AxaiInfoGetChannelCount

Purpose

시스템에 장착된 아날로그 입력 모듈의 전체 채널 개수를 확인합니다.

Format

C++

```
DWORD AxaiInfoGetChannelCount(long *lpAdcChannelCount);
```

Visual Basic

```
Function AxaiInfoGetChannelCount(ByRef lpAdcChannelCount As Long) As
Long
```

Delphi

```
functionAxaiInfoGetChannelCount(lpAdcChannelCount : PLongInt) : DWord;
stdcall;
```

Parameters

[in/out] Name	[Init Value] Explanation
[out]lpAdcChannelCount	입력 모듈의 전체 채널 개수

Return Values

[0000] AXT_RT_SUCCESS :함수실행성공
[1053] AXT_RT_NOT_OPEN :라이브러리초기화실패
* See error code Table for more information on status error codes

Description

시스템에 장착된 모든 아날로그 입력 모듈의 전체 채널 개수를 확인합니다.

Example

C++

```
// 아날로그입력모듈의전체채널개수를확인합니다.
long lCount;
CString strData;

AxaiInfoGetChannelCount(&lCount);
strData.Format("아날로그입력모듈의전체채널개수는 %d개입니다.", lCount);

AfxMessageBox(strData);
```

Visual Basic

```
'아날로그입력모듈의전체채널개수를확인합니다.
Dim lCount As Long
Dim strData As String

AxaiInfoGetChannelCount lCount
```

```
strData = "아날로그입력모듈의전체채널개수는" + CStr(lCount) + "개입니다."

MsgBox strData
```

Delphi

```
{ 아날로그입력모듈의전체채널개수를확인합니다. }
var
  lCount : LongInt;
  strData : String;

begin
  AxaiInfoGetChannelCount(@lCount);
  strData  :='아날로그입력모듈의전체채널개수는' + IntToStr(lCount) + '개입니다.';

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

C#

```
// 아날로그입력모듈의전체채널개수를확인합니다.
int  nCount;
String  strData;

CAXA.AxaiInfoGetChannelCount(ref nCount);
strData.Format("아날로그입력모듈의전체채널개수는 %d개입니다.", nCount);

MessageBox.Show(strData);
```

See Also

AxaiInfoGetModuleNoOfChannelNo

입력모듈인터럽터/채널이벤트설정및확인

Function	Description
AxaiEventSetChannel	지정한 모듈에 이벤트 메시지를 받아오기 위하여 윈도우 메시지, 콜백 함수 또는 이벤트 방식을 사용한다.
AxaiEventSetChannelEnable	지정한 입력 채널에 이벤트 사용 유무를 설정한다.
AxaiEventGetChannelEnable	지정한 입력 채널의 이벤트 사용 유무를 확인한다.
AxaiEventSetMultiChannelEnable	지정한 여러 입력 채널에 이벤트 사용 유무를 설정한다.
AxaiEventSetChannelMask	지정한 입력 채널에 이벤트 종류를 설정한다.
AxaiEventGetChannelMask	지정한 입력 채널에 이벤트 종류를 확인한다.
AxaiEventSetMultiChannelMask	지정한 여러 입력 채널에 이벤트 종류를 설정한다.
AxaiEventRead	이벤트 발생 위치를 확인한다.
AxaiInterruptSetModuleMask	지정한 모듈의 인터럽터 마스크를 설정한다.
AxaiInterruptGetModuleMask	지정한 모듈의 설정된 인터럽터 마스크를 확인한다.

AxaiEventSetChannel

Purpose

지정한 입력 채널에 수집된 데이터 처리를 위한 이벤트 수신 방법을 설정합니다.

Format

C++

```
DWORD AxaiEventSetChannel(long lAdcChannelNo, HWND hWnd, DWORD
uMesssage, AXT_EVENT_PROC pProc, HANDLE *pEvent);
```

Visual Basic

```
Function AxaiEventSetChannel(ByVal lAdcChannelNo As Long, ByVal hWnd
As Long, ByVal uMesssage As Long, ByVal pProc As Long, ByRef pEvent
As Long) As Long
```

Delphi

```
functionAxaiEventSetChannel(lAdcChannelNo : LongInt; hWnd : HWND;
uMesssage : DWord; pProc : AXT_EVENT_PROC; pEvent : PDWord) : DWord;
stdcall;
```

C#

```
uint AxaiEventSetChannel(int lModuleNo, IntPtr hWnd, uintuMessage,
CAXHS.AXT_INTERRUPT_PROC pProc, ref uint pEvent);
```

Parameters

[in/out] Name	[Init Value] Explanation
[in]lAdcChannelNo	입력 채널 번호 (0 ~ 입력 채널 -1)
[in]hWnd	윈도우 핸들 (사용하지 않으면 NULL)
[in]uMesssage	윈도우 메시지 (사용하지 않으면 NULL)
[in]pProc	콜백 함수 (사용하지 않으면 NULL)
[out]pEvent	이벤트 핸들 (사용하지 않으면 NULL)

정의된 콜백 함수 typedef void (stdcall *AXT_EVENT_PROC)(long lActiveNo, DWORD uFlag);

Return Values

[0000] AXT_RT_SUCCESS :함수실행성공
[1053] AXT_RT_NOT_OPEN :라이브러리초기화실패
[2102] AXT_RT_AIO_INVALID_CHANNEL_NO :유효하지않은 AIO 채널번호
* See error code Table for more information on status error codes

Description

H/W 타이머(Timer Trigger Mode, External Trigger Mode)를 이용한 연속적 데이터 수집 시에 사용합니다. AxaiSetTriggerMode 함수에서 TIMER_MODE 또는 EXTERNAL_MODE로 설정 후 연속적으로 획득되는 데이터를 처리하기 위한 방법을 설정합니다. 데이터 획득 이벤트는 채널 단위로 설정 가능하나 효율성을 위하여 모듈 내에 대표 채널을 지정하여 대표 채널에서 데이터를 처리할 때 나머지 채널도 함께 하는 것이 효율적입니다. 이벤트 처리 방법은 콜백, 윈도우 메시지, 이벤트 세 가지 중 한 가지로 선택

가능합니다. 사용할 방법을 선택 후 나머지 파라메터에는 NULL을 입력합니다. 세 가지 방법 중 콜백 방식은 이벤트 발생 시점에 즉시 콜백 함수가 호출됨으로 가장 빠르게 이벤트를 전달받을 수 있는 장점이 있으나 콜백 함수 내에 데이터를 처리하는 루틴 등의 시스템 부하가 걸리는 작업이 있을 때는 데이터 유실의 위험이 있습니다. 이때에는 데이터 처리 루틴 부분을 따로 동작할 수 있게 하는 윈도우 메시지 방식이나 이벤트 방식을 사용할 것을 권장합니다. 고속의 샘플링에서는 윈도우 메시지 방식도 윈도우 메시지 큐의 Dispatch루틴 또한 단일 스레드이므로 메시지를 적절히 빠른 시간에 처리하지 않을 때 메시지가 적체되어 Application 동작에 영향을 미치게 됩니다. 그러나 이벤트 방식은 이벤트의 발생 여부를 감시하는 특정 스레드를 사용하며 그 스레드 내에서 수집된 데이터를 처리하게 되므로 MultiProcessor 시스템등에서 자원을 가장 효율적으로 사용할 수 있으므로 특히 권장하는 방식입니다.
메시지 방식과 콜백 방식은 함수 첫 번째 인자 값이 이벤트가 발생한 채널이 되고 두 번째 인자 값이 발생한 이벤트의 플래그입니다. 이벤트 방식을 사용할 때 발생한 이벤트의 채널과 플래그를 알기 위해 AxaiEventRead를 사용합니다. 이벤트가 발생할 시 나타나는 이벤트 플래그는 다음과 같습니다.

이벤트 플래그

Enum	Value	Explanation
AIO_EVENT_DATA_UPPER	1	버퍼에 데이터가 상한 설정 값보다 많아질 때 이벤트 발생
AIO_EVENT_DATA_LOWER	2	버퍼에 데이터가 하한 설정 값보다 적어질 때 이벤트 발생
AIO_EVENT_DATA_FULL	3	버퍼에 데이터가 꽉 찼을 때 이벤트 발생
AIO_EVENT_DATA_EMPTY	4	버퍼에 데이터가 없을 때 이벤트 발생

Example

C++

```
// 0번입력채널에이벤트메시지를받아오기위하여윈도우메시지사용
BEGIN_MESSAGE_MAP(CDIODlg, CDialog)
   //{{AFX_MSG_MAP
   // 선언
   ON_MESSAGE(WM_AXL_EVENT, OnAdcDataEvent)
//}}AFX_MSG_MAP
END_MESSAGE_MAP()

// AXL의인터럽트기능을사용할수있도록설정한다
AxlInterruptEnable();

// 이벤트발생시'OnAdcDataEvent()'호출됨
AxaiEventSetChannel(0, m_hWnd, WM_AXL_EVENT, NULL, NULL);

// 이벤트방식으로이용할때
HANDLE g_hEvent;
AxaiEventSetChannel(0, NULL, NULL, NULL, &g_hEvent);
// 라이브러리내부에서핸들생성
```

Visual Basic

```
'0번입력채널에이벤트메시지를받아오기위하여윈도우메시지사용
'선언
Private Sub AxtMsg1_OnMessage1(ByVal wParam As Long, ByVal lParam As Long)
'이벤트메시지처리구문
End Sub
```

```
'이벤트발생시'AxtMsg1_OnMessage1()'호출됨
AxtMsg1.Message1 = WM_AXL_EVENT

// AXL의인터럽트기능을사용할수있도록설정한다
AxlInterruptEnable
AxaiEventSetChannel 0, Me.hWnd, WM_AXL_EVENT, 0, 0
```

Delphi

```
{ 0번입력채널에이벤트메시지를받아오기위하여윈도우메시지사용 }
procedure OnEventMessage(var Msg : TMessage); message WM_AXL_EVENT;

{ 선언 }
procedure TForm1.OnEventMessage(var Msg : TMessage);
begin
{ 이벤트메시지처리구문 }
end;

{ 이벤트발생시'OnEventMessage()'호출됨 }

// AXL의인터럽트기능을사용할수있도록설정한다
AxlInterruptEnable();
AxaiEventSetChannel(0, Form1.Handle, WM_AXL_EVENT, nil, nil);
```

C#

```
// 인터럽트메시지를 받아오기 위하여 윈도우 메시지 사용
protectedoverridevoidWndProc(refMessagem)
{
    switch(m.Msg)
    {
        case (int)AXT_EVENT.WM_AXL_INTERRUPT:
            InterruptProc(AXT_INTERRUPT_CLASS.KIND_MESSAGE, (int)m.WParam,
            (uint)m.LParam);
            break;
    }
    base.WndProc( refm );
}

// AXL의인터럽트기능을사용할수있도록설정한다
CAXL.AxlInterruptEnable();

// 이벤트발생시'InterruptProc()'호출됨
AxaiEventSetChannel(0, (uint)this.handle, (uint)AXT_EVENT.WM_AXL_INTERRUPT, null,
null);
```

```
// 이벤트방식으로이용할때
HANDLE g_hEvent;
AxaiEventSetChannel(0, NULL, NULL, NULL, &g_hEvent);
```

See Also

AxaiEventSetChannelEnable, AxaiEventGetChannelEnable, AxaiEventSetMultiChannelEnable, AxaiEventSetChannelMask, AxaiEventGetChannelMask,AxaiEventSetMultiChannelMask, AxaiEventRead, AxaiInterruptSetModuleMask, AxaiInterruptGetModuleMask

AxaiEventSetChannelEnable

Purpose

지정한 입력 채널에 수집된 데이터 처리를 위한 이벤트 사용 여부를 설정합니다.

Format

C++

```
DWORD AxaiEventSetChannelEnable(long lAdcChannelNo, DWORD uUse);
```

Visual Basic

```
Function AxaiEventSetChannelEnable(ByVal lAdcChannelNo As Long, ByVal
uUse As Long) As Long
```

Delphi

```
functionAxaiEventSetChannelEnable(lAdcChannelNo : LongInt; uUse :
DWord) : DWord; stdcall;
```

C#

```
uint AxaiEventSetChannelEnable(int lChannelNo, uint uUse);
```

Parameters

[in/out] Name	[Init Value] Explanation
[in]lAdcChannelNo	입력 채널 번호 (0 ~ 입력 채널 개수 -1)
[in]uUse	[00h]이벤트 사용여부 설정: AXT_USE - [00h] 이벤트 사용 해제 - [01h] 이벤트 사용 설정

Return Values

[0000] AXT_RT_SUCCESS :함수실행성공
[1053] AXT_RT_NOT_OPEN :라이브러리초기화실패
[2102] AXT_RT_AIO_INVALID_CHANNEL_NO :유효하지않은 AIO 채널번호
* See error code Table for more information on status error codes

Description

H/W 타이머(Timer Trigger Mode, External Trigger Mode)를 이용한 연속적 데이터 수집 시에 데이터 처리를 위한 방법으로 이벤트 사용 여부를 설정합니다.

Example

C++

```
// 0번입력채널에이벤트를사용가능하게설정합니다.
AxaiEventSetChannelEnable(0, ENABLE);
```

Visual Basic

```
' 0번째입력채널에이벤트를사용가능하게설정합니다.
AxaiEventSetChannelEnable 0, ENABLE
```

Delphi

```
{ 0번째입력채널에이벤트를사용가능하게설정합니다. }
AxaiEventSetChannelEnable(0, ENABLE);
```

C#

```
// 0번입력채널에이벤트를사용가능하게설정합니다.
CAXA.AxaiEventSetChannelEnable(0, (uint)AXT_USE.ENABLE);
```

See Also

AxaiEventSetChannel, AxaiEventGetChannelEnable, AxaiEventSetMultiChannelEnable, AxaiEventSetChannelMask, AxaiEventGetChannelMask,AxaiEventSetMultiChannelMask, AxaiEventRead, AxaiInterruptSetModuleMask, AxaiInterruptGetModuleMask

AxaiEventGetChannelEnable

Purpose

지정한 입력 채널의 수집된 데이터 처리를 위한 이벤트 사용 여부를 확인합니다.

Format

C++

```
DWORD AxaiEventGetChannelEnable(long lAdcChannelNo, DWORD *upUse);
```

Visual Basic

```
Function AxaiEventGetChannelEnable(ByVal lAdcChannelNo As Long, ByRef
upUse As Long) As Long
```

Delphi

```
functionAxaiEventGetChannelEnable(lAdcChannelNo : LongInt; upUse :
PDWord) : DWord; stdcall;
```

C#

```
uint AxaiEventGetChannelEnable(int lChannelNo, ref uint upUse);
```

Parameters

[in/out] Name	[Init Value] Explanation
[in]lAdcChannelNo	입력 채널 번호 (0 ~ 입력 채널 개수 -1)
[out]upUse	[00h]이벤트 사용 여부 설정: AXT_USE - [00h] 이벤트 사용 해제 - [01h] 이벤트 사용 설정

Return Values

[0000] AXT_RT_SUCCESS :함수실행성공
[1053] AXT_RT_NOT_OPEN :라이브러리초기화실패
[2102] AXT_RT_AIO_INVALID_CHANNEL_NO :유효하지않은 AIO 채널번호
* See error code Table for more information on status error codes

Description

H/W 타이머(Timer Trigger Mode, External Trigger Mode)를 이용한 연속적 데이터 수집 시에 데이터 처리를 위한 방법으로 설정된 이벤트 사용 여부를 확인합니다.

Example

C++

```
// 0번입력채널의이벤트사용가능여부를확인합니다.
DWORD dwUse;
AxaiEventGetChannelEnable(0, &dwUse);
if (dwUse)
{
  AfxMessageBox("0번채널은이벤트사용가능합니다.");
}
```

```
else
{
  AfxMessageBox("0번채널은이벤트사용불가능합니다.");
}
```

Visual Basic

```
' 0번입력채널의이벤트사용가능여부를확인합니다.
Dim lUse As Long
AxaiEventGetChannelEnable 0, lUse

If lUse = ENABLE Then
  MsgBox "0번채널은이벤트사용가능합니다."
Else
  MsgBox "0번채널은이벤트사용불가능합니다."
End If
```

Delphi

```
{ 0번입력채널의이벤트사용가능여부를확인합니다. }
var
  dwUse : DWord;

begin
  AxaiEventGetChannelEnable(0, @dwUse);
  if (dwUse = ENABLE) then
     Application.MessageBox ('0번채널은이벤트사용가능합니다.', 'Ajinextek', MB_OK);
  else
     Application.MessageBox ('0번채널은이벤트사용불가능합니다.', 'Ajinextek', MB_OK);
end;
```

C#

```
// 0번입력채널의이벤트사용가능여부를확인합니다.
uint uUse;
CAXA.AxaiEventGetChannelEnable(0, ref uUse);
if (uUse)
{
  MessageBox.Show("0번채널은이벤트사용가능합니다.");
}
else
{
  MessageBox.Show("0번채널은이벤트사용불가능합니다.");
}
```

See Also

AxaiEventSetChannel, AxaiEventSetChannelEnable, AxaiEventSetMultiChannelEnable, AxaiEventSetChannelMask, AxaiEventGetChannelMask,AxaiEventSetMultiChannelMask, AxaiEventRead, AxaiInterruptSetModuleMask, AxaiInterruptGetModuleMask

AxaiEventSetMultiChannelEnable

Purpose

지정한 여러 입력 채널에 수집된 데이터 처리를 위한 이벤트 사용 여부를 설정합니다.

Format

C++

```
DWORD AxaiEventSetMultiChannelEnable(long lSize, long *lpAdcChannelNo,
DWORD uUse);
```

Visual Basic

```
Function AxaiEventSetMultiChannelEnable(ByVal lSize As Long, ByRef
lpAdcChannelNo As Long, ByVal uUse As Long) As Long
```

Delphi

```
functionAxaiEventSetMultiChannelEnable(lSize : LongInt;
lpAdcChannelNo : PLongInt; uUse : DWord) : DWord; stdcall;
```

C#

```
uint AxaiEventSetMultiChannelEnable(int lSize, ref int lpChannelNo,
uint uUse);
```

Parameters

[in/out] Name	[Init Value] Explanation
[in]lSize	사용할 입력 채널의 개수
[in]lpAdcChannelNo	사용할 채널 번호의 배열
[in]uUse	[00h]이벤트 사용여부 설정: AXT_USE - [00h] 이벤트 사용 해제 - [01h] 이벤트 사용 설정

Return Values

[0000] AXT_RT_SUCCESS :함수실행성공
[1053] AXT_RT_NOT_OPEN :라이브러리초기화실패
[2102] AXT_RT_AIO_INVALID_CHANNEL_NO :유효하지않은 AIO 채널번호
* See error code Table for more information on status error codes

Description

H/W 타이머(Timer Trigger Mode, External Trigger Mode)를 이용한 연속적 데이터 수집 시에 데이터 처리를 위한 방법으로 여러 입력 채널에 이벤트 사용 여부를 설정합니다.

Example

C++

```
// 여러입력채널에이벤트를사용가능하게설정합니다.
long lAdcChannelNo[4] = {0, 1, 2, 3};

AxaiEventSetMultiChannelEnable(4, lAdcChannelNo, ENABLE);
```

Visual Basic

```
'여러입력채널에이벤트를사용가능하게설정합니다.
Dim lAdcChannelNo(0 To 3) As Long

lAdcChannelNo(0) = 0
lAdcChannelNo(1) = 1
lAdcChannelNo(2) = 2
lAdcChannelNo(3) = 3

AxaiEventSetMultiChannelEnable 4, lAdcChannelNo(0), ENABLE
```

Delphi

```
{ 여러입력채널에이벤트를사용가능하게설정합니다. }
var
  lAdcChannelNo: array [0..3] of LongInt;

begin
  lAdcChannelNo[0]:= 0;
  lAdcChannelNo[1]:= 1;
  lAdcChannelNo[2]:= 2;
  lAdcChannelNo[3]:= 3;

  AxaiEventSetMultiChannelEnable(4, @lAdcChannelNo, ENABLE);
end;
```

C++

```
// 여러입력채널에이벤트를사용가능하게설정합니다.
int[] nAdcChannelNo = new int[4];

nAdcChannelNo[0] = 0;
nAdcChannelNo[1] = 1;
nAdcChannelNo[2] = 2;
nAdcChannelNo[3] = 3;

CAXA.AxaiEventSetMultiChannelEnable(4, ref nAdcChannelNo[0], AXT_USE.ENABLE);
```

See Also

AxaiEventSetChannel, AxaiEventSetChannelEnable, AxaiEventGetChannelEnable, AxaiEventSetChannelMask, AxaiEventGetChannelMask,AxaiEventSetMultiChannelMask, AxaiEventRead, AxaiInterruptSetModuleMask, AxaiInterruptGetModuleMask

AxaiEventSetChannelMask

Purpose

지정한 입력 채널에 수집된 데이터 처리를 위한 이벤트 종류를 설정합니다.

Format

C++

```
DWORD AxaiEventSetChannelMask(long lAdcChannelNo, DWORD uMask);
```

Visual Basic

```
Function AxaiEventSetChannelMask(ByVal lAdcChannelNo As Long, ByVal
uMask As Long) As Long
```

Delphi

```
functionAxaiEventSetChannelMask(lAdcChannelNo : LongInt; uMask :
DWord) : DWord; stdcall;
```

C#

```
uint AxaiEventSetChannelMask(int lChannelNo, uint uMask);
```

Parameters

[in/out] Name	[Init Value] Explanation
[in]lAdcChannelNo	입력 채널 번호
[in]uMask	[00h]이벤트 종류

Return Values

[0000] AXT_RT_SUCCESS :함수실행성공
[1053] AXT_RT_NOT_OPEN :라이브러리초기화실패
[2102] AXT_RT_AIO_INVALID_CHANNEL_NO :유효하지않은 AIO 채널번호
* See error code Table for more information on status error codes

Description

H/W 타이머(Timer Trigger Mode, External Trigger Mode)를 이용한 연속적 데이터 수집 시에 데이터 처리를 위한 방법으로 지정 입력 채널에 이벤트 종류를 설정합니다. 설정한 이벤트 발생 시 AxaiEventSetChannel 함수에서 설정된 이벤트 처리 방법으로 서비스 루틴 함수가 호출됩니다.
이벤트 종류는 아래의 Table 의 Bit 설정 값을 참조합니다.

이벤트 종류

Enum	Bit	Explanation
DATA_EMPTY	0x01	버퍼에 데이터가 없을 때 이벤트 발생
DATA_MANY	0x02	버퍼에 데이터가 상한 설정 값보다 많아질 때 이벤트 발생
DATA_SMALL	0x04	버퍼에 데이터가 하한 설정 값보다 적어질 때 이벤트 발생
DATA_FULL	0x08	버퍼에 데이터가 꽉 찼을 때 이벤트 발생

Example

C++

```
// 0번입력채널에이벤트종류를상한설정값보다많아질때와하한설정값보다적어질때발생하도록
// 설정합니다.
AxaiEventSetChannelMask (0, DATA_MANY | DATA_SMALL);
```

Visual Basic

```
' 0번입력채널에이벤트종류를상한설정값보다많아질때와하한설정값보다적어질때발생하도록
'설정합니다.
AxaiEventSetChannelMask 0, 6
```

Delphi

```
{ 0번입력채널에이벤트종류를상한설정값보다많아질때와하한설정값보다적어질때발생하도록설정합니다. }
AxaiEventSetChannelMask (0, 6);
```

C#

```
// 0번입력채널에이벤트종류를상한설정값보다많아질때와하한설정값보다적어질때발생하도록
// 설정합니다.
CAXA.AxaiEventSetChannelMask(0, (uint)AXT_AIO_INTERRUPT_MASK.DATA_MANY |
AXT_AIO_INTERRUPT_MASK.DATA_SMALL);
```

See Also

AxaiEventSetChannel, AxaiEventSetChannelEnable, AxaiEventGetChannelEnable, AxaiEventSetMultiChannelEnable, AxaiEventGetChannelMask,AxaiEventSetMultiChannelMask, AxaiEventRead, AxaiInterruptSetModuleMask, AxaiInterruptGetModuleMask

AxaiEventGetChannelMask

Purpose

지정한 입력 채널에 수집된 데이터 처리를 위해 설정한 이벤트 종류를 확인합니다.

Format

C++

```
DWORD AxaiEventGetChannelMask(long lAdcChannelNo, DWORD *upMask);
```

Visual Basic

```
Function AxaiEventGetChannelMask(ByVal lAdcChannelNo As Long, ByRef
upMask As Long) As Long
```

Delphi

```
functionAxaiEventGetChannelMask(lAdcChannelNo : LongInt; upMask :
PDWord) : DWord; stdcall;
```

C#

```
uint AxaiEventGetChannelMask(int lChannelNo, ref uint upMask);
```

Parameters

[in/out] Name	[Init Value] Explanation
[in]lAdcChannelNo	입력 채널 번호
[out]upMask	[00h]이벤트 종류

Return Values

[0000] AXT_RT_SUCCESS :함수실행성공
[1053] AXT_RT_NOT_OPEN :라이브러리초기화실패
[2102] AXT_RT_AIO_INVALID_CHANNEL_NO :유효하지않은 AIO 채널번호
* See error code Table for more information on status error codes

Description

H/W 타이머(Timer Trigger Mode, External Trigger Mode)를 이용한 연속적 데이터 수집 시에 데이터 처리를 위한 방법으로 지정 입력 채널에 설정한 이벤트 종류를 확인합니다. 설정한 이벤트 발생 시 AxaiEventSetChannel 함수에서 설정된 이벤트 처리 방법으로 서비스 루틴 함수가 호출됩니다. 이벤트 종류는 아래의 Table 의 Bit 설정 값을 참조합니다.

이벤트 종류

Enum	Bit	Explanation
DATA_EMPTY	0x01	버퍼에 데이터가 없을 때 이벤트 발생
DATA_MANY	0x02	버퍼에 데이터가 상한 설정 값보다 많아질 때 이벤트 발생
DATA_SMALL	0x04	버퍼에 데이터가 하한 설정 값보다 적어질 때 이벤트 발생
DATA_FULL	0x08	버퍼에 데이터가 꽉 찼을 때 이벤트 발생

Example

C++

```
// 0번입력채널에이벤트종류를확인합니다.
DWORD dwMask;

AxaiEventGetChannelMask(0, &dwMask);
```

Visual Basic

```
' 0번입력채널에이벤트종류를확인합니다.
Dim lMask As Long

AxaiEventGetChannelMask 0, dwMask
```

Delphi

```
{ 0번입력채널에이벤트종류를확인합니다. }
var
  dwMask: DWord;

begin
  AxaiEventGetChannelMask(0, @dwMask);
end;
```

C#

```
// 0번입력채널에이벤트종류를확인합니다.
uintuMask;

CAXA.AxaiEventGetChannelMask(0, ref uMask);
```

See Also

AxaiEventSetChannel, AxaiEventSetChannelEnable, AxaiEventGetChannelEnable, AxaiEventSetMultiChannelEnable, AxaiEventSetChannelMask, AxaiEventSetMultiChannelMask, AxaiEventRead, AxaiInterruptSetModuleMask, AxaiInterruptGetModuleMask

AxaiEventSetMultiChannelMask

Purpose

지정한 여러 입력 채널에 이벤트 종류를 설정한다.

Format

C++

```
DWORD AxaiEventSetMultiChannelMask(long lSize, long *lpAdcChannelNo,
DWORD uMask);
```

Visual Basic

```
Function AxaiEventSetMultiChannelMask(ByVal lSize As Long, ByRef
lpAdcChannelNo As Long, ByVal uMask As Long) As Long
```

Delphi

```
functionAxaiEventSetMultiChannelMask(lSize : LongInt; lpAdcChannelNo :
PLongInt; uMask : DWord) : DWord; stdcall;
```

Parameters

[in/out] Name	[Init Value] Explanation
[>>]lSize	사용할 입력 채널의 개수
[>>]lpAdcChannelNo	사용할 채널 번호의 배열
[>>]uMask	- [00h] 이벤트 사용 해제 - [01h] 이벤트 사용 설정

Return Values

[0000] AXT_RT_SUCCESS : AXL 라이브러리초기화성공
[1053] AXT_RT_NOT_OPEN :라이브러리초기화실패
[2102] AXT_RT_AIO_INVALID_CHANNEL_NO :유효하지않은 AIO 채널번호
* See error code Table for more information on status error codes

Description

사용자가 지정한 여러 입력 채널에 이벤트 종류를 설정한다.

여기서 설정한 이벤트만 걸리게 된다. 이벤트의 종류로는 버퍼에 데이터가 없을 때, 버퍼에 데이터가 상한 설정 값보다 많아질 때, 버퍼에 데이터가 하한 설정 값보다 적어질 때, 버퍼에 데이터가 꽉 찼을 때가 있다.

lSize 는 사용할 채널의 개수를 넣어주면 된다.

*lpAdcChannelNo 는 채널의 정보를 담을 배열로서 사용할 채널의 번호를 배열의 값으로 넣어주면 된다. 예를 들어 long lpAdcChannelNo[4] = {0, 2, 3, 4}; 처럼 하면 0, 2, 3, 4 번 채널을 설정한다는 것이다.

이벤트 종류

Enum	Bit	Explanation

DATA_EMPTY	0x01	버퍼에 데이터가 없을 때 이벤트 발생
DATA_MANY	0x02	버퍼에 데이터가 상한 설정 값보다 많아질 때 이벤트 발생
DATA_SMALL	0x04	버퍼에 데이터가 하한 설정 값보다 적어질 때 이벤트 발생
DATA_FULL	0x08	버퍼에 데이터가 꽉 찼을 때 이벤트 발생

Example

C++

```
// 여러입력채널에이벤트종류를상한설정값보다많아질때와하한설정값보다적어질때발생하도록
// 설정한다.
long lAdcChannelNo[4] = {0, 1, 2, 3};

AxaiEventSetMultiChannelMask(4, lAdcChannelNo, DATA_MANY | DATA_SMALL);
```

Visual Basic

```
'여러입력채널에이벤트상한설정값보다많아질때와하한설정값보다적어질때발생하도록설정한다.
Dim lAdcChannelNo(0 To 3) As Long

lAdcChannelNo(0) = 0
lAdcChannelNo(1) = 1
lAdcChannelNo(2) = 2
lAdcChannelNo(3) = 3

AxaiEventSetMultiChannelMask 4, lAdcChannelNo(0), 6
```

Delphi

```
{ 여러입력채널에이벤트종류를상한설정값보다많아질때와하한설정값보다적어질때발생하도록설정한다. }
var
  lAdcChannelNo: array [0..3] of LongInt;

begin
  lAdcChannelNo[0]:= 0;
  lAdcChannelNo[1]:= 1;
  lAdcChannelNo[2]:= 2;
  lAdcChannelNo[3]:= 3;

  AxaiEventSetMultiChannelMask(4, @lAdcChannelNo, 6);
end;
```

See Also

AxaiEventSetChannel, AxaiEventSetChannelEnable, AxaiEventGetChannelEnable, AxaiEventSetMultiChannelEnable, AxaiEventSetChannelMask, AxaiEventGetChannelMask, AxaiEventRead, AxaiInterruptSetModuleMask, AxaiInterruptGetModuleMask

AxaiEventRead

Purpose

수집된 데이터 처리를 위한 이벤트 발생 시 이벤트 발생 입력 채널 번호를 확인합니다.

Format

C++

```
DWORD AxaiEventRead(long *lpAdcChannelNo, DWORD *upMode);
```

Visual Basic

```
Function AxaiEventRead(ByRef lpAdcChannelNo As Long, ByRef upMode As
Long) As Long
```

Delphi

```
functionAxaiEventRead(lpAdcChannelNo : PLongInt; upMode : PDWord) :
DWord; stdcall;
```

C#

```
uint AxaiEventRead(ref int lpChannelNo, ref uint upMode);
```

Parameters

[in/out] Name	[Init Value] Explanation
[out]lpAdcChannelNo	입력 채널 번호
[out]upMode	이벤트 플래그

Return Values

[0000] AXT_RT_SUCCESS :함수실행성공
[1053] AXT_RT_NOT_OPEN :라이브러리초기화실패
[2052]AXT_RT_AIO_NOT_EVENT :AIO 이벤트읽지못함
* See error code Table for more information on status error codes

Description

H/W 타이머(Timer Trigger Mode, External Trigger Mode)를 이용한 연속적 데이터 수집 시에 데이터 처리를 위한 방법으로 지정 입력 채널에이벤트 설정 후 이벤트가 발생하였을 때 입력 채널 번호와 이벤트 플래그를 확인합니다. 이벤트 플래그는 아래의 Table 값을 참조합니다.

이벤트 종류

Enum	Value	Explanation
AIO_EVENT_DATA_UPPER	1	버퍼에 데이터가 상한 설정 값보다 많아질 때 이벤트 발생
AIO_EVENT_DATA_LOWER	2	버퍼에 데이터가 하한 설정 값보다 적어질 때 이벤트 발생
AIO_EVENT_DATA_FULL	3	버퍼에 데이터가 꽉 찼을 때 이벤트 발생
AIO_EVENT_DATA_EMPTY	4	버퍼에 데이터가 없을 때 이벤트 발생

Example

C++

```
// 0번째채널의이벤트발생위치를확인합니다.
longlAdcChannelNo;
DWORD dwMode;

AxaiEventRead(&lAdcChannelNo, &dwMode);
```

Visual Basic

```
' 0번째채널의이벤트발생위치를확인합니다.
Dim lAdcChannelNo As long
Dim lMode As long

AxaiEventReadlAdcChannelNo, lMode
```

Delphi

```
{ 0번째채널의이벤트발생위치를확인합니다. }
var
  lAdcChannelNo : LongInt;
  dwMode : DWord;

begin
  AxaiEventRead(@lAdcChannelNo, @dwMode);
end;
```

C#

```
// 0번째채널의이벤트발생위치를확인합니다.
intnAdcChannelNo;
uintuMode;

CAXA.AxaiEventRead(ref nAdcChannelNo, ref uMode);
```

See Also

AxaiEventSetChannel, AxaiEventSetChannelEnable, AxaiEventGetChannelEnable, AxaiEventSetMultiChannelEnable, AxaiEventSetChannelMask, AxaiEventGetChannelMask, AxaiEventSetMultiChannelMask, AxaiInterruptSetModuleMask, AxaiInterruptGetModuleMask

AxaiInterruptSetModuleMask

Purpose

지정한 모듈에인터럽트발생 소스를 설정합니다.

Format

C++

```
DWORD AxaiInterruptSetModuleMask(long lModuleNo, DWORD uMask);
```

Visual Basic

```
Function AxaiInterruptSetModuleMask(ByVal lModuleNo As Long, ByVal
uMask As Long) As Long
```

Delphi

```
functionAxaiInterruptSetModuleMask(lModuleNo : LongInt; uMask :
DWord) : DWord; stdcall;
```

C#

```
uint AxaiInterruptSetModuleMask(int lModuleNo, uint uMask);
```

Parameters

[in/out] Name	[Init Value] Explanation
[in]lModuleNo	입력 모듈 번호
[in]uMask	[02h]지정할 인터럽터 마스크: AXT_AIO_INTERRUPT_MASK

Return Values

[0000] AXT_RT_SUCCESS :함수실행성공
[1053] AXT_RT_NOT_OPEN :라이브러리초기화실패
[2101]AXT_RT_AIO_INVALID_MODULE_NO :유효하지않은 AIO 모듈
* See error code Table for more information on status error codes

Description

이 함수는 H/W 타이머(Timer Trigger Mode, External Trigger Mode)를 이용한 연속적 데이터 수집 시에 FIFO에서 사용자 지정 버퍼로 데이터 처리를 위한 방법으로 인터럽트를 이용하기 위해 사용됩니다. 인터럽트 발생 기본 설정은 FIFO_HALF_FULL 이며 인터럽트 소스의 종류는 다음과 같습니다.

※참고 :SIO-AI4RB 는 지원하지 않습니다.

Enum	Explanation
SCAN_END(1)	설정된 채널 모두 A/D 변환을 할 때 마다 인터럽트가 발생하며 메모리 버퍼로 데이터 복사가 이루어 집니다. (저속 샘플링시 – 트리거 신호가 100~200 Hz 이하의 저속일 경우 사용 권장)
FIFO_HALF_FULL(2)	H/W FIFO가 반 이상 찼을 경우 인터럽트 발생하며 메모리 버퍼로 데이터 복사가 이루어 집니다.(고속 샘플링시 권장)

만약, 고속 샘플링을 하며인터럽트 소스를 SCAN_END 설정하면 한번의 AD 변환을 할 때마다 인터럽트가 발생하게 되므로 오버헤드가 심해집니다. 반대로 저속 샘플링에서 FIFO_HALF_FULL 로

마스크를 지정하게 되면 FIFO 의 반 까지 데이터가 차야만 메모리 버퍼로 데이터 복사가 이루어지기 때문에 데이터 수집 지연이 심해집니다.

예) 샘플링 주파수 : 10Hz (100msec)

인터럽트 소스 :FIFO_HALF_FULL

PCI-AI16HR FIFO SIZE : 1024

위와 같은 설정에서 인터럽트 소스는 FIFO_HALF_FULL 이므로 512 개의 데이터가 수집되어야만 인터럽트가 발생하게 됩니다. 이때, 소요시간은 샘플링 시간이 0.1 초이므로 512 * 0.1 = 51.2 초의 시간이 소요됩니다. 반면, 인터럽트 소스를 SCAN_END 로 설정할 경우 한번의 샘플링이 끝나는 0.1 초만에 메모리 버퍼에서 수집된 첫번째 데이터를 얻어 올 수 있습니다. 이와 같이 Application 의 특성과 샘플링 주파수를 고려 후 설정하여야 효율적으로 사용할 수 있습니다.

Example

C++

```
// 0번째모듈의하드웨어인터럽터마스크를결정합니다.

AxaiSetTriggerMode(0, TIMER_MODE);
AxaiHwSetSampleFreq(0, 10); // 10Hz 저속샘플링
AxaiInterruptSetModuleMask(0, SCAN_END);
// External 또는 Timer Trigger Mode로연속적신호감시에서저속샘플링일경우
// SCAN_END사용
```

Visual Basic

```
'0번째모듈의하드웨어인터럽터마스크를결정합니다.

AxaiInterruptSetModuleMask0, FIFO_HALF_FULL
```

Delphi

```
{ 0번째모듈의하드웨어인터럽터마스크를결정합니다. }
begin
  AxaiInterruptSetModuleMask(0, FIFO_HALF_FULL);
end;
```

C#

```
// 0번째모듈의하드웨어인터럽터마스크를결정합니다.

CAXA.AxaiSetTriggerMode(0, (uint)AXT_TRIGGER_MODE.TIMER_MODE);
CAXA.AxaiHwSetSampleFreq(0, 10); // 10Hz 저속샘플링
CAXA.AxaiInterruptSetModuleMask(0, SCAN_END);
// External 또는 Timer Trigger Mode로연속적신호감시에서저속샘플링일경우
// SCAN_END사용
```

See Also

AxaiEventSetChannel, AxaiEventSetChannelEnable, AxaiEventGetChannelEnable, AxaiEventSetMultiChannelEnable, AxaiEventSetChannelMask, AxaiEventGetChannelMask, AxaiEventSetMultiChannelMask, AxaiEventRead, AxaiInterruptGetModuleMask

AxaiInterruptGetModuleMask

Purpose

지정한 모듈에 설정한 인터럽트소스를 확인합니다.

Format

C++

```
DWORD AxaiInterruptGetModuleMask(long lModuleNo, DWORD *upMask);
```

Visual Basic

```
Function AxaiInterruptGetModuleMask(ByVal lModuleNo As Long, ByRef
upMask As Long) As Long
```

Delphi

```
functionAxaiInterruptGetModuleMask(lModuleNo : LongInt; upMask :
PDWord) : DWord; stdcall;
```

C#

```
uint AxaiInterruptGetModuleMask(int lModuleNo, ref uint upMask);
```

Parameters

[in/out] Name	[Init Value] Explanation
[in]lModuleNo	입력 모듈 번호
[out]upMask	[02h]지정할 인터럽트 마스크: AXT_AIO_OUTTERRUPT_MASK

Return Values

[0000] AXT_RT_SUCCESS :함수실행성공
[1053] AXT_RT_NOT_OPEN :라이브러리초기화실패
[2101]AXT_RT_AIO_OUTVALID_MODULE_NO :유효하지않은 AIO 모듈
* See error code Table for more outformation on status error codes

Description

H/W 타이머(Timer Trigger Mode, External Trigger Mode)를 이용한 연속적 데이터 수집 시에 FIFO에서 사용자 지정 버퍼로 데이터 처리를 위한 방법으로 설정한 인터럽트 소스를 확인합니다.
인터럽트 발생 기본 설정은 FIFO_HALF_FULL 이며 인터럽트 소스의 종류는 다음과 같습니다.

※참고 :SIO-AI4RB 는 지원하지 않습니다.

Enum	Explanation
SCAN_END(1)	설정된 채널 모두 A/D 변환을 할 때 마다 인터럽트가 발생하며 메모리 버퍼로 데이터 복사가 이루어 집니다. (저속 샘플링시 – 트리거 신호가 100~200 Hz 이하의 저속일 경우 사용 권장)
FIFO_HALF_FULL(2)	H/W FIFO가 반 이상 찼을 경우 인터럽트 발생하며 메모리 버퍼로 데이터 복사가 이루어 집니다.(고속 샘플링시 권장)

Example

C++

```
// 0번째모듈의하드웨어인터럽터마스크를확인합니다.
DWORD upMask;
AxaiInterruptGetModuleMask(0,&upMask);
```

Visual Basic

```
'0번째모듈의하드웨어인터럽터마스크를확인합니다.
Dim lMode As long
AxaiInterruptGetModuleMask0, lMode
```

Delphi

```
{ 0번째모듈의하드웨어인터럽터마스크를확인합니다. }
var
  dwMode : DWord;

begout
  AxaiInterruptGetModuleMask(0, @dwMode);
end;
```

C#

```
// 0번째모듈의하드웨어인터럽터마스크를확인합니다.
uint uMask;
CAXA.AxaiInterruptGetModuleMask(0,ref uMask);
```

See Also

AxaiEventSetChannel, AxaiEventSetChannelEnable, AxaiEventGetChannelEnable, AxaiEventSetMultiChannelEnable, AxaiEventSetChannelMask, AxaiEventGetChannelMask, AxaiEventSetMultiChannelMask, AxaiEventRead, AxaiInterruptGetModuleMask

입력모듈파라미터설정및확인

Function	Description
AxaiSetRange	지정한 입력 채널에 입력 전압 범위를 설정한다.
AxaiGetRange	지정한 입력 채널의 입력 전압 범위를 확인한다.
AxaiSetRangeModule	지정한 입력 모듈에 입력 전압 범위를 설정한다. (RTEX AI16F 전용)
AxaiGetRangeModule	지정한 입력 모듈의 입력 전압 범위를 확인한다. (RTEX AI16F 전용)
AxaiSetMultiRange	지정한 여러 입력 채널에 허용 입력 전압 범위를 설정한다.
AxaiHwSetBufferOverflowMode	지정한 입력 채널에 버퍼가 Full로 찼을 때 관리 방식을 설정한다.
AxaiHwGetBufferOverflowMode	지정한 입력 채널이 버퍼가 Full로 찼을 때 관리 방식을 확인한다.
AxaiHwSetMultiBufferOverflowMode	지정한 여러 입력 채널에 버퍼가 Full로 찼을 때 관리 방식을 설정한다.
AxaiHwSetLimit	지정한 입력 채널에 버퍼의 상한 값과 하한 값을 설정한다.
AxaiHwGetLimit	지정한 입력 채널에 버퍼의 상한 값과 하한 값을 확인한다.
AxaiHwSetMultiLimit	지정한 여러 입력 채널에 버퍼의 상한 값과 하한 값을 설정한다.
AxaiSetTriggerMode	지정한 입력 모듈에 트리거 모드를 설정한다.
AxaiGetTriggerMode	지정한 모듈에 트리거 모드를 확인한다.
AxaiHwSetSampleFreq	지정한 모듈에 샘플링 간격을 주파수 단위로 설정한다.
AxaiHwGetSampleFreq	지정한 모듈에 샘플링 간격을 주파수 단위로 설정된 값을 확인한다.
AxaiHwSetSamplePeriod	지정한 모듈에 샘플링 간격을 시간 단위로 설정한다.
AxaiHwGetSamplePeriod	지정한 모듈에 샘플링 간격을 시간 단위로 설정된 값을 확인한다.
AxaiSetModuleOffsetValue	지정한 입력 모듈의 Offset 값을 조절한다.
AxaiGetModuleOffsetValue	지정한 입력 모듈의 설정된 Offset값을 확인한다

AxaiSetRange

Purpose

지정한 입력 채널에 입력 전압 범위를 설정합니다.

Format

C++

```
DWORD AxaiSetRange(long lAdcChannelNo, double dMinVolt, double
dMaxVolt);
```

Visual Basic

```
Function AxaiSetRange(ByVal lAdcChannelNo As Long, ByVal dMinVolt As
Double, ByVal dMaxVolt As Double) As Long
```

Delphi

```
functionAxaiSetRange(lAdcChannelNo : LongInt; dMinVolt : Double;
dMaxVolt : Double) : DWord; stdcall;
```

C#

```
uint AxaiSetRange(int lChannelNo, double dMinVolt, double dMaxVolt);
```

Parameters

[in/out] Name	[Init Value] Explanation
[in]lAdcChannelNo	입력 채널 번호(0 ~ 입력 채널 개수 -1)
[in]dMinVolt	[0]최소 입력 전압
[in]dMaxVolt	[0]최대 입력 전압

Return Values

[0000] AXT_RT_SUCCESS :함수실행성공
[1053] AXT_RT_NOT_OPEN :라이브러리초기화실패
[2102] AXT_RT_AIO_INVALID_CHANNEL_NO :유효하지않은 AIO 채널번호
* See error code Table for more information on status error codes

Description

지정한 입력 채널에 허용 입력 전압 범위를 설정합니다. 입력 전압 범위는 A/D 변환 성능과 관련이 있으므로 측정 범위 확인 후 설정하시기 바랍니다. 모듈별로 허용 입력 전압이 정해져 있습니다. 사용자가 입력 전압 설정범위 이외 전압을 입력하면 자동으로 그 값을 포함하는 근삿값으로 자동 설정됩니다. 자동 설정된 값은 AxaiGetRange 함수를 이용해서 확인할 수 있습니다. 최소 입력 전압은 아날로그 입력 모듈의 허용전압 중 최소 전압을 의미합니다. 즉, 사용자는 이 전압보다 작은 전압을 입력할 수 없습니다. 그리고 사용자가 설정한 값은 하드웨어가 지원하는 최소 허용전압을 포함하는 가장 근사치 값으로 설정됩니다. SIO-AI4RB 의 경우 최소 기준 전압으로 -10V, -5V, 0V 를 제공하므로 사용자가 -8V 를 입력하면 -10V 로 자동 설정됩니다. 최대 입력 전압은 아날로그 입력의 허용 전압 중 최대 전압을 의미합니다. 즉, 사용자는 이 전압보다 높은 전압을 입력할 수 없다. 그리고 사용자가 설정한 값은 하드웨어가 지원하는 최대 허용전압을 포함하는 가장근사치 값으로 설정됩니다. SIO-AI4RB 의 경우 최대 허용전압으로 10V, 5V 를 제공하므로 사용자가 8V 를 입력하면 10V 로 자동

설정됩니다. 반면 분해능이 16bit 인 PCI-AI16HR 의 경우 입력전압의 범위가 고정되어 있으므로 함수 수행시 AXT_RT_AIO_INVALID_USE 에러가 반환됩니다.

※참고 :PCI-AI16HR 은 입력 전압 범위가 -10V ~ 10V 로 고정되어 있습니다.

입력 전압 설정 범위

Module	최소 전압	최대 전압	최소 측정 단위
SIO-AI4RB	-10V	10V	약 19.5mV
	-5V	5V	약 9.7mV
PCI-AI16HR(SIO-AI16HB) 입력전압범위고정	-10V	10V	약 1.2mV

Example

C++

```
// 0번입력채널에입력전압범위를설정합니다.
AxaiSetRange(0, -10, 10);
```

Visual Basic

```
' 0번입력채널에입력전압범위를설정합니다.
AxaiSetRange 0, -10, 10
```

Delphi

```
{ 0번입력채널에입력전압범위를설정합니다. }
AxaiSetRange(0, -10, 10);
```

C#

```
// 0번입력채널에입력전압범위를설정합니다.
CAXA.AxaiSetRange(0, -10, 10);
```

See Also

AxaiGetRange, AxaiSetMultiRange, AxaiHwSetBufferOverflowMode, AxaiHwGetBufferOverflowMode, AxaiHwSetMultiBufferOverflowMode, AxaiHwSetLimit,AxaiHwGetLimit, AxaiHwSetMultiLimit, AxaiSetTriggerMode, AxaiGetTriggerMode, AxaiHwSetSampleFreq, AxaiHwGetSampleFreq, AxaiHwSetSamplePeriod, AxaiHwGetSamplePeriod,AxaiSetModuleOffsetValue, AxaiGetModuleOffsetValue

AxaiGetRange

Purpose

지정한 입력 채널에 설정한 입력 전압 범위를 확인합니다.

Format

C++

```
DWORD AxaiGetRange(long lAdcChannelNo, double *dpMinVolt, double
*dpMaxVolt);
```

Visual Basic

```
Function AxaiGetRange(ByVal lAdcChannelNo As Long, ByRef dpMinVolt As
Double, ByRef dpMaxVolt As Double) As Long
```

Delphi

```
functionAxaiGetRange(lAdcChannelNo : LongInt; dpMinVolt : PDouble;
dpMaxVolt : PDouble) : DWord; stdcall;
```

C#

```
uint AxaiGetRange(int lChannelNo, ref double dpMinVolt, ref double
dpMaxVolt);
```

Parameters

[in/out] Name	[Init Value] Explanation
[in]lAdcChannelNo	입력 채널 번호 (0 ~ 입력 채널 개수 -1)
[out]dpMinVolt	[0]최소 입력 전압
[out]dpMaxVolt	[0]최대 입력 전압

Return Values

[0000] AXT_RT_SUCCESS :함수실행성공
[1053] AXT_RT_NOT_OPEN :라이브러리초기화실패
[2102] AXT_RT_AIO_INVALID_CHANNEL_NO :유효하지않은 AIO 채널번호
* See error code Table for more information on status error codes

Description

사용자가 AxaiSetRange함수를 이용해서 허용 입력 전압을 설정한 후 실제로 설정된 값을 확인합니다. 사용자가 H/W에서 제공되지 않는 허용 입력 전압을 설정하였을 경우 실제 설정 값과 다를 수 있습니다.

※참고 :PCI-AI16HR 은 입력 전압 범위가 -10V ~ 10V 로 고정되어 있습니다.

입력 전압 설정 범위

Module	최소 전압	최대 전압	최소 측정 단위
SIO-AI4RB	-10V	10V	약 19.5mV
	-5V	5V	약 9.7mV
PCI-AI16HR(SIO-AI16HB)	-10V	10V	약 1.2mV

입력전압범위고정			

Example

C++

```
// 0번입력채널의입력전압범위를확인합니다.
double  dMinVolt;
double  dMaxVolt;

AxaiGetRange(0, &dMinVolt, &dMaxVolt);
```

Visual Basic

```
' 0번입력채널의입력전압범위를확인합니다.
Dim lMask As Long

AxaiGetRange 0, dMinVolt, dMaxVolt
```

Delphi

```
{ 0번입력채널의입력전압범위를확인합니다. }
var
 dMinVolt : Double;
 dMaxVolt : Double;
 strData : String;

begin
  AxaiGetRange(0, @dMinVolt, @dMaxVolt);

 strData := 'MinVolt : ' + FloatToStr(dMinVolt) + ', MaxVolt : ' + loatToStr(dMaxVolt);
 Application.MessageBox (PCHAR(strData), 'Ajinextek', MB_OK);
end;
```

C#

```
// 0번입력채널의입력전압범위를확인합니다.
double  dMinVolt;
double  dMaxVolt;

CAXA.AxaiGetRange(0, ref dMinVolt, ref dMaxVolt);
```

See Also

AxaiSetRange, AxaiSetMultiRange, AxaiHwSetBufferOverflowMode, AxaiHwGetBufferOverflowMode, AxaiHwSetMultiBufferOverflowMode, AxaiHwSetLimit, AxaiHwGetLimit, AxaiHwSetMultiLimit, AxaiSetTriggerMode, AxaiGetTriggerMode, AxaiHwSetSampleFreq, AxaiHwGetSampleFreq, AxaiHwSetSamplePeriod, AxaiHwGetSamplePeriod,AxaiSetModuleOffsetValue, AxaiGetModuleOffsetValue

AxaiSetRangeModule

Purpose

지정한 입력 모듈에 입력 전압 범위를 설정합니다. (RTEX AI16F 전용)

Format

C++

```
DWORD AxaiSetRangeModule(long lModuleNo, double dMinVolt, double
dMaxVolt);
```

Visual Basic

```
Function AxaiSetRange(ByVal lModuleNo As Long, ByVal dMinVolt As
Double, ByVal dMaxVolt As Double) As Long
```

Delphi

```
functionAxaiSetRange(lModuleNo: LongInt; dMinVolt : Double; dMaxVolt :
Double) : DWord; stdcall;
```

C#

```
uint AxaiSetRange(int lModuleNo, double dMinVolt, double dMaxVolt);
```

Parameters

[in/out] Name	[Init Value] Explanation
[in] lModuleNo	입력 모듈 번호(0 ~ 입력 모듈 개수 -1)
[in]dMinVolt	[0]최소 입력 전압 (-5, -10)
[in]dMaxVolt	[0]최대 입력 전압 (+5, +10)

Return Values

[0000] AXT_RT_SUCCESS :함수실행성공
[1053] AXT_RT_NOT_OPEN :라이브러리초기화실패
[2101] AXT_RT_AIO_INVALID_MODULE_NO :유효하지않은 AIO 모듈
* See error code Table for more information on status error codes

Description

지정한 입력 모듈에 허용 입력 전압 범위를 설정합니다. 자동 설정된 값은 AxaiGetRangeMoudle 함수를 이용해서 확인할 수 있습니다. 최소 입력 전압은 아날로그 입력 모듈의 허용전압 중 최소 전압을 의미합니다. 즉, 사용자는 이 전압보다 작은 전압을 입력할 수 없습니다. 그리고 사용자가 설정한 값은 하드웨어가 지원하는 최소 허용전압을 포함하는 가장 근사치 값으로 설정됩니다. RTEX-AI16F 의 경우 최소 기준 전압으로 -10V, -5V 를 제공하므로 사용자가 -8V 를 입력하면 AXT_RT_NOT_SUPPORT_VERSION 을 Return 하게 됩니다. 최대 입력 전압은 아날로그 입력의 허용 전압 중 최대 전압을 의미합니다. 즉, 사용자는 이 전압보다 높은 전압을 입력할 수 없습니다. 그리고 사용자가 설정한 값은 하드웨어가 지원하는 최대 허용전압을 포함하는 가장근사치 값으로 설정됩니다. RTEX-AI16F 의 경우 최대 허용전압으로 10V, 5V 를 제공하므로 사용자가 8V 를 입력하면 AXT_RT_NOT_SUPPORT_VERSION 을 Return 하게 됩니다. 반면 분해능이 16bit 인 PCI-AI16HR 의 경우 입력전압의 범위가 고정되어 있으므로 함수 수행시 AXT_RT_AIO_INVALID_USE 에러가 반환됩니다.

※참고 :PCI-AI16HR 은 입력 전압 범위가 -10V ~ 10V 로 고정되어 있습니다.

입력 전압 설정 범위

Module	최소 전압	최대 전압	최소 측정 단위
SIO-AI4RB	-10V	10V	약 19.5 mV
	-5V	5V	약 9.7 mV
PCI-AI16HR(SIO-AI16HB) 입력전압범위고정	-10V	10V	약 1.2 mV
RTEX-AI16F	-10V	10V	약 305 μV
	-5V	5V	약 152 μV

Example

C++

```
// 0번입력채널에입력전압범위를설정합니다.
AxaiSetRangeModule(0, -10, 10);
```

Visual Basic

```
' 0번입력채널에입력전압범위를설정합니다.
AxaiSetRangeModule 0, -10, 10
```

Delphi

```
{ 0번입력채널에입력전압범위를설정합니다. }
AxaiSetRangeModule(0, -10, 10);
```

C#

```
// 0번입력채널에입력전압범위를설정합니다.
CAXA.AxaiSetRangeModule(0, -10, 10);
```

See Also

AxaiGetRangeModule, AxaiGetRange, AxaiSetMultiRange, AxaiHwSetBufferOverflowMode, AxaiHwGetBufferOverflowMode, AxaiHwSetMultiBufferOverflowMode, AxaiHwSetLimit,AxaiHwGetLimit, AxaiHwSetMultiLimit, AxaiSetTriggerMode, AxaiGetTriggerMode, AxaiHwSetSampleFreq, AxaiHwGetSampleFreq, AxaiHwSetSamplePeriod, AxaiHwGetSamplePeriod,AxaiSetModuleOffsetValue, AxaiGetModuleOffsetValue

AxaiGetRangeModule

Purpose

지정한 입력 모듈에 설정한 입력 전압 범위를 확인합니다.

Format

C++

```
DWORD AxaiGetRangeModule(long lModuleNo, double *dpMinVolt, double
*dpMaxVolt);
```

Visual Basic

```
Function AxaiGetRangeModule (ByVal lModuleNo As Long, ByRef dpMinVolt
As Double, ByRef dpMaxVolt As Double) As Long
```

Delphi

```
Function AxaiGetRangeModule (lModuleNo: LongInt; dpMinVolt : PDouble;
dpMaxVolt : PDouble) : DWord; stdcall;
```

C#

```
uint AxaiGetRangeModule (int lModuleNo, ref double dpMinVolt, ref
double dpMaxVolt);
```

Parameters

[in/out] Name	[Init Value] Explanation
[in]lAdcChannelNo	입력 모듈 번호(0 ~ 입력 모듈 개수 -1)
[out]dpMinVolt	[0]최소 입력 전압 (-5, -10)
[out]dpMaxVolt	[0]최대 입력 전압 (+5, +10)

Return Values

[0000] AXT_RT_SUCCESS :함수실행성공
[1053] AXT_RT_NOT_OPEN :라이브러리초기화실패
[2101] AXT_RT_AIO_INVALID_MODULE_NO :유효하지않은 AIO 모듈
* See error code Table for more information on status error codes

Description

사용자가 AxaiSetRangeModule 함수를 이용해서 허용 입력 전압을 설정한 후 실제로 설정된 값을 확인합니다. 사용자가 H/W에서 제공되지 않는 허용 입력 전압을 설정하였을 경우 실제 설정 값과 다를 수 있습니다.

※참고 :PCI-AI16HR 은 입력 전압 범위가 -10V ~ 10V 로 고정되어 있습니다.

입력 전압 설정 범위

Module	최소 전압	최대 전압	최소 측정 단위
SIO-AI4RB	-10V	10V	약 19.5mV
	-5V	5V	약 9.7mV

PCI-AI16HR(SIO-AI16HB) 입력전압범위고정	-10V	10V	약 1.2mV
RTEX-AI16F	-10V	10V	약 305 μV
	-5V	5V	약 152 μV

Example

C++

```
// 0번입력채널의입력전압범위를확인합니다.
double  dMinVolt;
double  dMaxVolt;

AxaiGetRangeModule(0, &dMinVolt, &dMaxVolt);
```

Visual Basic

```
' 0번입력채널의입력전압범위를확인합니다.
Dim lMask As Long

AxaiGetRangeModule 0, dMinVolt, dMaxVolt
```

Delphi

```
{ 0번입력채널의입력전압범위를확인합니다. }
var
  dMinVolt : Double;
  dMaxVolt : Double;
  strData : String;

begin
   AxaiGetRangeModule (0, @dMinVolt, @dMaxVolt);

  strData := 'MinVolt : ' + FloatToStr(dMinVolt) + ', MaxVolt : ' + loatToStr(dMaxVolt);
  Application.MessageBox (PCHAR(strData), 'Ajinextek', MB_OK);
end;
```

C#

```
// 0번입력채널의입력전압범위를확인합니다.
double  dMinVolt;
double  dMaxVolt;

CAXA.AxaiGetRangeModule (0, ref dMinVolt, ref dMaxVolt);
```

See Also

AxaiSetRangeModule,AxaiSetRange, AxaiSetMultiRange, AxaiHwSetBufferOverflowMode, AxaiHwGetBufferOverflowMode, AxaiHwSetMultiBufferOverflowMode, AxaiHwSetLimit, AxaiHwGetLimit, AxaiHwSetMultiLimit, AxaiSetTriggerMode, AxaiGetTriggerMode, AxaiHwSetSampleFreq, AxaiHwGetSampleFreq, AxaiHwSetSamplePeriod, AxaiHwGetSamplePeriod,AxaiSetModuleOffsetValue, AxaiGetModuleOffsetValue

AxaiSetMultiRange

Purpose

지정한 여러 입력 채널에 허용 입력 전압 범위를 설정합니다.

Format

C++

```
DWORD AxaiSetMultiRange(long lSize, long *lpAdcChannelNo, double
dMinVolt, double dMaxVolt);
```

Visual Basic

```
Function AxaiSetMultiRange(ByVal lSize As Long, ByRef lpAdcChannelNo
As Long, ByVal dMinVolt As Double, ByVal dMaxVolt As Double) As Long
```

Delphi

```
functionAxaiSetMultiRange(lSize : LongInt; lpAdcChannelNo : PLongInt;
dMinVolt : Double; dMaxVolt : Double) : DWord; stdcall;
```

C#

```
uint AxaiSetMultiRange(int lSize, ref int lpChannelNo, double
dMinVolt, double dMaxVolt);
```

Parameters

[in/out] Name	[Init Value] Explanation
[in]lSize	사용할 입력 채널의 개수
[in]lpAdcChannelNo	사용할 채널 번호의 배열
[in]dMinVolt	[0]최소 입력 전압
[in]dMaxVolt	[0]최대 입력 전압

Return Values

[0000] AXT_RT_SUCCESS :함수실행성공
[1053] AXT_RT_NOT_OPEN :라이브러리초기화실패
[2102] AXT_RT_AIO_INVALID_CHANNEL_NO :유효하지않은 AIO 채널번호
* See error code Table for more information on status error codes

Description

지정한 여러 입력 채널의 허용 입력 전압 범위를 설정합니다. 아날로그 입력 모듈의 허용 입력 전압 범위를 여러 채널 동시에 설정합니다. 입력 전압 범위는 A/D변환 성능과 관련이 있으므로 측정 범위 확인 후 설정하시기 바랍니다. 모듈별로 허용 입력 전압이 정해져 있습니다. 사용자가 입력 전압 설정범위 이외 전압을 입력하면 자동으로 그 값을 포함하는 근삿값으로 자동 설정됩니다. 자동 설정된 값은 AxaiGetRange를 이용해서 확인하면 됩니다. 최소 입력 전압은 아날로그 입력 모듈의 허용전압 중 최소 전압을 의미합니다. 즉, 사용자는 이 전압보다 작은 전압을 입력할 수 없다. 그리고 사용자가 설정한 값은 하드웨어가 지원하는 최소 허용전압을 포함하는 가장 근사치 값으로 설정됩니다. SIO-AI4RB의 경우 최소 기준 전압으로 -10V, -5V, 0V를 제공하므로 사용자가 -8V를 입력하면 -10V로 자동 설정됩니다. 최대 입력 전압은 아날로그 입력의 허용 전압 중 최대 전압을 의미합니다. 즉 사용자는 이 전압보다 높은 전압을 입력할 수 없다. 그리고 사용자가 설정한 값은 하드웨어가 지원하는 최대

허용전압을 포함하는 가장근사치 값으로 설정됩니다. SIO-AI4R의 경우 최대 허용전압으로 10V, 5V를 제공하므로 사용자가 8V를 입력하면 10V로 자동 설정됩니다. 반면 분해능이 16bit 인 PCI-AI16HR의 경우 입력전압의 범위가 고정되어 있으므로 함수 수행시 AXT_RT_AIO_INVALID_USE 에러가 반환됩니다.

입력 전압 설정 범위

Module	최소 전압	최대 전압	최소 측정 단위
SIO-AI4RB	-10V	10V	약 19.5mV
	-5V	5V	약 9.7mV
PCI-AI16HR(SIO-AI16HB) 입력전압범위고정	-10V	10V	약 1.2mV

Example

C++

```
// 여러입력채널에입력전압범위를설정합니다.
long lAdcChannelNo[4] = {0, 1, 2, 3};

AxaiSetMultiRange(4, lAdcChannelNo, -10, 10);
```

Visual Basic

```
'여러입력채널에입력전압범위를설정합니다.
Dim lAdcChannelNo(0 To 3) As Long

lAdcChannelNo(0) = 0
lAdcChannelNo(1) = 1
lAdcChannelNo(2) = 2
lAdcChannelNo(3) = 3

AxaiSetMultiRange 4, lAdcChannelNo(0), -10, 10
```

Delphi

```
{ 여러입력채널에입력전압범위를설정합니다. }
var
  lAdcChannelNo: array [0..3] of LongInt;

begin
  lAdcChannelNo[0]:= 0;
  lAdcChannelNo[1]:= 1;
  lAdcChannelNo[2]:= 2;
  lAdcChannelNo[3]:= 3;

  AxaiSetMultiRange(4, @lAdcChannelNo, -10, 10);
end;
```

C#

```
// 여러입력채널에입력전압범위를설정합니다.
int[] nAdcChannelNo = new int[4];

nAdcChannelNo[0] = 0;
```

```
nAdcChannelNo[1] = 1;
nAdcChannelNo[2] = 2;
nAdcChannelNo[3] = 3;

CAXA.AxaiSetMultiRange(4, ref nAdcChannelNo[0], -10, 10);
```

See Also

AxaiSetRange, AxaiGetRange, AxaiHwSetBufferOverflowMode, AxaiHwGetBufferOverflowMode, AxaiHwSetMultiBufferOverflowMode, AxaiHwSetLimit, AxaiHwGetLimit, AxaiHwSetMultiLimit, AxaiSetTriggerMode, AxaiGetTriggerMode, AxaiHwSetSampleFreq, AxaiHwGetSampleFreq, AxaiHwSetSamplePeriod, AxaiHwGetSamplePeriod, AxaiSetModuleOffsetValue, AxaiGetModuleOffsetValue

AxaiHwSetBufferOverflowMode

Purpose

지정한 입력 채널에 버퍼가 Full 로 찼을 때 데이터 관리 방식을 설정합니다.

Format

C++

```
DWORD AxaiHwSetBufferOverflowMode(long lAdcChannelNo, DWORD
uFullMode);
```

Visual Basic

```
Function AxaiHwSetBufferOverflowMode(ByVal lAdcChannelNo As Long,
ByVal uFullMode As Long) As Long
```

Delphi

```
functionAxaiHwSetBufferOverflowMode(lAdcChannelNo : LongInt;
uFullMode : DWord) : DWord; stdcall;
```

C#

```
uint AxaiHwSetBufferOverflowMode(int lChannelNo, uint uFullMode);
```

Parameters

[in/out] Name	[Init Value] Explanation
[in]lAdcChannelNo	입력 채널 번호 (0 ~ 입력 채널 개수 -1)
[in]uFullMode	[00h]데이터 버퍼 모드: AXT_AIO_FULL_MODE - [00h] 새로운 데이터 유지 - [01h] 이전 데이터 유지

Return Values

[0000] AXT_RT_SUCCESS :함수실행성공
[1053] AXT_RT_NOT_OPEN :라이브러리초기화실패
[2102] AXT_RT_AIO_INVALID_CHANNEL_NO :유효하지않은 AIO 채널번호
* See error code Table for more information on status error codes

Description

H/W 타이머(Timer Trigger Mode, External Trigger Mode)를 이용한 A/D 변환 시 사용됩니다. 버퍼가 Full 상태일 때uFullMode 설정에 따라 최초 수집된 데이터에를 지우고 새로운 데이터를 저장하거나 최초 데이터를 유지하고 새로운 데이터를 버리게 됩니다.

Enum	Explanation
NEW_DATA_KEEP(0)	버퍼가 가득 찼을 때 기존 데이터에 새로운 데이터를 덮어씁니다.
CURR_DATA_KEEP(1)	버퍼가 가득 찼을 때 기존 데이터를 유지하고 새로운 데이터를 버립니다.

Example

C++

```
// 0번입력채널에버퍼가 Full로찼을때새로운데이터가유지되도록관리방식을설정합니다.
AxaiHwSetBufferOverflowMode(0, NEW_DATA_KEEP);
```

Visual Basic

```
' 0번입력채널에버퍼가 Full로찼을때새로운데이터가유지되도록관리방식을설정합니다.
AxaiHwSetBufferOverflowMode 0, NEW_DATA_KEEP
```

Delphi

```
{ 0번입력채널에버퍼가 Full로찼을때새로운데이터가유지되도록관리방식을설정합니다. }
AxaiHwSetBufferOverflowMode(0, NEW_DATA_KEEP);
```

C#

```
// 0번입력채널에버퍼가 Full로찼을때새로운데이터가유지되도록관리방식을설정합니다.
CAXA.AxaiHwSetBufferOverflowMode(0, (uint)AXT_AIO_FULL_MODE.NEW_DATA_KEEP);
```

See Also

AxaiSetRange, AxaiGetRange, AxaiSetMultiRange, AxaiHwGetBufferOverflowMode, AxaiHwSetMultiBufferOverflowMode, AxaiHwSetLimit, AxaiHwGetLimit, AxaiHwSetMultiLimit, AxaiSetTriggerMode, AxaiGetTriggerMode, AxaiHwSetSampleFreq, AxaiHwGetSampleFreq, AxaiHwSetSamplePeriod, AxaiHwGetSamplePeriod, AxaiSetModuleOffsetValue, AxaiGetModuleOffsetValue

AxaiHwGetBufferOverflowMode

Purpose

지정한 입력 채널이 버퍼가 가득 찼을 때 데이터 관리 방식을 확인합니다.

Format

C++

```
DWORD AxaiHwGetBufferOverflowMode(long lAdcChannelNo, DWORD
*upFullMode);
```

Visual Basic

```
Function AxaiHwGetBufferOverflowMode(ByVal lAdcChannelNo As Long,
ByRef upFullMode As Long) As Long
```

Delphi

```
functionAxaiHwGetBufferOverflowMode(lAdcChannelNo : LongInt;
upFullMode : PDWord) : DWord; stdcall;
```

C#

```
uint AxaiHwGetBufferOverflowMode(int lChannelNo, ref uint upFullMode);
```

Parameters

[in/out] Name	[Init Value] Explanation
[in]lAdcChannelNo	입력 채널 번호 (0 ~ 입력 채널 개수 -1)
[out]upFullMode	[00h]데이터 버퍼 모드: AXT_AIO_FULL_MODE - [00h] 새로운 데이터 유지 - [01h] 이전 데이터 유지

Return Values

[0000] AXT_RT_SUCCESS :함수실행성공
[1053] AXT_RT_NOT_OPEN :라이브러리초기화실패
[2102] AXT_RT_AIO_INVALID_CHANNEL_NO :유효하지않은 AIO 채널번호
* See error code Table for more information on status error codes

Description

H/W 타이머(Timer Trigger Mode, External Trigger Mode)를 이용한 A/D 변환 시 사용됩니다. 버퍼가 Full상태일 때 uFullMode 설정에 따라 최초 수집된 데이터에를 지우고 새로운 데이터를 저장하거나 최초 데이터를 유지하고 새로운 데이터를 버리게 됩니다.

Enum	Explanation
NEW_DATA_KEEP(0)	버퍼가 가득 찼을 때 기존 데이터에 새로운 데이터를 덮어씁니다.
CURR_DATA_KEEP(1)	버퍼가 가득 찼을 때 기존 데이터를 유지하고 새로운 데이터를 버립니다.

Example

C++

```
// 0번입력채널에버퍼가 Full로찼을때관리방식을확인합니다.
DWORD dwFullMode;

AxaiHwGetBufferOverflowMode(0, &dwFullMode);
```

Visual Basic

```
' 0번입력채널에버퍼가 Full로찼을때관리방식을확인합니다.
Dim lFullMode As Long

AxaiHwGetBufferOverflowMode 0, dwFullMode
```

Delphi

```
{ 0번입력채널에버퍼가 Full로찼을때관리방식을확인합니다. }
var
  dwFullMode: DWord;

begin
  AxaiHwGetBufferOverflowMode(0, @dwFullMode);
  Application.MessageBox (PCHAR(IntToStr(dwFullMode)), 'Ajinextek', MB_OK);
end;
```

C#

```
// 0번입력채널에버퍼가 Full로찼을때관리방식을확인합니다.
uint uFullMode;

CAXA.AxaiHwGetBufferOverflowMode(0, ref uFullMode);
```

See Also

AxaiSetRange, AxaiGetRange, AxaiSetMultiRange, AxaiHwSetBufferOverflowMode, AxaiHwSetMultiBufferOverflowMode, AxaiHwSetLimit, AxaiHwGetLimit, AxaiHwSetMultiLimit, AxaiSetTriggerMode, AxaiGetTriggerMode, AxaiHwSetSampleFreq, AxaiHwGetSampleFreq, AxaiHwSetSamplePeriod, AxaiHwGetSamplePeriod,AxaiSetModuleOffsetValue, AxaiGetModuleOffsetValue

AxaiHwSetMultiBufferOverflowMode

Purpose

지정한 여러 입력 채널에 버퍼가 가득 찼을 때 관리 방식을 설정합니다.

Format

C++

```
DWORD AxaiHwSetMultiBufferOverflowMode(long lSize, long
*lpAdcChannelNo, DWORD uFullMode);
```

Visual Basic

```
Function AxaiHwSetMultiBufferOverflowMode(ByVal lSize As Long, ByRef
lpAdcChannelNo As Long, ByVal uFullMode As Long) As Long
```

Delphi

```
functionAxaiHwSetMultiBufferOverflowMode(lSize : LongInt;
lpAdcChannelNo : PLongInt; uFullMode : DWord) : DWord; stdcall;
```

C#

```
uint AxaiHwSetMultiBufferOverflowMode(int lSize, ref int lpChannelNo,
uint uFullMode);
```

Parameters

[in/out] Name	[Init Value] Explanation
[in]lSize	사용할 입력 채널의 개수
[in]lpAdcChannelNo	사용할 채널 번호의 배열
[in]uFullMode	[00h]데이터 버퍼 모드: AXT_AIO_FULL_MODE - [00h] 새로운 데이터 유지 - [01h] 이전 데이터 유지

Return Values

[0000] AXT_RT_SUCCESS :함수실행성공
[1053] AXT_RT_NOT_OPEN :라이브러리초기화실패
[2102] AXT_RT_AIO_INVALID_CHANNEL_NO :유효하지않은 AIO 채널번호
* See error code Table for more information on status error codes

Description

H/W 타이머(Timer Trigger Mode, External Trigger Mode)를 이용한 A/D 변환 시 사용됩니다. 버퍼가 Full상태일 때 uFullMode 설정에 따라 최초 수집된 데이터에를 지우고 새로운 데이터를 저장하거나 최초 데이터를 유지하고 새로운 데이터를 버리게 됩니다.

Example

C++

```
// 여러입력채널에버퍼가 Full로찼을때새로운데이터가유지되도록관리방식을설정합니다.
long lAdcChannelNo[4] = {0, 1, 2, 3};
```

```
AxaiHwSetMultiBufferOverflowMode(4, lAdcChannelNo, NEW_DATA_KEEP);
```

Visual Basic

```
'여러입력채널에버퍼가 Full로찼을때새로운데이터가유지되도록관리방식을설정합니다.
Dim lAdcChannelNo(0 To 3) As Long

lAdcChannelNo(0) = 0
lAdcChannelNo(1) = 1
lAdcChannelNo(2) = 2
lAdcChannelNo(3) = 3

AxaiHwSetMultiBufferOverflowMode 4, lAdcChannelNo(0), NEW_DATA_KEEP
```

Delphi

```
{ 여러입력채널에버퍼가 Full로찼을때새로운데이터가유지되도록관리방식을설정합니다. }
var
  lAdcChannelNo: array [0..3] of LongInt;

begin
  lAdcChannelNo[0]:= 0;
  lAdcChannelNo[1]:= 1;
  lAdcChannelNo[2]:= 2;
  lAdcChannelNo[3]:= 3;

  AxaiHwSetMultiBufferOverflowMode(4, @lAdcChannelNo, NEW_DATA_KEEP);
end;
```

C#

```
// 여러입력채널에버퍼가 Full로찼을때새로운데이터가유지되도록관리방식을설정합니다.
int[] nAdcChannelNo = new int[4];

nAdcChannelNo[0] = 0;
nAdcChannelNo[1] = 1;
nAdcChannelNo[2] = 2;
nAdcChannelNo[3] = 3;

CAXA.AxaiHwSetMultiBufferOverflowMode(4, ref lAdcChannelNo[0],
(uint)AXT_AIO_FULL_MODE.NEW_DATA_KEEP);
```

See Also

AxaiSetRange, AxaiGetRange, AxaiSetMultiRange, AxaiHwSetBufferOverflowMode, AxaiHwGetBufferOverflowMode, AxaiHwSetLimit, AxaiHwGetLimit, AxaiHwSetMultiLimit, AxaiSetTriggerMode, AxaiGetTriggerMode, AxaiHwSetSampleFreq, AxaiHwGetSampleFreq, AxaiHwSetSamplePeriod, AxaiHwGetSamplePeriod, AxaiSetModuleOffsetValue, AxaiGetModuleOffsetValue

AxaiHwSetLimit

Purpose

지정한 입력 채널에 버퍼의 상한 값과 하한값을 설정합니다.

Format

C++

```
DWORD AxaiHwSetLimit(long lAdcChannelNo, longlLowLimit, longlUpLimit);
```

Visual Basic

```
Function AxaiHwSetLimit(ByVal lAdcChannelNo As Long, ByVal lLowLimit
As Long, ByVal lUpLimit As Long) As Long
```

Delphi

```
functionAxaiHwSetLimit(lAdcChannelNo : LongInt; lLowLimit : LongInt;
lUpLimit : LongInt) : DWord; stdcall;
```

C#

```
uint AxaiHwSetLimit(int lChannelNo, int lLowLimit, int lUpLimit);
```

Parameters

[in/out] Name	[Init Value] Explanation
[in]lAdcChannelNo	입력 채널 번호
[in]lLowLimit	[10]버퍼의 하한값
[in]lUpLimit	[100]버퍼의상한값 상한값은 AxaiHwStartMultiChannel의 lBuffSize 값보다 클 수 없습니다.

Return Values

[0000] AXT_RT_SUCCESS :함수실행성공
[1053] AXT_RT_NOT_OPEN :라이브러리초기화실패
[2102] AXT_RT_AIO_INVALID_CHANNEL_NO :유효하지않은 AIO 채널번호
* See error code Table for more information on status error codes

Description

H/W 타이머(Timer Trigger Mode, External Trigger Mode)를 이용한 A/D 변환 시 사용됩니다. 지정 입력 채널에 버퍼의 하한값과 상한값을 설정하여 적재된 데이터 개수가 상한값 보다 많거나 하한값 보다 작을 때 이벤트를 발생시켜 수집된 데이터를 처리할 수 있습니다.

Example

C++

```
// 0번입력채널에버퍼의상한값과하한값을설정한다.
AxaiHwSetLimit(0, 0, 1000);
```

Visual Basic

```
' 0번입력채널에버퍼의상한값과하한값을설정한다.
AxaiHwSetLimit 0, 0, 1000
```

Delphi

```
{ 0번입력채널에버퍼의상한값과하한값을설정한다. }
AxaiHwSetLimit(0, 0, 1000);
```

C#

```
// 0번입력채널에버퍼의상한값과하한값을설정한다.
CAXA.AxaiHwSetLimit(0, 0, 1000);
```

See Also

AxaiSetRange, AxaiGetRange, AxaiSetMultiRange, AxaiHwSetBufferOverflowMode, AxaiHwGetBufferOverflowMode, AxaiHwSetMultiBufferOverflowMode, AxaiHwGetLimit, AxaiHwSetMultiLimit, AxaiSetTriggerMode, AxaiGetTriggerMode, AxaiHwSetSampleFreq, AxaiHwGetSampleFreq, AxaiHwSetSamplePeriod, AxaiHwGetSamplePeriod, AxaiSetModuleOffsetValue, AxaiGetModuleOffsetValue

AxaiHwGetLimit

Purpose

지정한 입력 채널에 설정한 버퍼의 상한값과 하한값을 확인합니다.

Format

C++

```
DWORD AxaiHwGetLimit(long lAdcChannelNo, long*lpLowLimit,
long*lpUpLimit);
```

Visual Basic

```
Function AxaiHwGetLimit(ByVal lAdcChannelNo As Long, ByRef lpLowLimit
As Long, ByRef lpUpLimit As Long) As Long
```

Delphi

```
functionAxaiHwGetLimit(lAdcChannelNo : LongInt; lpLowLimit : PLongInt;
lpUpLimit : PLongInt) : DWord; stdcall;
```

C#

```
uint AxaiHwGetLimit(int lChannelNo, ref int lpLowLimit, ref int
lpUpLimit);
```

Parameters

[in/out] Name	[Init Value] Explanation
[in]lAdcChannelNo	입력 채널 번호
[out]lpLowLimit	버퍼의 하한값
[out]lpUpLimit	버퍼의 상한값

Return Values

[0000] AXT_RT_SUCCESS :함수실행성공
[1053] AXT_RT_NOT_OPEN :라이브러리초기화실패
[2102] AXT_RT_AIO_INVALID_CHANNEL_NO :유효하지않은 AIO 채널번호
* See error code Table for more information on status error codes

Description

H/W 타이머(Timer Trigger Mode, External Trigger Mode)를 이용한 A/D 변환 시 사용됩니다. 지정 입력 채널에 설정한 버퍼의 하한값과 상한값을 확인합니다. 적재된 데이터 개수가 상한값 보다 많거나 하한값 보다 작을 때 이벤트를 발생시켜 수집된 데이터를 처리할 수 있습니다.

Example

C++

```
// 0번입력채널에버퍼의상한값과하한값을확인한다.
long lLowLimit = 0;
long lUpLimit = 0;
```

```
AxaiHwGetLimit(0, &lLowLimit, &lUpLimit);
```

Visual Basic

```
' 0번입력채널에버퍼의상한값과하한값을확인한다.
Dim lLowLimit As Long
Dim lUpLimit As Long

AxaiHwGetLimit 0, lLowLimit, lUpLimit
```

Delphi

```
{ 0번입력채널에버퍼의상한값과하한값을확인한다. }
var
  lLowLimit: LongInt;
  lUpLimit: LongInt;
  strData : String;

begin
  AxaiHwGetLimit(0, @lLowLimit, @lUpLimit);
  strData := 'LowLimit : ' + FloatToStr(lLowLimit) + ', UpLimit : ' +
     FloatToStr(lUpLimit);
  Application.MessageBox (PCHAR(strData), 'Ajinextek', MB_OK);
end;
```

C#

```
// 0번입력채널에버퍼의상한값과하한값을확인한다.
int nLowLimit = 0;
int nUpLimit = 0;

CAXA.AxaiHwGetLimit(0, ref nLowLimit, ref nUpLimit);
```

See Also

AxaiSetRange, AxaiGetRange, AxaiSetMultiRange, AxaiHwSetBufferOverflowMode, AxaiHwGetBufferOverflowMode, AxaiHwSetMultiBufferOverflowMode, AxaiHwSetLimit, AxaiHwSetMultiLimit, AxaiSetTriggerMode, AxaiGetTriggerMode, AxaiHwSetSampleFreq, AxaiHwGetSampleFreq, AxaiHwSetSamplePeriod, AxaiHwGetSamplePeriod, AxaiSetModuleOffsetValue, AxaiGetModuleOffsetValue

AxaiHwSetMultiLimit

Purpose

지정한 여러 입력 채널에 버퍼의 상한 값과 하한값을 설정합니다.

Format

C++

```
DWORD AxaiHwSetMultiLimit(long lSize, long *lpAdcChannelNo,
longlLowLimit, longlUpLimit);
```

Visual Basic

```
Function AxaiHwSetMultiLimit(ByVal lSize As Long, ByRef
lpAdcChannelNo As Long, ByVal lLowLimit As Long, ByVal lUpLimit As
Long) As Long
```

Delphi

```
functionAxaiHwSetMultiLimit(lSize : LongInt; lpAdcChannelNo :
PLongInt; lLowLimit : LongInt; lUpLimit : LongInt) : DWord; stdcall;
```

C#

```
uint AxaiHwSetMultiLimit(int lSize, ref int lpChannelNo, int
lLowLimit, int lUpLimit);
```

Parameters

[in/out] Name	[Init Value] Explanation
[in]lSize	사용할 입력 채널의 개수
[in]lpAdcChannelNo	사용할 채널 번호의 배열
[in]lLowLimit	[10]버퍼의 하한값
[in]lUpLimit	[100]버퍼의 상한값

Return Values

[0000] AXT_RT_SUCCESS :함수실행성공
[1053] AXT_RT_NOT_OPEN :라이브러리초기화실패
[2102] AXT_RT_AIO_INVALID_CHANNEL_NO :유효하지않은 AIO 채널번호
* See error code Table for more information on status error codes

Description

H/W 타이머(Timer Trigger Mode, External Trigger Mode)를 이용한 A/D 변환 시 사용됩니다. 여러 입력 채널에 버퍼의 하한값과 상한값을 동시에 설정하여 적재된 데이터 개수가 상한값 보다 많거나 하한값 보다 작을 때 이벤트를 발생시켜 수집된 데이터를 처리할 수 있습니다.

Example

C++

```
// 여러입력채널에버퍼의상한값과하한값을설정한다.
long lAdcChannelNo[4] = {0, 1, 2, 3};
```

```
AxaiHwSetMultiLimit(4, lAdcChannelNo, 0, 1000);
```

Visual Basic

```
'여러입력채널에버퍼의상한값과하한값을설정한다.
Dim lAdcChannelNo(0 To 3) As Long

lAdcChannelNo(0) = 0
lAdcChannelNo(1) = 1
lAdcChannelNo(2) = 2
lAdcChannelNo(3) = 3

AxaiHwSetMultiLimit 4, lAdcChannelNo(0), 0, 1000
```

Delphi

```
{ 여러입력채널에버퍼의상한값과하한값을설정한다. }
var
  lAdcChannelNo: array [0..3] of LongInt;

begin
  lAdcChannelNo[0]:= 0;
  lAdcChannelNo[1]:= 1;
  lAdcChannelNo[2]:= 2;
  lAdcChannelNo[3]:= 3;

  AxaiHwSetMultiLimit(4, @lAdcChannelNo, 0, 1000);
end;
```

C#

```
// 여러입력채널에버퍼의상한값과하한값을설정한다.
int nAdcChannelNo[4] = {0, 1, 2, 3};

CAXA.AxaiHwSetMultiLimit(4, ref nAdcChannelNo[0], 0, 1000);
```

See Also

AxaiSetRange, AxaiGetRange, AxaiSetMultiRange, AxaiHwSetBufferOverflowMode, AxaiHwGetBufferOverflowMode, AxaiHwSetMultiBufferOverflowMode, AxaiHwSetLimit, AxaiHwGetLimit, AxaiSetTriggerMode, AxaiGetTriggerMode, AxaiHwSetSampleFreq, AxaiHwGetSampleFreq, AxaiHwSetSamplePeriod, AxaiHwGetSamplePeriod, AxaiSetModuleOffsetValue, AxaiGetModuleOffsetValue

AxaiSetTriggerMode

Purpose

지정한 입력 모듈에 A/D 변환 방식을 설정합니다.

Format

C++

```
DWORD AxaiSetTriggerMode(long lModuleNo, DWORD uTriggerMode);
```

Visual Basic

```
Function AxaiSetTriggerMode(ByVal lModuleNo As Long, ByVal
uTriggerMode As Long) As Long
```

Delphi

```
functionAxaiSetTriggerMode(lModuleNo : LongInt; uTriggerMode :
DWord) : DWord; stdcall;
```

C#

```
uint AxaiSetTriggerMode(int lModuleNo, uint uTriggerMode);
```

Parameters

[in/out] Name	[Init Value] Explanation
[in]lModuleNo	입력 모듈 번호
[in]uTriggerMode	[01h]A/D변환 방식

Return Values

[0000] AXT_RT_SUCCESS :함수실행성공
[1053] AXT_RT_NOT_OPEN :라이브러리초기화실패
[2101] AXT_RT_AIO_INVALID_MODULE_NO :유효하지않은 AIO 모듈
* See error code Table for more information on status error codes

Description

지정한 입력 모듈에 A/D 변환 방식을 설정하는 함수로 A/D 변환 전 반드시 호출해야 합니다. A/D 변환 방식은 사용자가 원하는 시점에서 Sampling을 하여 A/D변환하는 Normal Trigger방식과 H/W 내부 Timer를 이용해서 일정한 주기로 A/D변환하는Timer Trigger방식과 외부 신호가 입력될 때 A/D변환하는 External Trigger방식이 있습니다. 내부 Timer를 이용하는 Timer Trigger 방식 사용 시 반드시 AxaiHwSetSampleFreq 또는 AxaiHwSetSamplePeriod 함수를 이용하여 Sampling Rate를 설정하여야 합니다. A/D 변환 방식에 따른 Trigger Mode 설정은 아래 Table과 같습니다.

Trigger Mode 설정

Enum	Value	Explanation
NORMAL_MODE	1	사용자가 원하는 시점에 A/D 변환하는 Software Trigger 방식
TIMER_MODE	2	H/W의 내부 클럭을 이용해서 A/D 변환하는 Trigger 방식
EXTERNAL_MODE	3	외부 입력단자의 클럭을 이용해서 A/D 변환하는 Trigger 방식

Trigger Mode 관련 함수

Trigger Mode	주요 관련 함수
Normal Trigger Mode	AxaiSwReadVoltage, AxaiSwReadMultiVoltage
Timer Trigger Mode	(함수 그룹 : AxaiHw.., AxaiEvent..,AxaiInterrupt..) AxaiEventSetChannel AxaiHwSetSampleFreq, AxaiHwSetSamplePeriod AxaiHwStartMultiChannel, AxaiHwStartMultiFilter, AxaiHwStopMultiChannel, AxaiHwSetLimit, AxaiHwReadDataLength AxaiHwReadSampleVoltage AxaiHwSetMultiAccess(ImmediateMode), AxaiHwStartMultiAccess(Immediate Mode) Etc..
External Trigger Mode	Timer Trigger Mode와 동일하나 Immediate Mode 함수와 AxaiHwSetSampleFreq / AxaiHwSetSamplePeriod 함수는 해당 사항 없음.

Example

C++

```
// 0번입력모듈에 Normal Trigger 모드로설정합니다.
AxaiSetTriggerMode(0, NORMAL_MODE);
```

Visual Basic

```
'0번입력모듈에 Normal Trigger 모드로설정합니다.
AxaiSetTriggerMode0, NORMAL_MODE
```

Delphi

```
{ 0번입력모듈에 Normal Trigger 모드로설정합니다. }
AxaiSetTriggerMode(0, NORMAL_MODE);
```

C#

```
// 0번입력모듈에 Normal Trigger 모드로설정합니다.
CAXA.AxaiSetTriggerMode(0, (uint)AXT_AXIO_TRIGGER_MODE.NORMAL_MODE);
```

See Also

AxaiSetRange, AxaiGetRange, AxaiSetMultiRange, AxaiHwSetBufferOverflowMode, AxaiHwGetBufferOverflowMode, AxaiHwSetMultiBufferOverflowMode, AxaiHwSetLimit, AxaiHwGetLimit, AxaiHwSetMultiLimit, AxaiGetTriggerMode, AxaiHwSetSampleFreq, AxaiHwGetSampleFreq, AxaiHwSetSamplePeriod, AxaiHwGetSamplePeriod, AxaiSetModuleOffsetValue, AxaiGetModuleOffsetValue

AxaiGetTriggerMode

Purpose

지정한 입력 모듈에 설정한 A/D 변환 방식을 확인합니다.

Format

C++

```
DWORD AxaiGetTriggerMode(long lModuleNo, DWORD *upTriggerMode);
```

Visual Basic

```
Function AxaiGetTriggerMode(ByVal lModuleNo As Long, ByRef
upTriggerMode As Long) As Long
```

Delphi

```
functionAxaiGetTriggerMode(lModuleNo : LongInt; upTriggerMode :
PDWord) : DWord; stdcall;
```

C#

```
uint AxaiGetTriggerMode(int lModuleNo, ref uint upTriggerMode);
```

Parameters

[in/out] Name	[Init Value] Explanation
[in]lModuleNo	입력 모듈 번호
[out]upTriggerMode	A/D 변환 방식

Return Values

[0000] AXT_RT_SUCCESS :함수실행성공
[1053] AXT_RT_NOT_OPEN :라이브러리초기화실패
[2101] AXT_RT_AIO_INVALID_MODULE_NO :유효하지않은 AIO 모듈
* See error code Table for more information on status error codes

Description

지정한 입력 모듈에 설정한 A/D 변환 방식을 확인하는 함수입니다. A/D 변환 방식은 사용자가 원하는 시점에서 Sampling을 하여 A/D변환하는 Normal Trigger방식과 H/W 내부 Timer를 이용해서 일정한 주기로 A/D변환하는Timer Trigger방식과 외부 신호가 입력될 때 A/D변환하는 External Trigger방식이 있습니다. 내부 Timer를 이용하는 Timer Trigger 방식 사용 시 반드시 AxaiHwSetSampleFreq 또는 AxaiHwSetSamplePeriod 함수를 이용하여 Sampling Rate를 설정하여야 합니다. A/D 변환 방식에 따른 Trigger Mode 설정은 아래 Table과 같습니다

Trigger Mode 설정

Enum	Value	Explanation
NORMAL_MODE	1	사용자가 원하는 시점에 A/D 변환하는 Software Trigger 방식
TIMER_MODE	2	H/W의 내부 클럭을 이용해서 A/D 변환하는 Trigger 방식
EXTERNAL_MODE	3	외부 입력단자의 클럭을 이용해서 A/D 변환하는 Trigger 방식

Trigger Mode 관련 함수

Trigger Mode	주요 관련 함수
Normal Trigger Mode	AxaiSwReadVoltage, AxaiSwReadMultiVoltage
Timer Trigger Mode	(함수 그룹 : AxaiHw.., AxaiEvent..,AxaiInterrupt..) AxaiEventSetChannel AxaiHwSetSampleFreq, AxaiHwSetSamplePeriod AxaiHwStartMultiChannel, AxaiHwStartMultiFilter, AxaiHwStopMultiChannel, AxaiHwSetLimit, AxaiHwReadDataLength AxaiHwReadSampleVoltage AxaiHwSetMultiAccess(ImmediateMode), AxaiHwStartMultiAccess(Immediate Mode) Etc..
External Trigger Mode	Timer Trigger Mode와 동일하나 Immediate Mode 함수와 AxaiHwSetSampleFreq / AxaiHwSetSamplePeriod 함수는 해당 사항 없음.

Example

C++

```
// 0번입력모듈에트리거모드를확인합니다.
DWORD dwTriggerMode;

AxaiGetTriggerMode(0, &dwTriggerMode);
```

Visual Basic

```
' 0번입력채널에트리거모드를확인합니다.
Dim lTriggerMode As Long

AxaiGetTriggerMode 0, lTriggerMode
```

Delphi

```
{ 0번입력채널에트리거모드를확인합니다. }
var
  dwTriggerMode: DWord;

begin
  AxaiGetTriggerMode(0, @dwTriggerMode);
  Application.MessageBox (PCHAR(IntToStr(dwTriggerMode)), 'Ajinextek', MB_OK);
end;
```

C#

```
// 0번입력모듈에트리거모드를확인합니다.
uint uTriggerMode;

CAXA.AxaiGetTriggerMode(0, ref uTriggerMode);
```

See Also

AxaiSetRange, AxaiGetRange, AxaiSetMultiRange, AxaiHwSetBufferOverflowMode, AxaiHwGetBufferOverflowMode, AxaiHwSetMultiBufferOverflowMode, AxaiHwSetLimit, AxaiHwGetLimit, AxaiHwSetMultiLimit, AxaiSetTriggerMode, AxaiHwSetSampleFreq, AxaiHwGetSampleFreq, AxaiHwSetSamplePeriod, AxaiHwGetSamplePeriod, AxaiSetModuleOffsetValue, AxaiGetModuleOffsetValue

AxaiHwSetSampleFreq

Purpose

지정한 입력 모듈에 샘플링 주기 단위를 주파수로 설정합니다.

Format

C++

```
DWORD AxaiHwSetSampleFreq(long lModuleNo, double dSampleFreq);
```

Visual Basic

```
Function AxaiHwSetSampleFreq(ByVal lModuleNo As Long, ByVal
dSampleFreq As Double) As Long
```

Delphi

```
functionAxaiHwSetSampleFreq(lModuleNo : LongInt; dSampleFreq :
Double) : DWord; stdcall;
```

C#

```
uint AxaiHwSetSampleFreq(int lModuleNo, double dSampleFreq);
```

Parameters

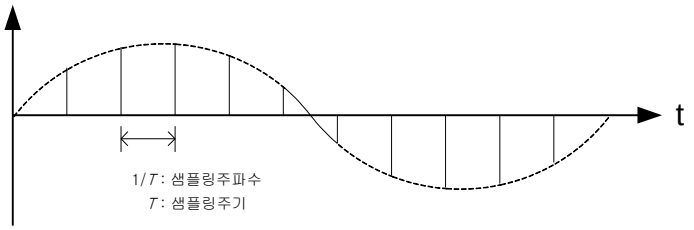
[in/out] Name	[Init Value] Explanation
[in]lModuleNo	입력 모듈 번호
[in]dSampleFreq	샘플링 주파수(단위 : Hz) / 설정범위 (10 ~ 100,000)

Return Values

[0000] AXT_RT_SUCCESS :함수실행성공
[1053] AXT_RT_NOT_OPEN :라이브러리초기화실패
[2101] AXT_RT_AIO_INVALID_MODULE_NO :유효하지않은 AIO 모듈
* See error code Table for more information on status error codes

Description

H/W 타이머(Timer Trigger Mode, External Trigger Mode)를 이용한 A/D 변환 시 사용됩니다. Timer Trigger Mode 방식 사용 시 반드시 샘플링 주기를 설정해야 합니다. PCI-AI16HR은 16개의 입력 채널이 모두 공통의 A/DConverter를 사용하므로 해당 함수의 설정으로 16개 채널 모두 샘플링됩니다. 이때 하나의 채널만 샘플링 할 경우 최대 100Khz의 주파수(0.01msec 주기)로 샘플링이 가능하며 16개의 채널을 동시에 샘플링 할 때 100000/16 = 6,250Hz보다 작게 설정하여야 합니다. 만약 하드웨어적인 최대 샘플링 한계치를 벗어난 값을 입력할 때 주파수는 한계치로 자동 조정되어 동작이 수행됩니다. 즉, 이 함수에서 설정한 dSampleFreq 값은한 개 채널의 샘플링 주파수값이 됩니다. 반면 하드웨어 특성이 다른 SIO-AI4RB는 4채널 A/DConverter를 사용하므로 선택된 채널의 개수로 말미암은 주파수의 한계치를 정하지 않습니다.



아날로그 데이터를 디지털 데이터로 변환할 때는 일정 간격으로 데이터를 변환해야 합니다. 이 변환시간을 샘플링 주기 T 라 하면 1/T(시간)를 샘플링 주파수라 합니다. 이 값은 샘플링 할 주파수나 주기를 의미합니다. 주파수의 단위는 Hz 입니다.

Example

C++

```
// 0번입력모듈에샘플링간격을 1000Hz로설정합니다.
AxaiHwSetSampleFreq(0, 1000);
```

Visual Basic

```
'0번입력모듈에샘플링간격을 1000Hz로설정합니다.
AxaiHwSetSampleFreq0, 1000
```

Delphi

```
{ 0번입력모듈에샘플링간격을 1000Hz로설정합니다. }
AxaiHwSetSampleFreq(0, 1000);
```

C#

```
// 0번입력모듈에샘플링간격을 1000Hz로설정합니다.
CAXA.AxaiHwSetSampleFreq(0, 1000);
```

See Also

AxaiSetRange, AxaiGetRange, AxaiSetMultiRange, AxaiHwSetBufferOverflowMode, AxaiHwGetBufferOverflowMode, AxaiHwSetMultiBufferOverflowMode, AxaiHwSetLimit, AxaiHwGetLimit, AxaiHwSetMultiLimit, AxaiSetTriggerMode, AxaiGetTriggerMode, AxaiHwGetSampleFreq, AxaiHwSetSamplePeriod, AxaiHwGetSamplePeriod, AxaiSetModuleOffsetValue, AxaiGetModuleOffsetValue

AxaiHwGetSampleFreq

Purpose

지정한 입력 모듈에 설정한 샘플링 주파수를 확인합니다.

Format

C++

```
DWORD AxaiHwGetSampleFreq(long lModuleNo, double *dpSampleFreq);
```

Visual Basic

```
Function AxaiHwGetSampleFreq(ByVal lModuleNo As Long, ByRef
dpSampleFreq As Double) As Long
```

Delphi

```
functionAxaiHwGetSampleFreq(lModuleNo : LongInt; dpSampleFreq :
PDouble) : DWord; stdcall;
```

C#

```
uint AxaiHwGetSampleFreq(int lModuleNo, ref double dpSampleFreq);
```

Parameters

[in/out] Name	[Init Value] Explanation
[in]lModuleNo	입력 모듈 번호
[out]dpSampleFreq	샘플링 주파수(단위 : Hz) / 설정범위 (10 ~ 100,000)

Return Values

[0000] AXT_RT_SUCCESS :함수실행성공
[1053] AXT_RT_NOT_OPEN :라이브러리초기화실패
[2101] AXT_RT_AIO_INVALID_MODULE_NO :유효하지않은 AIO 모듈
* See error code Table for more information on status error codes

Description

지정한 입력 모듈에 설정한 샘플링 주파수를 확인합니다. 샘플링 주파수의 단위는 Hz 입니다.

Example

C++

```
// 0번입력모듈에샘플링간격을주파수단위로설정된값을확인합니다.
double  dSampleFreq;

AxaiHwGetSampleFreq(0, &dSampleFreq);
```

Visual Basic

```
' 0번입력모듈에샘플링간격을주파수단위로설정된값을확인합니다.
Dim dSampleFreq As Double
```

```
AxaiHwGetSampleFreq 0, dSampleFreq
```

Delphi

```
{ 0번입력모듈에샘플링간격을주파수단위로설정된값을확인합니다. }
var
  dSampleFreq: Double;

begin
  AxaiHwGetSampleFreq(0, @dSampleFreq);
  Application.MessageBox (PCHAR(FloatToStr(dSampleFreq)), 'Ajinextek', MB_OK);
end;
```

C#

```
// 0번입력모듈에샘플링간격을주파수단위로설정된값을확인합니다.
double  dSampleFreq;

CAXA.AxaiHwGetSampleFreq(0, ref dSampleFreq);
```

See Also

AxaiSetRange, AxaiGetRange, AxaiSetMultiRange, AxaiHwSetBufferOverflowMode, AxaiHwGetBufferOverflowMode, AxaiHwSetMultiBufferOverflowMode, AxaiHwSetLimit, AxaiHwGetLimit, AxaiHwSetMultiLimit, AxaiSetTriggerMode, AxaiGetTriggerMode, AxaiHwSetSampleFreq, AxaiHwSetSamplePeriod, AxaiHwGetSamplePeriod, AxaiSetModuleOffsetValue, AxaiGetModuleOffsetValue

AxaiHwSetSamplePeriod

Purpose

지정한 모듈에 샘플링 주기 단위를 시간(uSec)으로 설정합니다.

Format

C++

```
DWORD AxaiHwSetSamplePeriod(long lModuleNo, double dSamplePeriod);
```

Visual Basic

```
Function AxaiHwSetSamplePeriod(ByVal lModuleNo As Long, ByVal
dSamplePeriod As Double) As Long
```

Delphi

```
functionAxaiHwSetSamplePeriod(lModuleNo : LongInt; dSamplePeriod :
Double) : DWord; stdcall;
```

C#

```
uint AxaiHwSetSamplePeriod(int lModuleNo, double dSamplePeriod);
```

Parameters

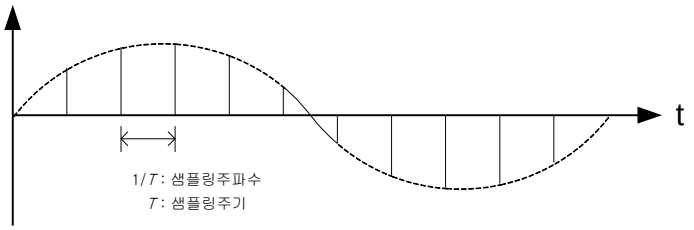
[in/out] Name	[Init Value] Explanation
[in]lModuleNo	입력 모듈 번호
[in]dSamplePeriod	샘플링 시간(단위 : uSec) / 설정범위 (10 ~ 100,000)

Return Values

[0000] AXT_RT_SUCCESS :함수실행성공
[1053] AXT_RT_NOT_OPEN :라이브러리초기화실패
[2101] AXT_RT_AIO_INVALID_MODULE_NO :유효하지않은 AIO 모듈
* See error code Table for more information on status error codes

Description

H/W 타이머(Timer Trigger Mode, External Trigger Mode)를 이용한 A/D 변환 시 사용됩니다. Timer Trigger Mode 방식 사용 시 반드시 샘플링 주기를 설정해야 합니다. PCI-AI16HR은 16개의 입력 채널이 모두 공통의 A/DConverter를 사용하므로 해당 함수의 설정으로 16개 채널 모두 샘플링됩니다. 이때 하나의 채널만 샘플링 할 때 최대 0.01msec 주기(100Khz의 주파수)로 샘플링이 가능하며 16개의 채널을 동시에 샘플링 할 때(1,000,000 * (100K/16 = 6250Hz)) = 160uSec보다 크게 설정하여야 합니다. 만약 하드웨어적인 최대 샘플링 한계치를 벗어난 값을 입력할 때 주파수는 한계치로 자동 조정되어 동작이 수행됩니다. 즉, 이 함수에서 설정한 dSampleFreq 값은한 개 채널의 샘플링 주파수값이 됩니다. 반면 하드웨어 특성이 다른 SIO-AI4RB는 4채널 A/DConverter를 사용하므로 선택된 채널의 개수로 말미암은 주파수의 한계치를 정하지 않습니다.



아날로그 데이터를 디지털 데이터로 변환할 때는 일정 간격으로 데이터를 변환해야 합니다. 이 변환시간을 샘플링 주기 T 라 하면 1/T(시간)를 샘플링 주파수라 합니다. 이 값은 샘플링 할 주파수나 주기를 의미합니다. 시간 단위는 uSec 입니다.

Example

C++

```
// 0번입력모듈에샘플링간격을100mSec로설정합니다.
AxaiHwSetSamplePeriod(0, 100);
```

Visual Basic

```
'0번입력모듈에샘플링간격을 100mSec로설정합니다.
AxaiHwSetSamplePeriod0, 100
```

Delphi

```
{ 0번입력모듈에샘플링간격을 100mSec로설정합니다. }
AxaiHwSetSamplePeriod(0, 100);
```

C#

```
// 0번입력모듈에샘플링간격을100mSec로설정합니다.
CAXA.AxaiHwSetSamplePeriod(0, 100);
```

See Also

AxaiSetRange, AxaiGetRange, AxaiSetMultiRange, AxaiHwSetBufferOverflowMode, AxaiHwGetBufferOverflowMode, AxaiHwSetMultiBufferOverflowMode, AxaiHwSetLimit, AxaiHwGetLimit, AxaiHwSetMultiLimit, AxaiSetTriggerMode, AxaiGetTriggerMode, AxaiHwGetSampleFreq, AxaiHwGetSamplePeriod,AxaiSetModuleOffsetValue, AxaiGetModuleOffsetValue

AxaiHwGetSamplePeriod

Purpose

지정한 모듈에 샘플링 시간(uSec)을 확인합니다.

Format

C++

```
DWORD AxaiHwGetSamplePeriod(long lModuleNo, double *dpSamplePeriod);
```

Visual Basic

```
Function AxaiHwGetSamplePeriod(ByVal lModuleNo As Long, ByRef
dpSamplePeriod As Double) As Long
```

Delphi

```
functionAxaiHwGetSamplePeriod(lModuleNo : LongInt; dpSamplePeriod :
PDouble) : DWord; stdcall;
```

C#

```
uint AxaiHwGetSamplePeriod(int lModuleNo, ref double dpSamplePeriod);
```

Parameters

[in/out] Name	[Init Value] Explanation
[in]lModuleNo	입력 모듈 번호
[out]dpSamplePeriod	샘플링 시간(단위 : uSec) / 설정범위 (10 ~ 100,000)

Return Values

[0000] AXT_RT_SUCCESS :함수실행성공
[1053] AXT_RT_NOT_OPEN :라이브러리초기화실패
[2101] AXT_RT_AIO_INVALID_MODULE_NO :유효하지않은 AIO 모듈
* See error code Table for more information on status error codes

Description

지정한 입력 모듈에 설정한 샘플링 시간을 확인합니다. 샘플링 시간의 단위는 uSec 입니다.

Example

C++

```
// 0번입력모듈에샘플링간격을시간단위로설정된값을확인합니다.
double  dSamplePeriod;

AxaiHwGetSamplePeriod(0, &dSamplePeriod);
```

Visual Basic

```
' 0번입력모듈에샘플링간격을시간단위로설정된값을확인합니다.
Dim dSamplePeriod As Double
```

```
AxaiHwGetSamplePeriod 0, dSamplePeriod
```

Delphi

```
{ 0번입력모듈에샘플링간격을시간단위로설정된값을확인합니다. }
var
  dSamplePeriod: Double;

begin
  AxaiHwGetSamplePeriod(0, @dSamplePeriod);
  Application.MessageBox (PCHAR(FloatToStr(dSamplePeriod)), 'Ajinextek', MB_OK);
end;
```

C#

```
// 0번입력모듈에샘플링간격을시간단위로설정된값을확인합니다.
double  dSamplePeriod;

CAXA.AxaiHwGetSamplePeriod(0, ref dSamplePeriod);
```

See Also

AxaiSetRange, AxaiGetRange, AxaiSetMultiRange, AxaiHwSetBufferOverflowMode, AxaiHwGetBufferOverflowMode, AxaiHwSetMultiBufferOverflowMode, AxaiHwSetLimit, AxaiHwGetLimit, AxaiHwSetMultiLimit, AxaiSetTriggerMode, AxaiGetTriggerMode, AxaiHwSetSampleFreq, AxaiHwGetSampleFreq, AxaiHwSetSamplePeriod, , AxaiSetModuleOffsetValue, AxaiGetModuleOffsetValue

AxaiSetModuleOffsetValue

Purpose

지정한 입력모듈의 Offset 을 mV 단위로 설정합니다.

Format

C++

```
DWORD AxaiSetModuleOffsetValue(long lModuleNo, double dMiliVolt);
```

Visual Basic

```
Function AxaiSetModuleOffsetValue(ByVal lModuleNo As Long, ByVal
dMiliVolt As Double) As Long
```

Delphi

```
functionAxaiSetModuleOffsetValue(lModuleNo : LongInt; dMiliVolt :
Double) : DWord; stdcall;
```

C#

```
uint AxaiSetModuleOffsetValue(int lModuleNo, double dMiliVolt);
```

Parameters

[in/out] Name	[Init Value] Explanation
[in]lModuleNo	입력 모듈 번호
[in]dMiliVolt	[0]Offset 값(단위 : mV) (-100 ~ 100)

Return Values

[0000] AXT_RT_SUCCESS :함수실행성공
[1053] AXT_RT_NOT_OPEN :라이브러리초기화실패
[1054]AXT_RT_NOT_SUPPORT_VERSION :지원하지않는하드웨어
[2101] AXT_RT_AIO_INVALID_MODULE_NO :유효하지않은 AIO 모듈
* See error code Table for more information on status error codes

Description

하드웨어적인 조정의 한계로 미세한 오차가 발생하게 되는데 이를 보정하기 위해 Calibrator(정확한 정전압 또는 정전류를 출력하여 계측기를 보정할 때 사용되는 정전원 공급기)로부터 0V를 입력받을 때 발생하는 오차값을 보상하기 위하여 설정하는 값이다. 즉, Calibrator를 통하여 정확한 0V를 입력하는경우, 측정된 입력값이 0.010V가 나온다면 이를 보정하기 위해 dMilivolt 값을 -10으로 설정하면 된다. 이후 모든 측정값은 Offset값 만큼 조절된 값이 측정된다.설정 단위는 mVolt 이며 최대 조절 가능한 범위는 -100mV~100mV 이다.

Example

C++

```
// 0번입력모듈에 Offset 을 -10mV로조절합니다.

AxaiSetModuleOffsetValue(0, -10);
```

Visual Basic

```
' 0번입력모듈에 Offset 을 -10mV로조절합니다.

AxaiSetModuleOffsetValue 0, -10
```

Delphi

```
{ 0번입력모듈에 Offset 을 -10mV로조절합니다.}

begin
  AxaiSetModuleOffsetValue(0, -10);
end;
```

C#

```
// 0번입력모듈에 Offset 을 -10mV로조절합니다.

CAXA.AxaiSetModuleOffsetValue(0, -10);
```

See Also

AxaiSetRange, AxaiGetRange, AxaiSetMultiRange, AxaiHwSetBufferOverflowMode, AxaiHwGetBufferOverflowMode, AxaiHwSetMultiBufferOverflowMode, AxaiHwSetLimit, AxaiHwGetLimit, AxaiHwSetMultiLimit, AxaiSetTriggerMode, AxaiGetTriggerMode, AxaiHwSetSampleFreq, AxaiHwGetSampleFreq, AxaiHwSetSamplePeriod, AxaiHwGetSamplePeriod,AxaiGetModuleOffsetValue

AxaiGetModuleOffsetValue

Purpose

지정한 입력모듈에 설정한 Offset 을 확인합니다.

Format

C++

```
DWORD AxaiGetModuleOffsetValue(long lModuleNo, double *dpMiliVolt);
```

Visual Basic

```
Function AxaiGetModuleOffsetValue(ByVal lModuleNo As Long, ByRef
dpMiliVolt As Double) As Long
```

Delphi

```
functionAxaiGetModuleOffsetValue(lModuleNo : LongInt; dpMiliVolt :
PDouble) : DWord; stdcall;
```

C#

```
uint AxaiGetModuleOffsetValue(int lModuleNo, ref double dpMiliVolt);
```

Parameters

[in/out] Name	[Init Value] Explanation
[in]lModuleNo	입력 모듈 번호
[out]dpMiliVolt	[0]Offset 값(단위 : mV) (-100 ~ 100)

Return Values

[0000] AXT_RT_SUCCESS :함수실행성공
[1053] AXT_RT_NOT_OPEN :라이브러리초기화실패
[1054]AXT_RT_NOT_SUPPORT_VERSION :지원하지않는하드웨어
[2101] AXT_RT_AIO_INVALID_MODULE_NO :유효하지않은 AIO 모듈
* See error code Table for more information on status error codes

Description

지정한 입력 모듈에 설정한 Offset 값을 확인합니다. 단위는 mV 입니다.

Example

C++

```
// 0번입력모듈에 Offset 을확인합니다.
double dMiliVolt;
AxaiGetModuleOffsetValue(0, &dMiliVolt);
```

Visual Basic

```
' 0번입력모듈에 Offset 을 -10mV로조절합니다.
Dim dMiliVolt As Double
AxaiGetModuleOffsetValue 0, dMiliVolt
```

Delphi

```
{ 0번입력모듈에 Offset 을 -10mV로조절합니다.}
var
  dMiliVolt: Double;

begin
  AxaiGetModuleOffsetValue(0, @dMiliVolt);
end;
```

C#

```
// 0번입력모듈에 Offset 을확인합니다.
double dMiliVolt;
CAXA.AxaiGetModuleOffsetValue(0, ref dMiliVolt);
```

See Also

AxaiSetRange, AxaiGetRange, AxaiSetMultiRange, AxaiHwSetBufferOverflowMode, AxaiHwGetBufferOverflowMode, AxaiHwSetMultiBufferOverflowMode, AxaiHwSetLimit, AxaiHwGetLimit, AxaiHwSetMultiLimit, AxaiSetTriggerMode, AxaiGetTriggerMode, AxaiHwSetSampleFreq, AxaiHwGetSampleFreq, AxaiHwSetSamplePeriod, AxaiHwGetSamplePeriod,AxaiSetModuleOffsetValue

입력모듈 A/D 변환

Function	Description
AxaiSwReadVoltage	지정한 입력 채널에 아날로그 입력 값을 전압 값으로 반환한다.
AxaiSwReadDigit	지정한 입력 채널에 아날로그 입력 값을 Digit 값으로 반환한다.
AxaiSwReadMultiVoltage	지정한 여러 입력 채널에 아날로그 입력 값을 전압 값으로 반환한다.
AxaiSwReadMultiDigit	지정한 여러 입력 채널에 아날로그 입력 값을 Digit 값으로 반환한다.
AxaiHwStartMultiChannel	지정한 여러 입력 채널에 H/W타이머를 이용한 A/D변환을 시작한다.
AxaiHwStartMultiFilter	지정한 여러 입력 채널에 A/D변환을 시작 후 지정한 개수만큼 필터 처리해서 전압으로 반환한다.
AxaiHwStopMultiChannel	H/W타이머를 이용한 A/D변환을 중지한다.
AxaiHwSetMultiAccess	지정한 여러 입력 채널에 Immediate모드를 사용하기 위해 설정 값을 설정한다.
AxaiHwStartMultiAccess	지정한 개수만큼 A/D변환 후 데이터를 반환한다.
AxaiHwReadSampleVoltage	지정한 입력 채널에 H/W타이머를 이용하여 A/D변환된 값을 전압 값으로 읽는다.
AxaiHwReadSampleDigit	지정한 입력 채널에 H/W타이머를 이용하여 A/D변환된 값을 Digit 값으로 읽는다.
AxaiHwStartSingleChannelAdc	지정한 입력 채널에 H/W타이머를 이용한 A/D변환을 시작한다
AxaiHwStopSingleChannelAdc	H/W타이머를 이용한 연속 신호 A/D변환을 중지한다

AxaiSwReadVoltage

Purpose

지정한 입력 채널에 아날로그 입력 값을 전압으로 반환합니다.

Format

C++

```
DWORD AxaiSwReadVoltage(long lAdcChannelNo, double *dpVolt);
```

Visual Basic

```
Function AxaiSwReadVoltage(ByVal lAdcChannelNo As Long, ByRef dpVolt
As Double) As Long
```

Delphi

```
functionAxaiSwReadVoltage(lAdcChannelNo : LongInt; dpVolt : PDouble) :
DWord; stdcall;
```

C#

```
uint AxaiSwReadVoltage(int lChannelNo, ref double dpVolt);
```

Parameters

[in/out] Name	[Init Value] Explanation
[in]lAdcChannelNo	입력 채널 번호
[out]dpVolt	입력 전압 값(단위 : Volt)

Return Values

[0000] AXT_RT_SUCCESS :함수실행성공
[1053] AXT_RT_NOT_OPEN :라이브러리초기화실패
[2102] AXT_RT_AIO_INVALID_CHANNEL_NO :유효하지않은 AIO 채널번호
* See error code Table for more information on status error codes

Description

지정한 입력 채널에 아날로그 입력 값을 A/D 변환한 후 전압 값으로 반환합니다. 이 함수를 사용하기 전에 AxaiSetTriggerMode 함수를 사용하여 Normal Trigger Mode 로 설정해야 합니다.

Example

C++

```
// 0번입력채널에아날로그입력값을전압값으로반환합니다.
double  dVolt;

AxaiSwReadVoltage(0, &dVolt);
```

Visual Basic

```
' 0번입력채널에아날로그입력값을전압값으로반환합니다.
Dim dVolt As Double
```

```
AxaiSwReadVoltage 0, dVolt
```

Delphi

```
{ 0번입력채널에아날로그입력값을전압값으로반환합니다. }
var
  dVolt : Double;

begin
  AxaiSwReadVoltage(0, @dVolt);
  Application.MessageBox (PCHAR(FloatToStr(dVolt)), 'Ajinextek', MB_OK);
end;
```

C#

```
// 0번입력채널에아날로그입력값을전압값으로반환합니다.
double  dVolt;

CAXA.AxaiSwReadVoltage(0, ref dVolt);
```

See Also

AxaiSwReadVoltage, AxaiSwReadDigit, AxaiSwReadMultiVoltage, AxaiSwReadMultiDigit, AxaiHwStartMultiFilter, AxaiHwStopMultiChannel, AxaiHwSetMultiAccess,AxaiHwStartMultiAccess, AxaiHwReadSampleVoltage, AxaiHwReadSampleDigit, AxaiHwStartSingleChannelAdc, AxaiHwStopSingleChannelAdc

AxaiSwReadDigit

Purpose

지정한 입력 채널에 아날로그 입력 값을 Digit 값으로 반환합니다.

Format

C++

```
DWORD AxaiSwReadDigit(long lAdcChannelNo, DWORD *upDigit);
```

Visual Basic

```
Function AxaiSwReadDigit(ByVal lAdcChannelNo As Long, ByRef upDigit
As Long) As Long
```

Delphi

```
functionAxaiSwReadDigit(lAdcChannelNo : LongInt; upDigit : PDWord) :
DWord; stdcall;
```

C#

```
uint AxaiSwReadDigit(int lChannelNo, ref uint upDigit);
```

Parameters

[in/out] Name	[Init Value] Explanation
[in]lAdcChannelNo	입력 채널 번호
[out]upDigit	Digit 값

Return Values

[0000] AXT_RT_SUCCESS :함수실행성공
[1053] AXT_RT_NOT_OPEN :라이브러리초기화실패
[2102] AXT_RT_AIO_INVALID_CHANNEL_NO :유효하지않은 AIO 채널번호
* See error code Table for more information on status error codes

Description

Normal Trigger Mode를 이용한 A/D 변환 시 사용됩니다. 지정한 입력 채널에 아날로그 입력 값을 변환한 후 Digit 값으로 반환합니다. 이 값의 범위는 하드웨어적인 분해능 특성에 따릅니다. Digit값이 최소 값일 때 전압 값은 AxaiSetRange에서 설정한 최소 값을 의미합니다. 이 함수 사용 전 AxaiSetTriggerMode 함수를 사용하여 Normal Trigger Mode로 설정하여야 합니다.

Digit 범위

Module	Resolution	Digit range
SIO-AI4RB	12Bit	0 ~ 4095
PCI-AI16HR	16But	-32767 ~ 32767

Example

C++

```
// 0번입력채널에아날로그입력값을Digit 값으로반환합니다.
DWORD dwDigit;

AxaiSwReadDigit(0, &dwDigit);
short retShort = (short) dwDigit;// 부호있는 2Byte 로형변환
double dVolt = (((double)retShort ) / 32767.0) * 10;
// PCI-AI16HR의경우 Digit범위가 -32767 ~ 32767이므로위와같이변환하면 Voltage 근사값을얻게
// 된다.
```

Visual Basic

```
'0번입력채널에아날로그입력값을Digit 값으로반환합니다.
Dim lDigit As Long

AxaiSwReadDigit 0, lDigit
```

Delphi

```
{ 0번입력채널에아날로그입력값을전압값으로반환합니다. }
var
  dwDigit: DWord;

begin
  AxaiSwReadDigit(0, @dwDigit);
  Application.MessageBox (PCHAR(IntToStr(dwDigit)), 'Ajinextek', MB_OK);
end;
```

C#

```
// 0번입력채널에아날로그입력값을Digit 값으로반환합니다.
uint uDigit;

CAXA.AxaiSwReadDigit(0, ref uDigit);
intnConvert = Convert.ToInt32(uDigit);
double dVolt = (Convert.ToDouble(nConvert) / 32768.0) * 10;
// PCI-AI16HR의경우 Digit범위가 -32767 ~ 32767이므로위와같이변환하면 Voltage 근사값을얻게
// 된다.
```

See Also

AxaiSwReadVoltage, AxaiSwReadDigit, AxaiSwReadMultiVoltage, AxaiSwReadMultiDigit, AxaiHwStartMultiFilter, AxaiHwStopMultiChannel, AxaiHwSetMultiAccess,AxaiHwStartMultiAccess, AxaiHwReadSampleVoltage, AxaiHwReadSampleDigit, AxaiHwStartSingleChannelAdc, AxaiHwStopSingleChannelAdc

AxaiSwReadMultiVoltage

Purpose

지정한 여러 입력 채널에 아날로그 입력 값을 전압 값으로 반환합니다.

Format

C++

```
DWORD AxaiSwReadMultiVoltage(long lSize, long *lpAdcChannelNo, double
*dpVolt);
```

Visual Basic

```
Function AxaiSwReadMultiVoltage(ByVal lSize As Long, ByRef
lpAdcChannelNo As Long, ByRef dpVolt As Double) As Long
```

Delphi

```
functionAxaiSwReadMultiVoltage(lSize : LongInt; lpAdcChannelNo :
PLongInt; dpVolt : PDouble) : DWord; stdcall;
```

C++

```
uint AxaiSwReadMultiVoltage(int lSize, ref int lpChannelNo, ref
double dpVolt);
```

Parameters

[in/out] Name	[Init Value] Explanation
[in]lSize	입력 채널의 개수
[in]lpAdcChannelNo	채널 번호 배열
[out]dpVolt	입력받을 전압값 배열(단위 : Volt)

Return Values

[0000] AXT_RT_SUCCESS :함수실행성공
[1053] AXT_RT_NOT_OPEN :라이브러리초기화실패
[2102] AXT_RT_AIO_INVALID_CHANNEL_NO :유효하지않은 AIO 채널번호
* See error code Table for more information on status error codes

Description

Normal Trigger Mode를 이용한 A/D 변환 시 사용됩니다. 지정한 여러 입력 채널에서아날로그 값을 변환한 후 전압 값으로 반환합니다. 이 함수를 사용하기 전에 AxaiSetTriggerMode 함수를 사용하여 Normal Trigger Mode로 설정하여야 합니다.

Example

C++

```
// 여러입력채널에아날로그입력값을전압값으로반환합니다.
long lAdcChannelNo[4] = {0, 1, 2, 3};
double dVolt[4];
```

```
AxaiSwReadMultiVoltage(4, lAdcChannelNo, dVolt);
```

Visual Basic

```
'여러입력채널에아날로그입력값을전압값으로반환합니다.
Dim lAdcChannelNo(0 To 3) As Long
Dim dVolt(0 To 3) As Double

lAdcChannelNo(0) = 0
lAdcChannelNo(1) = 1
lAdcChannelNo(2) = 2
lAdcChannelNo(3) = 3

AxaiSwReadMultiVoltage 4, lAdcChannelNo(0), dVolt(0)
```

Delphi

```
{ 여러입력채널에아날로그입력값을전압값으로반환합니다. }
var
  lAdcChannelNo: array [0..3] of LongInt;
  dVolt : array [0..3] of Double;

begin
  lAdcChannelNo[0]:= 0;
  lAdcChannelNo[1]:= 1;
  lAdcChannelNo[2]:= 2;
  lAdcChannelNo[3]:= 3;

  AxaiSwReadMultiVoltage(4, @lAdcChannelNo, @dVolt);
end;
```

C#

```
// 여러입력채널에아날로그입력값을전압값으로반환합니다.
uint[] nAdcChannelNo = new uint[4];
double[] dVolt = new double[4];

nAdcChannelNo[0] = 0;
nAdcChannelNo[1] = 1;
nAdcChannelNo[2] = 2;
nAdcChannelNo[3] = 3;

AxaiSwReadMultiVoltage(4, ref lAdcChannelNo[0], ref dVolt[0]);
```

See Also

AxaiSwReadVoltage, AxaiSwReadDigit, AxaiSwReadMultiVoltage, AxaiSwReadMultiDigit, AxaiHwStartMultiFilter, AxaiHwStopMultiChannel, AxaiHwSetMultiAccess,AxaiHwStartMultiAccess, AxaiHwReadSampleVoltage, AxaiHwReadSampleDigit, AxaiHwStartSingleChannelAdc, AxaiHwStopSingleChannelAdc

AxaiSwReadMultiDigit

Purpose

지정한 여러 입력 채널에 아날로그 입력 값을 Digit 값으로 반환합니다.

Format

C++

```
DWORD AxaiSwReadMultiDigit(long lSize, long *lpAdcChannelNo, DWORD
*upDigit);
```

Visual Basic

```
Function AxaiSwReadMultiDigit(ByVal lSize As Long, ByRef
lpAdcChannelNo As Long, ByRef upDigit As Long) As Long
```

Delphi

```
functionAxaiSwReadMultiDigit(lSize : LongInt; lpAdcChannelNo :
PLongInt; upDigit : PDWord) : DWord; stdcall;
```

C#

```
uint AxaiSwReadMultiDigit(int lSize, ref int lpChannelNo, ref uint
upDigit);
```

Parameters

[in/out] Name	[Init Value] Explanation
[in]lSize	입력 채널의 개수
[in]lpAdcChannelNo	채널 번호 배열
[out]upDigit	입력 받을 Digit 값 배열

Return Values

[0000] AXT_RT_SUCCESS :함수실행성공
[1053] AXT_RT_NOT_OPEN :라이브러리초기화실패
[2102] AXT_RT_AIO_INVALID_CHANNEL_NO :유효하지않은 AIO 채널번호
* See error code Table for more information on status error codes

Description

Normal Trigger Mode를 이용한 A/D 변환 시 사용됩니다. 지정한 여러 입력 채널에 아날로그 입력 값을 변환한 후 Digit 값으로 반환합니다. 이 값의 범위는 하드웨어적인 분해능 특성에 따릅니다. Digit값이 최소 값일 때 전압 값은 AxaiSetRange에서 설정한 최소 값을 의미합니다. 이 함수 사용 전 AxaiSetTriggerMode 함수를 사용하여 Normal Trigger Mode로 설정하여야 합니다.

Digit 범위

Module	Resolution	Digit range
SIO-AI4RB	12Bit	0 ~ 4095
PCI-AI16HR	16Bit	-32767 ~ 32767

Example

C++

```
// 여러입력채널에아날로그입력값을Digit 값으로반환합니다.
long lAdcChannelNo[4] = {0, 1, 2, 3};
DWORD dwDigit[4];

AxaiSwReadMultiDigit(4, lAdcChannelNo, dwDigit);
short retShort = (short) dwDigit[0];// 부호있는 2Byte 로형변환
double dVoltFirst = (((double)retShort ) / 32767.0) * 10;
// PCI-AI16HR의경우 Digit범위가 -32767 ~ 32767이므로위와같이변환하면 Voltage 근사값을얻게
// 된다.
```

Visual Basic

```
'여러입력채널에아날로그입력값을 Digit 값으로반환합니다.
Dim lAdcChannelNo(0 To 3) As Long
Dim lDigit(0 To 3) As Long

lAdcChannelNo(0) = 0
lAdcChannelNo(1) = 1
lAdcChannelNo(2) = 2
lAdcChannelNo(3) = 3

AxaiSwReadMultiDigit 4, lAdcChannelNo(0), dwDigit(0)
```

Delphi

```
{ 여러입력채널에아날로그입력값을 Digit 값으로반환합니다. }
var
  lAdcChannelNo: array [0..3] of LongInt;
  dwDigit : array [0..3] of DWord;

begin
  lAdcChannelNo[0]:= 0;
  lAdcChannelNo[1]:= 1;
  lAdcChannelNo[2]:= 2;
  lAdcChannelNo[3]:= 3;

  AxaiSwReadMultiDigit(4, @lAdcChannelNo, @dwDigit);
end;
```

C#

```
// 여러입력채널에아날로그입력값을Digit 값으로반환합니다.
int[] nAdcChannelNo = new int[4];
uint[] uDigit = new uint[4];

nAdcChannelNo[0] = 0;
nAdcChannelNo[1] = 1;
nAdcChannelNo[2] = 2;
nAdcChannelNo[3] = 3;

CAXA.AxaiSwReadMultiDigit(4, ref lAdcChannelNo[0], uDigit);
```

```
int retShort = Convert.ToInt32(uDigit[0]);
double dVoltFirst = ((Convert.ToDouble(retShort) / 32768.0) * 10;
```

See Also

AxaiSwReadVoltage, AxaiSwReadDigit, AxaiSwReadMultiVoltage, AxaiSwReadMultiDigit, AxaiHwStartMultiFilter, AxaiHwStopMultiChannel, AxaiHwSetMultiAccess,AxaiHwStartMultiAccess, AxaiHwReadSampleVoltage, AxaiHwReadSampleDigit, AxaiHwStartSingleChannelAdc, AxaiHwStopSingleChannelAdc

AxaiHwStartMultiChannel

Purpose

지정한 여러 입력 채널에 H/W 타이머를 이용한 A/D 변환을 시작합니다.

Format

C++

```
DWORD AxaiHwStartMultiChannel(long lSize, long *lpAdcChannelNo, long
lBuffSize);
```

Visual Basic

```
Function AxaiHwStartMultiChannel(ByVal lSize As Long, ByRef
lpAdcChannelNo As Long, ByVal lBuffSize As Long) As Long
```

Delphi

```
functionAxaiHwStartMultiChannel(lSize : LongInt; lpAdcChannelNo :
PLongInt; lBuffSize : LongInt) : DWord; stdcall;
```

C#

```
uint AxaiHwStartMultiChannel(int lSize, ref int lpChannelNo, int
lBuffSize);
```

Parameters

[in/out] Name	[Init Value] Explanation
[in]lSize	입력 채널의 개수
[in]lpAdcChannelNo	채널 번호의 배열
[in]lBuffSize	[4096]각 채널에 할당되는 버퍼의 크기로 AxaiHwSetLimit의 상한값보다 큰 값이어야 합니다.

Return Values

[0000] AXT_RT_SUCCESS :함수실행성공
[1053] AXT_RT_NOT_OPEN :라이브러리초기화실패
[2102] AXT_RT_AIO_INVALID_CHANNEL_NO :유효하지않은 AIO 채널번호
* See error code Table for more information on status error codes

Description

지정한 여러 입력 채널에 H/W 타이머(Timer Trigger Mode, External Trigger Mode)를 이용한 A/D 변환을 시작합니다. H/W 타이머를 이용해서 AxaiHwStopMultiChannel 함수가 호출될 때까지 연속적으로 아날로그 입력값을 A/D 변환하여 버퍼에 저장합니다. 버퍼에 저장된 A/D 변환 값은 AxaiHwReadSampleVoltage 나 AxaiHwReadSampleDigit 함수를 이용해서 확인할 수 있습니다. 내부적으로 A/D 변환 데이터가 저장되는 버퍼는 환형 버퍼 형식으로 운용됩니다. 환형 버퍼는 한정된 버퍼에 무한히 데이터를 기록하기 위해 사용되는 것으로써, 데이터가 버퍼의 마지막 위치까지 다 채워지면 버퍼의 처음 위치로부터 다시 채워 나가는 방식을 말합니다. 이 함수 사용 전 연속적 데이터 수집을위한 관련 설정이 필요합니다.

※관련 함수

AxaiSetTriggerMode,AxaiInterruptSetModuleMask, AxaiEventSetChannel, AxaiHwSetLimit, AxaiEventSetChannelEnable, AxaiEventSetChannelMask , AxaiHwSetSampleFreq

Example

C++

```
// 여러입력채널에 H/W타이머를이용한 A/D변환을시작합니다.
long lAdcChannelNo[4] = {0, 1, 2, 3};

AxaiHwStartMultiChannel(4, lAdcChannelNo, 1000);
```

Visual Basic

```
'여러입력채널에 H/W타이머를이용한 A/D변환을시작합니다.
Dim lAdcChannelNo(0 To 3) As Long
Dim dVolt(0 To 3) As Double

lAdcChannelNo(0) = 0
lAdcChannelNo(1) = 1
lAdcChannelNo(2) = 2
lAdcChannelNo(3) = 3

AxaiHwStartMultiChannel 4, lAdcChannelNo(0), 1000
```

Delphi

```
{ 여러입력채널에 H/W타이머를이용한 A/D변환을시작합니다. }
var
  lAdcChannelNo: array [0..3] of LongInt;

begin
  lAdcChannelNo[0]:= 0;
  lAdcChannelNo[1]:= 1;
  lAdcChannelNo[2]:= 2;
  lAdcChannelNo[3]:= 3;

  AxaiHwStartMultiChannel(4, @lAdcChannelNo, 1000);
end;
```

C#

```
// 여러입력채널에 H/W타이머를이용한 A/D변환을시작합니다.
int[] nAdcChannelNo = new int[4];

nAdcChannelNo[0] = 0;
nAdcChannelNo[1] = 1;
nAdcChannelNo[2] = 2;
nAdcChannelNo[3] = 3;

CAXA.AxaiHwStartMultiChannel(4, ref lAdcChannelNo[0], 1000);
```

See Also

AxaiSwReadVoltage, AxaiSwReadDigit, AxaiSwReadMultiVoltage, AxaiSwReadMultiDigit, AxaiHwStartMultiFilter, AxaiHwStopMultiChannel, AxaiHwSetMultiAccess,AxaiHwStartMultiAccess, AxaiHwReadSampleVoltage, AxaiHwReadSampleDigit, AxaiHwStartSingleChannelAdc, AxaiHwStopSingleChannelAdc

AxaiHwStartMultiFilter

Purpose

지정한 여러 입력 채널에 A/D 변환 시작 후 지정한 개수만큼의 평균값을버퍼에 저장합니다.

Format

C++

```
DWORD AxaiHwStartMultiFilter(long lSize, long *lpAdcChannelNo, long
lFilterCount, long lBuffSize);
```

Visual Basic

```
Function AxaiHwStartMultiFilter(ByVal lSize As Long, ByRef
lpAdcChannelNo As Long, ByVal lFilterCount As Long, ByVal lBuffSize
As Long) As Long
```

Delphi

```
functionAxaiHwStartMultiFilter(lSize : LongInt; lpAdcChannelNo :
PLongInt; lFilterCount : LongInt; lBuffSize : LongInt) : DWord;
stdcall;
```

C#

```
uint AxaiHwStartMultiFilter(int lSize, ref int lpChannelNo, int
lFilterCount, int lBuffSize);
```

Parameters

[in/out] Name	[Init Value] Explanation
[in]lSize	입력 채널의 개수
[in]lpAdcChannelNo	채널 번호의 배열
[in]lFilterCount	[0]Filtering할 데이터의 개수
[in]lBuffSize	[4096]각 채널에 할당되는 버퍼의 사이즈로 AxaiHwSetLimit의 상한값보다 큰 값이어야 합니다.

Return Values

[0000] AXT_RT_SUCCESS :함수실행성공
[1053] AXT_RT_NOT_OPEN :라이브러리초기화실패
[2102] AXT_RT_AIO_INVALID_CHANNEL_NO :유효하지않은 AIO 채널번호
* See error code Table for more information on status error codes

Description

H/W 타이머(Timer Trigger Mode, External Trigger Mode)를 이용한 연속적 A/D 변환 시 사용됩니다. 연속적으로 수집된 데이터의 신뢰성 향상을 위해서 lFilterCount 개수만큼의 데이터 평균값을 계산하여 버퍼에 저장합니다. 예로 lFilterCount 의 값이 10 일 때 설정된 샘플링 주기로 10 개의 데이터 수집 후 데이터의 평균값을 계산하여 버퍼에 저장하므로 버퍼에 저장된 데이터의 개수는 1 개가 됩니다.

Digit 범위

Module	Resolution	Digit range
SIO-AI4RB	12Bit	0 ~ 4095
PCI-AI16HR	16Bit	-32767 ~ 32767

Example

C++

```
// 여러입력채널에 A/D변환을시작후지정한개수만큼필터처리해서전압으로반환합니다.
long lAdcChannelNo[4] = {0, 1, 2, 3};

AxaiHwStartMultiFilter(4, lAdcChannelNo, 500, 512);
```

Visual Basic

```
'여러입력채널에 A/D변환을시작후지정한개수만큼필터처리해서전압으로반환합니다.
Dim lAdcChannelNo(0 To 3) As Long

lAdcChannelNo(0) = 0
lAdcChannelNo(1) = 1
lAdcChannelNo(2) = 2
lAdcChannelNo(3) = 3

AxaiHwStartMultiFilter 4, lAdcChannelNo(0), 500, 512
```

Delphi

```
{ 여러입력채널에 A/D변환을시작후지정한개수만큼필터처리해서전압으로반환합니다. }
var
  lAdcChannelNo: array [0..3] of LongInt;

begin
  lAdcChannelNo[0]:= 0;
  lAdcChannelNo[1]:= 1;
  lAdcChannelNo[2]:= 2;
  lAdcChannelNo[3]:= 3;

  AxaiHwStartMultiFilter(4, @lAdcChannelNo, 500, 512);
end;
```

C#

```
// 여러입력채널에 A/D변환을시작후지정한개수만큼필터처리해서전압으로반환합니다.
int[] nAdcChannelNo = new int[4];

nAdcChannelNo[0] = 0;
nAdcChannelNo[1] = 1;
nAdcChannelNo[2] = 2;
nAdcChannelNo[3] = 3;

CAXA.AxaiHwStartMultiFilter(4, ref nAdcChannelNo[0], 500, 512);
```

See Also

AxaiSwReadVoltage, AxaiSwReadDigit, AxaiSwReadMultiVoltage, AxaiSwReadMultiDigit, AxaiHwStartMultiFilter, AxaiHwStopMultiChannel, AxaiHwSetMultiAccess,AxaiHwStartMultiAccess, AxaiHwReadSampleVoltage, AxaiHwReadSampleDigit, AxaiHwStartSingleChannelAdc, AxaiHwStopSingleChannelAdc

AxaiHwStopMultiChannel

Purpose

지정한 여러 입력 채널에 H/W 타이머를 이용한 연속 A/D 변환을 중지 합니다.

Format

C++

```
DWORD AxaiHwStopMultiChannel(long lModuleNo);
```

Visual Basic

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

Delphi

```
functionAxaiHwStopMultiChannel(lModuleNo : LongInt) : DWord; stdcall;
```

C#

```
uint AxaiHwStopMultiChannel(int lModuleNo);
```

Parameters

[in/out] Name	[Init Value] Explanation
[in]lModuleNo	입력 모듈 번호

Return Values

[0000] AXT_RT_SUCCESS :함수실행성공
[1053] AXT_RT_NOT_OPEN :라이브러리초기화실패
[2101] AXT_RT_AIO_INVALID_MODULE_NO :유효하지않은 AIO 모듈
* See error code Table for more information on status error codes

Description

H/W 타이머(Timer Trigger Mode, External Trigger Mode)를 이용한 연속적 A/D 변환을 중지합니다. lModuleNo를 -1로 설정할 때 시스템에 장착된 아날로그 입력 모듈중 AxaiHwStartMultiChannel 또는 AxaiHwStartMultiFilter를 사용하여 연속적 데이터 수집을 하고 있는 모든 모듈이 중지됩니다.

Example

C++

```
// H/W타이머를이용한 A/D변환을중지합니다.
AxaiHwStopMultiChannel(-1);
```

Visual Basic

```
' H/W타이머를이용한 A/D변환을중지합니다.
AxaiHwStopMultiChannel -1
```

Delphi

```
{ H/W타이머를이용한 A/D변환을중지합니다. }
AxaiHwStopMultiChannel(-1);
```

C#

```
// H/W타이머를이용한 A/D변환을중지합니다.
CAXA.AxaiHwStopMultiChannel(-1);
```

See Also

AxaiSwReadVoltage, AxaiSwReadDigit, AxaiSwReadMultiVoltage, AxaiSwReadMultiDigit, AxaiHwStartMultiFilter, AxaiHwStopMultiChannel, AxaiHwSetMultiAccess,AxaiHwStartMultiAccess, AxaiHwReadSampleVoltage, AxaiHwReadSampleDigit, AxaiHwStartSingleChannelAdc, AxaiHwStopSingleChannelAdc

AxaiHwSetMultiAccess

Purpose

지정한 여러 입력 채널에 Immediate 모드를 사용하기 위한 설정을 합니다.

Format

C++

```
DWORD AxaiHwSetMultiAccess(long lSize, long *lpAdcChannelNo, long
*lpWordSize);
```

Visual Basic

```
Function AxaiHwSetMultiAccess(ByVal lSize As Long, ByRef
lpAdcChannelNo As Long, ByRef lpWordSize As Long) As Long
```

Delphi

```
functionAxaiHwSetMultiAccess(lSize : LongInt; lpAdcChannelNo :
PLongInt; lpWordSize : PLongInt) : DWord; stdcall;
```

C#

```
uint AxaiHwSetMultiAccess(int lSize, ref int lpChannelNo, ref int
lpWordSize);
```

Parameters

[in/out] Name	[Init Value] Explanation
[in]lSize	입력 채널의 개수
[in]lpAdcChannelNo	채널 번호의 배열
[in]lpWordSize	채널 당 데이터의 개수

Return Values

[0000] AXT_RT_SUCCESS :함수실행성공
[1053] AXT_RT_NOT_OPEN :라이브러리초기화실패
[2101] AXT_RT_AIO_INVALID_MODULE_NO :유효하지않은 AIO 모듈
* See error code Table for more information on status error codes

Description

H/W 타이머(Timer Trigger Mode, External Trigger Mode)를 이용한 연속적 A/D 변환 시 사용됩니다. Immediate 모드는 AxaiHwStartMultiAccess 함수 호출 시점부터 A/D 변환되어 H/W 버퍼에 저장되어 있는 데이터를 *lpWordSize 배열 값의 개수만큼 읽어오는 방식입니다. *lpWordSize는 PCI-AI16HR는 1024개의 데이터를 저장할 수 있으며, SIO-AI4RB는 4096개를 저장할 수 있습니다. 여러 채널을 사용할 경우 한 채널당 저장할 수 있는 최대 데이터 개수 = 최대 데이터 개수 / 입력 채널 수가 됩니다. 사용자가 이 값보다 큰 값을 지정하게 되면 자동으로 제한되게 됩니다.이 함수 사용전 AxaiSetTriggerMode 함수를 사용하여 Timer Trigger Mode로 설정하여야 합니다.

H/W 버퍼 사이즈

Module	Buffer Size

SIO-AI4RB	4096
PCI-AI16HR	1024

Example

C++

```
// 여러입력채널에 Immediate모드를사용하기위해설정값을설정합니다.
long lAdcChannelNo[4] = {0, 1, 2, 3};
long lWordSize[4] = {1024, 1024, 1024, 1024};

AxaiSetTriggerMode(0, TIMER_MODE);
AxaiHwSetMultiAccess(4, lAdcChannelNo, lWordSize);
```

Visual Basic

```
'여러입력채널에 Immediate모드를사용하기위해설정값을설정합니다.
Dim lAdcChannelNo(0 To 3) As Long
Dim lWordSize(0 To 3) As Long

lAdcChannelNo(0) = 0
lAdcChannelNo(1) = 1
lAdcChannelNo(2) = 2
lAdcChannelNo(3) = 3

lWordSize(0) = 1024
lWordSize(1) = 1024
lWordSize(2) = 1024
lWordSize(3) = 1024

AxaiSetTriggerMode 0, TIMER_MODE
AxaiHwSetMultiAccess 4, lAdcChannelNo(0), lWordSize(0)
```

Delphi

```
{ 여러입력채널에 Immediate모드를사용하기위해설정값을설정합니다. }
var
  lAdcChannelNo : array [0..3] of LongInt;
  lWordSize : array [0..3] of LongInt;

begin
  lAdcChannelNo[0] := 0;
  lAdcChannelNo[1] := 1;
  lAdcChannelNo[2] := 2;
  lAdcChannelNo[3] := 3;

  lWordSize[0] := 1024;
  lWordSize[1] := 1024;
  lWordSize[2] := 1024;
  lWordSize[3] := 1024;

  AxaiSetTriggerMode(0, TIMER_MODE);
```

```
  AxaiHwSetMultiAccess(4, @lAdcChannelNo, @lWordSize);
end;
```

C#

```
// 여러입력채널에 Immediate모드를사용하기위해설정값을설정합니다.
int[]nAdcChannelNo = new int[4];
int[]nWordSize = new int[4];

nAdcChannelNo[0] = 0;
nAdcChannelNo[1] = 1;
nAdcChannelNo[2] = 2;
nAdcChannelNo[3] = 3;

nWordSize[0] = 1024;
nWordSize[1] = 1024;
nWordSize[2] = 1024;
nWordSize[3] = 1024;

CAXA.AxaiSetTriggerMode(0,(uint)AXT_AIO_TRIGGER_MODE.TIMER_MODE);
CAXA.AxaiHwSetMultiAccess(4, ref nAdcChannelNo[0], ref nWordSize[0]);
```

See Also

AxaiSwReadVoltage, AxaiSwReadDigit, AxaiSwReadMultiVoltage, AxaiSwReadMultiDigit, AxaiHwStartMultiFilter, AxaiHwStopMultiChannel, AxaiHwSetMultiAccess,AxaiHwStartMultiAccess, AxaiHwReadSampleVoltage, AxaiHwReadSampleDigit, AxaiHwStartSingleChannelAdc, AxaiHwStopSingleChannelAdc

AxaiHwStartMultiAccess

Purpose

Immediate 모드로 설정되어 수집된 데이터를 확인합니다.

Format

C++

```
DWORD AxaiHwStartMultiAccess(double *dpBuffer[]);
```

Visual Basic

```
Function AxaiHwStartMultiAccess(ByRef dpBuffer() As Double) As Long
```

Delphi

```
functionAxaiHwStartMultiAccess(dpBuffer : PDouble) : DWord; stdcall;
```

C#

```
uintAxaiHwStartMultiAccess(ref double dpBuffer);
```

Parameters

[in/out] Name	[Init Value] Explanation
[out]dpBuffer[]	A/D변환 전압 값 (2차원 배열)

Return Values

[0000] AXT_RT_SUCCESS :함수실행성공
[1053] AXT_RT_NOT_OPEN :라이브러리초기화실패
* See error code Table for more information on status error codes

Description

Immediate 모드는 함수 호출 시점부터 A/D 변환되어 H/W 버퍼에 저장되어 있는 데이터를 *lpWordSize 배열 값의 개수만큼 읽어오는 방식입니다. *lpWordSize는 PCI-AI16HR는 1024개의 데이터를 저장할 수 있으며, SIO-AI4RB는 4096개를 저장할 수 있습니다. 여러 채널을 사용할 경우 한 채널당 저장할 수 있는 최대 데이터 개수 = 최대 데이터 개수 / 입력 채널 수가 됩니다. 사용자가 이 값보다 큰 값을 지정하게 되면 자동으로 제한되게 됩니다.이 함수 사용전 AxaiSetTriggerMode 함수를 사용하여 Timer Trigger Mode로 설정하고 AxaiHwSetMultiAccess 함수로 입력받은 채널 번호와 데이터 개수를 설정하여야 합니다.

Sample

C++

```
// 지정한개수만큼 A/D변환후데이터를반환합니다.
double  pdCBuffer[4][1024];
double  *pdBuffer[4];

long lAdcChannelNo[4] = {0, 1, 2, 3};
long lWordSize[4] = {1024, 1024, 1024, 1024};
long lCount;
```

```
for(lCount = 0; lCount < 4; lCount++)
{
  // 버퍼할당
  pdBuffer[lCount]  = pdCBuffer[lCount];
}

AxaiSetTriggerMode(0, TIMER_MODE);
AxaiHwSetMultiAccess(4, lAdcChannelNo, lWordSize);
AxaiHwStartMultiAccess(pdBuffer);
```

Visual Basic

```
'지정한개수만큼 A/D변환후데이터를반환합니다.
Dim dpCBuffer(0 To 3, 0 To 1023) As Double
Dim dpBuffer(0 To 3) As Double
Dim lAdcChannelNo(0 To 3) As Long
Dim lWordSize(0 To 3) As Long

dpBuffer(0) = dpCBuffer(0, 0)
dpBuffer(1) = dpCBuffer(1, 0)
dpBuffer(2) = dpCBuffer(2, 0)
dpBuffer(3) = dpCBuffer(3, 0)

lAdcChannelNo(0) = 0
lAdcChannelNo(1) = 1
lAdcChannelNo(2) = 2
lAdcChannelNo(3) = 3

lWordSize(0) = 1024
lWordSize(1) = 1024
lWordSize(2) = 1024
lWordSize(3) = 1024

AxaiSetTriggerMode 0, TIMER_MODE
AxaiHwSetMultiAccess 4, lAdcChannelNo(0), lWordSize(0)
AxaiHwStartMultiAccess(dpBuffer)
```

Delphi

```
{ 지정한개수만큼 A/D변환후데이터를반환합니다. }
var
  dpCBuffer : array [0..3, 0..1023] of Double;
  dpBuffer : array[0..3] of Pointer;
   lAdcChannelNo : array [0..3] of LongInt;
  lWordSize : array [0..3] of LongInt;

begin
  dpBuffer[0] := @dpCBuffer[0,0];
  dpBuffer[1] := @dpCBuffer[1,0];
  dpBuffer[2] := @dpCBuffer[2,0];
  dpBuffer[3] := @dpCBuffer[3,0];
```

```
 lAdcChannelNo[0] := 0;
lAdcChannelNo[1] := 1;
lAdcChannelNo[2] := 2;
lAdcChannelNo[3] := 3;

lWordSize[0] := 1024;
lWordSize[1] := 1024;
lWordSize[2] := 1024;
lWordSize[3] := 1024;

AxaiHwStartMultiAccess(@dpBuffer);
AxaiSetTriggerMode(0, TIMER_MODE);
AxaiHwSetMultiAccess(4, @lAdcChannelNo, @lWordSize);
```

C#

```
// 지정한개수만큼 A/D변환후데이터를반환합니다.
double[,] pdCBuffer = new double[4,1024];
double[] pdBuffer = new double[4];

int[]nAdcChannelNo = new int[4];
int[]nWordSize = new int[4];
int  nCount;

nAdcChannelNo[0] = 0;
nAdcChannelNo[1] = 1;
nAdcChannelNo[2] = 2;
nAdcChannelNo[3] = 3;

nWordSize[0] = 1024;
nWordSize[1] = 1024;
nWordSize[2] = 1024;
nWordSize[3] = 1024;

for(nCount = 0; nCount < 4; nCount++)
{
  pdBuffer[nCount]  = pdCBuffer[nCount,0];  // 버퍼할당
}

CAXA.AxaiSetTriggerMode(0,(uint)AXT_AIO_TRIGGER_MODE.TIMER_MODE);
CAXA.AxaiHwSetMultiAccess(4, ref nAdcChannelNo[0], ref nWordSize[0]);
CAXA.AxaiHwStartMultiAccess(ref pdBuffer[0]);
```

See Also

AxaiSwReadVoltage, AxaiSwReadDigit, AxaiSwReadMultiVoltage, AxaiSwReadMultiDigit, AxaiHwStartMultiFilter, AxaiHwStopMultiChannel, AxaiHwSetMultiAccess,AxaiHwStartMultiAccess, AxaiHwReadSampleVoltage, AxaiHwReadSampleDigit, AxaiHwStartSingleChannelAdc, AxaiHwStopSingleChannelAdc

AxaiHwReadSampleVoltage

Purpose

지정한 입력 채널에 H/W 타이머를 이용하여 A/D 변환된 데이터를 전압 값으로 확인합니다.

Format

C++

```
DWORD AxaiHwReadSampleVoltage(long lAdcChannelNo, long *lpCount,
double *dpVolt);
```

Visual Basic

```
Function AxaiHwReadSampleVoltage(ByVal lAdcChannelNo As Long, ByRef
lpCount As Long, ByRef dpVolt As Double) As Long
```

Delphi

```
functionAxaiHwReadSampleVoltage(lAdcChannelNo : LongInt; lpCount :
PLongInt; dpVolt : PDouble) : DWord; stdcall;
```

C#

```
uint AxaiHwReadSampleVoltage(int lChannelNo, ref int lpSize, ref
double dpVolt);
```

Parameters

[in/out] Name	[Init Value] Explanation
[in]lAdcChannelNo	입력 채널 번호 (0 ~ 입력 채널 개수 -1)
[in/out]lpCount	In : dpVolt 배열의 크기 지정 및 확인하고자 하는 전압값의 개수를 지정 Out : 수집된 전압값의 유효 개수
[out]dpVolt	수집된 전압 값의 배열

Return Values

[0000] AXT_RT_SUCCESS :함수실행성공
[1053] AXT_RT_NOT_OPEN :라이브러리초기화실패
[2102] AXT_RT_AIO_INVALID_CHANNEL_NO :유효하지않은 AIO 채널번호
* See error code Table for more information on status error codes

Description

H/W 타이머(Timer Trigger Mode, External Trigger Mode)를 이용해서 A/D 변환되어 버퍼에 저장된 데이터를 lpCount 개수만큼 전압 값으로 읽는다. 이때, lpCount는 양 방향 파라메터로 lpCount 개수만큼 전압값을 dpVolt 배열에 저장하며 저장한 데이터 개수를 반환합니다.
이 함수사용전AxaiSetTriggerMode 함수를 사용하여 Timer Trigger Mode 또는 External Trigger Mode로 설정하여야 합니다.

Example

C++

```
// 0번입력채널에 H/W타이머를이용하여 A/D변환된값을전압값으로읽는다.
double  dpVolt[1024];
long lCount = 0;

AxaiSetTriggerMode(0, TIMER_MODE)
AxaiHwReadDataLength(0, &lCount); // 내부버퍼에수집된데이터개수얻어오기
if(lCount > 1024)
  lCount = 1024;
AxaiHwReadSampleVoltage(0, &lCount, dpVolt);
// lpCount에는실제읽어들인 count 반환됨
```

Visual Basic

```
' 0번입력채널에 H/W타이머를이용하여 A/D변환된값을전압값으로읽는다.
Dim dVolt(0 To 1024) As Double
Dim lCount As Long
lCount = 1024

AxaiSetTriggerMode 0, TIMER_MODE
AxaiHwReadSampleVoltage 0, lCount, dVolt(0)
' lCount에는실제읽어들인 count 반환됨
```

Delphi

```
{ 0번입력채널에H/W타이머를이용하여 A/D변환된값을전압값으로읽는다. }
var
  lCount : LongInt;
  dVolt : array [0..1024] of Double;

begin
  lCount = 1024;

  AxaiSetTriggerMode(0, TIMER_MODE)
  AxaiHwReadSampleVoltage(0, @lCount, @dVolt);
end;
{ lCount에는실제읽어들인 count 반환됨}
```

C#

```
// 0번입력채널에 H/W타이머를이용하여 A/D변환된값을전압값으로읽는다.
double[] dpVolt = new double[1024];
long lCount = 0;

AxaiHwReadDataLength(0, ref lCount); // 내부버퍼에수집된데이터개수얻어오기

if(lCount > 1024)
  lCount = 1024;

CAXA.AxaiSetTriggerMode(0,(uint)AXT_AXIO_TRIGGER_MODE.TIMER_MODE);
CAXA.AxaiHwReadSampleVoltage(0, ref lCount, ref dpVolt[0]);
// lpCount에는실제읽어들인 count 반환됨
```

See Also

AxaiSwReadVoltage, AxaiSwReadDigit, AxaiSwReadMultiVoltage, AxaiSwReadMultiDigit, AxaiHwStartMultiFilter, AxaiHwStopMultiChannel, AxaiHwSetMultiAccess,AxaiHwStartMultiAccess, AxaiHwReadSampleVoltage, AxaiHwReadSampleDigit, AxaiHwStartSingleChannelAdc, AxaiHwStopSingleChannelAdc

AxaiHwReadSampleDigit

Purpose

지정한 입력 채널에 H/W 타이머를 이용하여 A/D 변환된 데이터를 Digit 값으로 확인합니다.

Format

C++

```
DWORD AxaiHwReadSampleDigit(long lAdcChannelNo, long *lpCount, DWORD
*upDigit);
```

Visual Basic

```
Function AxaiHwReadSampleDigit(ByVal lAdcChannelNo As Long, ByRef
lpCount As Long, ByRef upDigit As Long) As Long
```

Delphi

```
functionAxaiHwReadSampleDigit(lAdcChannelNo : LongInt; lpCount :
PLongInt; upDigit : PDWord) : DWord; stdcall;
```

C#

```
uint AxaiHwReadSampleDigit(int lChannelNo, ref int lpSize, ref uint
upDigit);
```

Parameters

[in/out] Name	[Init Value] Explanation
[in]lAdcChannelNo	입력 채널 번호 (0 ~ 입력 채널 개수 -1)
[in/out]lpCount	In : dpVolt 배열의 크기 지정 및 확인하고자 하는 전압값의 개수를 지정 Out : 수집된 Digit 값의 유효 개수
[out]upDigit	수집된 Digit 값의 배열

Return Values

[0000] AXT_RT_SUCCESS :함수실행성공
[1053] AXT_RT_NOT_OPEN :라이브러리초기화실패
[2102] AXT_RT_AIO_INVALID_CHANNEL_NO :유효하지않은 AIO 채널번호
* See error code Table for more information on status error codes

Description

H/W 타이머(Timer Trigger Mode, External Trigger Mode)를 이용해서 A/D 변환되어 버퍼에 저장된 데이터를 lpCount 개수만큼 전압값으로 읽는다. 이때, lpCount는 양 방향 파라메터로 lpCount 개수만큼 전압값을 dpVolt 배열에 저장하며 저장한 데이터 개수를 반환합니다.
이 함수 사용전 AxaiSetTriggerMode 함수를 사용하여 Timer Trigger Mode 또는 External Trigger Mode로 설정 하여야 합니다.

Digit 범위

Module	Resolution	Digit range
SIO-AI4RB	12Bit	0 ~ 4095

PCI-AI16HR	16Bit	-32767 ~ 32767

Example

C++

```
// 0번입력채널에 H/W타이머를이용하여 A/D변환된값을 Digit 값으로읽는다.
DWORD upDigit[1024];
long lCount = 0;
AxaiSetTriggerMode(0, TIMER_MODE)
AxaiHwReadDataLength(0, &lCount);
if(lCount > 1024)
  lCount = 1024;
AxaiHwReadSampleDigit(0, &lCount, upDigit);
short retShort = (short) upDigit[0];          // 부호있는 2Byte 로형변환
double dVoltFirst = (((double)retShort ) / 32768.0) * 10;
//SIO-AI16HB의경우 Digit값이 Two's complement 이므로위와같이변환하면 Voltage 근사값을얻게
// 된다.
```

Visual Basic

```
'0번입력채널에 H/W타이머를이용하여 A/D변환된값을 Digit 값으로읽는다.
Dim lDigit(0 To 1024) As Long
Dim lCount As Long
lCount = 1024

AxaiSetTriggerMode 0, TIMER_MODE
AxaiHwReadSampleDigit 0, lCount, lDigit
```

Delphi

```
{ 0번입력채널에 H/W타이머를이용하여 A/D변환된값을 Digit 값으로읽는다. }
Var
  lCount : LongInt;
  dwDigit: array [0..1024] of DWord;

begin
  lCount = 1024;
  AxaiSetTriggerMode(0, TIMER_MODE);
  AxaiHwReadSampleDigit(0, @lCount, @dwDigit);
end;
```

C#

```
// 0번입력채널에 H/W타이머를이용하여 A/D변환된값을 Digit 값으로읽는다.
uint[] upDigit = new uint[1024];
int nCount = 0;
CAXA.AxaiSetTriggerMode(0, (uint)AXT_AIO_TRIGGER_MODE.TIMER_MODE);
CAXA.AxaiHwReadDataLength(0, ref nCount);
if(nCount > 1024)
  nCount = 1024;
CAXA.AxaiHwReadSampleDigit(0, ref nCount, ref upDigit[0]);
Int16nRetShort = Convert.ToInt16(upDigit[0]);// 부호있는 2Byte 로형변환
```

```
double dVoltFirst = (Convert.ToDouble(nRetShort) / 32768.0) * 10;
// PCI-AI16HR경우 Digit값이 Two's complement 이므로위와같이변환하면 Voltage 근사값을얻게된다.
```

See Also

AxaiSwReadVoltage, AxaiSwReadDigit, AxaiSwReadMultiVoltage, AxaiSwReadMultiDigit, AxaiHwStartMultiFilter, AxaiHwStopMultiChannel, AxaiHwSetMultiAccess,AxaiHwStartMultiAccess, AxaiHwReadSampleVoltage, AxaiHwReadSampleDigit, AxaiHwStartSingleChannelAdc, AxaiHwStopSingleChannelAdc

AxaiHwStartSingleChannelAdc

Purpose

지정한 입력 채널에 H/W 타이머를 이용한 A/D 변환을 시작합니다.

Format

C++

```
DWORD AxaiHwStartSingleChannelAdc (long lAdcChannelNo, long
lBuffSize);
```

Visual Basic

```
Function AxaiHwStartSingleChannelAdc (ByVal lAdcChannelNo As Long,
ByVal lBuffSize As Long) As Long
```

Delphi

```
Function AxaiHwStartSingleChannelAdc (lAdcChannelNo : LongInt;
lBuffSize : LongInt) : DWord; stdcall;
```

C#

```
uint AxaiHwStartSingleChannelAdc (int lChannelNo, int lBuffSize);
```

Parameters

[in/out] Name	[Init Value] Explanation
[in]lAdcChannelNo	입력 채널 번호 (0 ~ 입력 채널 개수 -1)
[in] lBuffSize	[4096]각 채널에 할당되는 버퍼의 크기로 AxaiHwSetLimit의 상한값보다 큰 값이어야 합니다.

Return Values

[0000] AXT_RT_SUCCESS :함수실행성공
[1053] AXT_RT_NOT_OPEN :라이브러리초기화실패
[2102] AXT_RT_AIO_INVALID_CHANNEL_NO :유효하지않은 AIO 채널번호
* See error code Table for more information on status error codes

Description

지정한 입력 채널에 H/W타이머(Timer Trigger Mode, External Trigger Mode)를 이용한 A/D변환을 시작합니다. H/W타이머를 이용해서 AxaiHwStopSingleChannelAdc 함수가 호출될 때까지 연속적으로 아날로그 입력값을 A/D 변환하여 버퍼에 저장합니다. 버퍼에 저장된 A/D 변환 값은 AxaiHwReadSampleVoltage나 AxaiHwReadSampleDigit 함수를 이용해서 확인할 수 있습니다. 내부적으로 A/D변환 데이터가 저장되는 버퍼는 환형 버퍼 형식으로 운용됩니다. 환형 버퍼는 한정된 버퍼에 무한히 데이터를 기록하기 위해 사용되는 것으로써, 데이터가 버퍼의 마지막 위치까지 다 채워지면 버퍼의 처음 위치로부터 다시 채워 나가는 방식을 말합니다. 이 함수 사용 전 연속적 데이터 수집을위한 관련 설정이 필요 합니다.

Example

C++

```
// 지정 입력채널에 H/W타이머를 이용한 A/D 변환을 시작 합니다.
AxaiHwStartSingleChannelAdc (4, 1000);
```

Visual Basic

```
'지정 입력채널에 H/W타이머를 이용한 A/D 변환을 시작 합니다.
Dim dVolt(0 To 3) As Double

AxaiHwStartSingleChannelAdc 4, 1000
```

Delphi

```
{ 지정 입력채널에 H/W타이머를 이용한 A/D 변환을 시작 합니다.}
begin
  AxaiHwStartSingleChannelAdc (4, 1000);
end;
```

C#

```
// 지정 입력채널에 H/W타이머를 이용한 A/D 변환을 시작 합니다.
CAXA. AxaiHwStartSingleChannelAdc (4, 1000);
```

See Also

AxaiSwReadVoltage, AxaiSwReadDigit, AxaiSwReadMultiVoltage, AxaiSwReadMultiDigit, AxaiHwStartMultiFilter, AxaiHwStopMultiChannel, AxaiHwSetMultiAccess,AxaiHwStartMultiAccess, AxaiHwReadSampleVoltage, AxaiHwReadSampleDigit, AxaiHwStartSingleChannelAdc, AxaiHwStopSingleChannelAdc

AxaiHwStopSingleChannelAdc

Purpose

지정한 여러 입력 채널에 H/W 타이머를 이용한 연속 A/D 변환을 중지 합니다.

Format

C++

```
DWORD AxaiHwStopSingleChannelAdc (long lModuleNo);
```

Visual Basic

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

Delphi

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

C#

```
uint AxaiHwStopSingleChannelAdc (int lModuleNo);
```

Parameters

[in/out] Name	[Init Value] Explanation
[in]lModuleNo	입력 모듈 번호

Return Values

[0000] AXT_RT_SUCCESS :함수실행성공
[1053] AXT_RT_NOT_OPEN :라이브러리초기화실패
[2101] AXT_RT_AIO_INVALID_MODULE_NO :유효하지않은 AIO 모듈
* See error code Table for more information on status error codes

Description

H/W 타이머(Timer Trigger Mode, External Trigger Mode)를 이용한 연속적 A/D 변환을 중지합니다. lModuleNo를 -1로 설정할 때 시스템에 장착된 아날로그 입력 모듈중 AxaiHwStartMultiChannel 또는 AxaiHwStartMultiFilter, AxaiHwStartSingleChannelAdc를 사용하여 연속적 데이터 수집을 하고 있는 모든 모듈이 중지됩니다.

Example

C++

```
// H/W타이머를이용한 A/D변환을중지합니다.
AxaiHwStopSingleChannelAdc (-1);
```

Visual Basic

```
' H/W타이머를이용한 A/D변환을중지합니다.
AxaiHwStopSingleChannelAdc -1
```

Delphi

```
{ H/W타이머를이용한 A/D변환을중지합니다. }
AxaiHwStopSingleChannelAdc (-1);
```

C#

```
// H/W타이머를이용한 A/D변환을중지합니다.
CAXA. AxaiHwStopSingleChannelAdc (-1);
```

See Also

AxaiSwReadVoltage, AxaiSwReadDigit, AxaiSwReadMultiVoltage, AxaiSwReadMultiDigit, AxaiHwStartMultiFilter, AxaiHwStopMultiChannel, AxaiHwSetMultiAccess,AxaiHwStartMultiAccess, AxaiHwReadSampleVoltage, AxaiHwReadSampleDigit, AxaiHwStartSingleChannelAdc, AxaiHwStopSingleChannelAdc

입력모듈버퍼상태체크

Function	Description
AxaiHwIsBufferEmpty	지정한 입력 채널의 메모리 버퍼에 데이터가 없는 지 검사한다.
AxaiHwIsBufferUpper	지정한 입력 채널의 메모리 버퍼에 설정되어 있는 상한 값보다 데이터가 많은 지 검사한다.
AxaiHwIsBufferLower	지정한 입력 채널의 메모리 버퍼에 설정되어 있는 하한 값보다 데이터가 적은 지 검사한다.
AxaiHwReadDataLength	지정한 입력 채널의 메모리 버퍼에 데이터가 몇 개인지 검사한다.

AxaiHwIsBufferEmpty

Purpose

지정한 입력 채널의 버퍼에 데이터 존재 유무를 확인합니다.

Format

C++

```
DWORD AxaiHwIsBufferEmpty(long lAdcChannelNo, DWORD *upEmpty);
```

Visual Basic

```
Function AxaiHwIsBufferEmpty(ByVal lAdcChannelNo As Long, ByRef
upEmpty As Long) As Long
```

Delphi

```
functionAxaiHwIsBufferEmpty(lAdcChannelNo : LongInt; upEmpty:
PDWord) : DWord; stdcall;
```

C#

```
uintAxaiHwIsBufferEmpty(int lAdcChannelNo, ref uint upEmpty);
```

Parameters

[in/out] Name	[Init Value] Explanation
[in]lAdcChannelNo	입력 채널 번호
[out]upEmpty	- [00h] 버퍼에 데이터가 있을 때 - [01h] 버퍼에 데이터가 없을 때

Return Values

[0000] AXT_RT_SUCCESS :함수실행성공
[1053] AXT_RT_NOT_OPEN :라이브러리초기화실패
[2102] AXT_RT_AIO_INVALID_CHANNEL_NO :유효하지않은 AIO 채널번호
* See error code Table for more information on status error codes

Description

H/W 타이머(Timer Trigger Mode, External Trigger Mode)를 이용한 연속적 A/D 변환 시 사용됩니다. 각 채널에 할당되어 있는 버퍼에 데이터 존재 유무를 확인하여 데이터가 없다면 'TRUE'를 반환하고 데이터가 한 개라도 있다면 'FALSE'를 반환합니다. 이 함수 사용 전 AxaiSetTriggerMode 함수를 사용하여 Timer Trigger Mode 또는 External Trigger Mode 로 설정되어 있어야 합니다.

Example

C++

```
// 0번입력채널의메모리버퍼에데이터가없는지검사합니다.
DWORD dwEmpty;
AxaiHwIsBufferEmpty(0, &dwEmpty);

if (dwEmpty == TRUE)
  AfxMessageBox("0번입력채널의메모리버퍼에데이터가없습니다.");
```

```
else
  AfxMessageBox("0번입력채널의메모리버퍼에데이터가있습니다.");
```

Visual Basic

```
' 0번입력채널의메모리버퍼에데이터가없는지검사합니다.
Dim lEmpty As Long

AxaiHwIsBufferEmpty 0, dwEmpty

If dwEmpty = TRUE Then
  MsgBox "0번입력채널의메모리버퍼에데이터가없습니다."
Else
  MsgBox "0번입력채널의메모리버퍼에데이터가있습니다."
End If
```

Delphi

```
{ 0번입력채널의메모리버퍼에데이터가없는지검사합니다. }
var
  dwEmpty : DWord;

begin
  AxaiHwIsBufferEmpty(0, @dwEmpty);

  if (dwEmpty = 1) then
    Application.MessageBox('0번입력채널의메모리버퍼에데이터가없습니다.', 'Ajinextek', MB_OK)
  else
    Application.MessageBox('0 번입력채널의메모리버퍼에데이터가있습니다.', 'Ajinextek', MB_OK);
end;
```

C#

```
// 0번입력채널의메모리버퍼에데이터가없는지검사합니다.
uint uEmpty;
CAXA.AxaiHwIsBufferEmpty(0, ref uEmpty);

if (uEmpty == TRUE)
  MessageBox.Show("0번입력채널의메모리버퍼에데이터가없습니다.");
else
  MessageBox.Show("0번입력채널의메모리버퍼에데이터가있습니다.");
```

See Also

AxaiHwIsBufferUpper, AxaiHwIsBufferLower, AxaiHwReadDataLength

AxaiHwIsBufferUpper

Purpose

지정한 입력 채널의 버퍼에 설정되어 있는 상한 값보다 데이터가 많은지 확인합니다.

Format

C++

```
DWORD AxaiHwIsBufferUpper(long lAdcChannelNo, DWORD *upUpper);
```

Visual Basic

```
Function AxaiHwIsBufferUpper(ByVal lAdcChannelNo As Long, ByRef
upUpper As Long) As Long
```

Delphi

```
functionAxaiHwIsBufferUpper(lAdcChannelNo : LongInt; upUpper:
PDWord) : DWord; stdcall;
```

C#

```
uintAxaiHwIsBufferUpper(intnAdcChannelNo, ref uintupUpper);
```

Parameters

[in/out] Name	[Init Value] Explanation
[in]lAdcChannelNo	입력 채널 번호
[out]upUpper	- [00h] 버퍼에 데이터가 상한 값보다 적을 때 - [01h] 버퍼에 데이터가 상한 값보다 많을 때

Return Values

[0000] AXT_RT_SUCCESS :함수실행성공
[1053] AXT_RT_NOT_OPEN :라이브러리초기화실패
[2102] AXT_RT_AIO_INVALID_CHANNEL_NO :유효하지않은 AIO 채널번호
* See error code Table for more information on status error codes

Description

H/W 타이머(Timer Trigger Mode, External Trigger Mode)를 이용한 연속적 A/D 변환 시 사용됩니다. 각 채널에 할당되어 있는 버퍼에 AxaiHwSetLimit 에서 설정한 상한 값보다 데이터가 많은를 확인하여 데이터가 많으면 'TRUE'를 반환하고 데이터가 상한 값보다 적으면 'FALSE'를 반환합니다. 이 함수 사용 전 AxaiSetTriggerMode 함수를 사용하여 Timer Trigger Mode 또는 External Trigger Mode 로 설정되어 있어야 합니다.

Example

C++

```
// 0번입력채널의메모리버퍼에설정되어있는상한값보다데이터가많은지검사합니다.
DWORD dwUpper;

AxaiHwIsBufferUpper(0, &dwUpper);
```

```
if (dwUpper == TRUE)
  AfxMessageBox("0번입력채널의메모리버퍼에상한값보다데이터가많습니다.");
else
  AfxMessageBox("0번입력채널의메모리버퍼에상한값보다데이터가적습니다.");
```

Visual Basic

```
' 0번입력채널의메모리버퍼에설정되어있는상한값보다데이터가많은지검사합니다.
Dim lUpper As Long

AxaiHwIsBufferUpper 0, lUpper

If lUpper = TRUE Then
  MsgBox "0번입력채널의메모리버퍼에상한값보다데이터가많습니다."
Else
  MsgBox "0번입력채널의메모리버퍼에상한값보다데이터가적습니다."
End If
```

Delphi

```
{ 0번입력채널의메모리버퍼에설정되어있는상한값보다데이터가많은지검사합니다. }
var
  dwUpper : DWord;

begin
  AxaiHwIsBufferUpper(0, @dwUpper);

  if (dwUpper = 1) then
    Application.MessageBox('0번입력채널의메모리버퍼에상한값보다데이터가많습니다.', 'Ajinextek',
      MB_OK)
  else
    Application.MessageBox('0 번입력채널의메모리버퍼에상한값보다데이터가적습니다.', 'Ajinextek',
      MB_OK);
end;
```

C#

```
// 0번입력채널의메모리버퍼에설정되어있는상한값보다데이터가많은지검사합니다.
uint uUpper;

CAXA.AxaiHwIsBufferUpper(0, ref uUpper);

if (uUpper == TRUE)
  MessageBox.Show("0번입력채널의메모리버퍼에상한값보다데이터가많습니다.");
else
  MessageBox.Show ("0번입력채널의메모리버퍼에상한값보다데이터가적습니다.");
```

See Also

AxaiHwIsBufferEmpty, AxaiHwIsBufferLower, AxaiHwReadDataLength

AxaiHwIsBufferLower

Purpose

지정한 입력 채널의 버퍼에 설정되어 있는 하한 값보다 데이터가 적은지 확인합니다.

Format

C++

```
DWORD AxaiHwIsBufferLower(long lAdcChannelNo, DWORD *upLower);
```

Visual Basic

```
Function AxaiHwIsBufferLower(ByVal lAdcChannelNo As Long, ByRef
upLower As Long) As Long
```

Delphi

```
functionAxaiHwIsBufferLower(lAdcChannelNo : LongInt; upLower:
PDWord) : DWord; stdcall;
```

C#

```
Uint AxaiHwIsBufferLower(int nAdcChannelNo, uint upLower);
```

Parameters

[in/out] Name	[Init Value] Explanation
[in]lAdcChannelNo	입력 채널 번호
[out]upLower	- [00h] 버퍼에 데이터가 하한 값보다 많을 때 - [01h] 버퍼에 데이터가 하한 값보다 적을 때

Return Values

[0000] AXT_RT_SUCCESS :함수실행성공
[1053] AXT_RT_NOT_OPEN :라이브러리초기화실패
[2102] AXT_RT_AIO_INVALID_CHANNEL_NO :유효하지않은 AIO 채널번호
* See error code Table for more information on status error codes

Description

H/W 타이머(Timer Trigger Mode, External Trigger Mode)를 이용한 연속적 A/D 변환 시 사용됩니다. 각 채널에 할당되어 있는 버퍼에 AxaiHwSetLimit 에서 설정한 하한 값보다 데이터가 적은지를 확인하여 데이터가 적으면 'TRUE'를 반환하고 데이터가 상한 값보다 많으면 'FALSE'를 반환합니다. 이 함수 사용 전 AxaiSetTriggerMode 함수를 사용하여 Timer Trigger Mode 또는 External Trigger Mode 로 설정되어 있어야 합니다.

Example

C++

```
// 0번입력채널의메모리버퍼에설정되어있는하한값보다데이터가적은지검사합니다.
DWORD dwLower;

AxaiHwIsBufferLower(0, &dwUpper);
```

```
if (dwUpper == TRUE)
  AfxMessageBox("0번입력채널의메모리버퍼에하한값보다데이터가적습니다.");
else
  AfxMessageBox("0번입력채널의메모리버퍼에하한값보다데이터가많습니다.");
```

Visual Basic

```
' 0번입력채널의메모리버퍼에설정되어있는하한값보다데이터가적은지검사합니다.
Dim lLower As Long

AxaiHwIsBufferLower 0, lLower

If lLower = TRUE Then
  MsgBox "0번입력채널의메모리버퍼에하한값보다데이터가적습니다."
Else
  MsgBox "0번입력채널의메모리버퍼에하한값보다데이터가많습니다."
End If
```

Delphi

```
{ 0번입력채널의메모리버퍼에설정되어있는하한값보다데이터가적은지검사합니다. }
var
  dwLower : DWord;

begin
  AxaiHwIsBufferLower(0, @dwLower);

  if (dwLower = 1) then
     Application.MessageBox('0번입력채널의메모리버퍼에하한값보다데이터가적습니다.', 'Ajinextek',
        MB_OK)
end;
```

C#

```
// 0번입력채널의메모리버퍼에설정되어있는하한값보다데이터가적은지검사합니다.
uint uLower;

CAXA.AxaiHwIsBufferLower(0, ref uUpper);

if (uUpper == TRUE)
  MessageBox.Show("0번입력채널의메모리버퍼에하한값보다데이터가적습니다.");
else
  MessageBox.Show("0번입력채널의메모리버퍼에하한값보다데이터가많습니다.");
```

See Also

AxaiHwIsBufferEmpty, AxaiHwIsBufferUpper, AxaiHwReadDataLength

AxaiHwReadDataLength

Purpose

지정한 입력 채널의 버퍼에 저장되어 있는 데이터 개수를 확인합니다.

Format

C++

```
DWORD AxaiHwReadDataLength(long lAdcChannelNo, long *lpDataLength);
```

Visual Basic

```
Function AxaiHwReadDataLength(ByVal lAdcChannelNo As Long, ByRef
lpDataLength As Long) As Long
```

Delphi

```
functionAxaiHwReadDataLength(lAdcChannelNo : LongInt; lpDataLength :
PLongInt) : DWord; stdcall;
```

C#

```
uintAxaiHwReadDataLength(intnAdcChannelNo, ref intnpDataLength);
```

Parameters

[in/out] Name	[Init Value] Explanation
[in]lAdcChannelNo	입력 채널 번호
[out]lpDataLength	버퍼에 저장되어 있는 데이터 개수

Return Values

[0000] AXT_RT_SUCCESS :함수실행성공
[1053] AXT_RT_NOT_OPEN :라이브러리초기화실패
[2101] AXT_RT_AIO_INVALID_MODULE_NO :유효하지않은 AIO 모듈
* See error code Table for more information on status error codes

Description

H/W 타이머(Timer Trigger Mode, External Trigger Mode)를 이용한 연속적 A/D 변환 시 사용됩니다. 각 채널에 할당되어 있는 버퍼에 현재 존재하는 데이터의 개수를 확인합니다. 이 함수 사용 전 AxaiSetTriggerMode 함수를 사용하여 Timer Trigger Mode 또는 External Trigger Mode 로 설정되어 있어야 합니다.

Example

C++

```
// 0번입력채널의메모리버퍼에데이터가몇개인지검사합니다.
long lDataLength;

AxaiHwReadDataLength(0, &lDataLength);
```

Visual Basic

```
'0번입력채널의메모리버퍼에데이터가몇개인지검사합니다.
Dim lDataLength As Long
AxaiHwReadDataLength 0, lDataLength
```

Delphi

```
{ 0번입력채널의메모리버퍼에데이터가몇개인지검사합니다.}
var
  lDataLength : LongInt;

begin
  AxaiHwReadDataLength(0, @lDataLength);
  Application.MessageBox (PCHAR(IntToStr(lDataLength)), 'Ajinextek', MB_OK);
end;
```

C#

```
// 0번입력채널의메모리버퍼에데이터가몇개인지검사합니다.
int nDataLength;

CAXA.AxaiHwReadDataLength(0, ref nDataLength);
```

See Also

AxaiHwIsBufferEmpty, AxaiHwIsBufferUpper, AxaiHwIsBufferLower

출력모듈정보검색

Function	Description
AxaoInfoGetModuleNoOfChannelNo	지정한 출력 채널 번호로 모듈 번호를 확인한다.
AxaoInfoGetChannelCount	아날로그 출력 모듈의 전체 채널 개수를 확인한다.

AxaoInfoGetModuleNoOfChannelNo

Purpose

지정한 출력 채널 번호로 모듈 번호를 확인합니다.

Format

C++

```
DWORD AxaoInfoGetModuleNoOfChannelNo(long lDacChannelNo, long
*lpModuleNo);
```

Visual Basic

```
Function AxaoInfoGetModuleNoOfChannelNo(ByVal lDacChannelNo As Long,
ByRef lpModuleNo As Long) As Long
```

Delphi

```
functionAxaoInfoGetModuleNoOfChannelNo(lDacChannelNo : LongInt;
lpModuleNo : PLongInt) : DWord; stdcall;
```

C#

```
uintAxaoInfoGetModuleNoOfChannelNo(intnDacChannelNo, ref
intnpModuleNo);
```

Parameters

[in/out] Name	[Init Value] Explanation
[in]lDacChannelNo	출력 채널 번호 (0 ~ 출력 채널 개수 -1)
[out]lpModuleNo	모듈 번호

Return Values

[0000] AXT_RT_SUCCESS :함수실행성공
[1053] AXT_RT_NOT_OPEN :라이브러리초기화실패
[2102] AXT_RT_AIO_INVALID_CHANNEL_NO :유효하지않은 AIO 채널번호
* See error code Table for more information on status error codes

Description

사용자가 지정한 아날로그 출력 채널 번호로 모듈 번호를 확인합니다. 모듈 번호를 파라메터로 갖는 함수들을 이용하여 각기 다른 설정을 함으로써 데이터 수집방식을 달리할 수 있습니다.

Example

C++

```
// 0번째출력채널의모듈번호를확인합니다.
long lModuleNo;
CString strData;

AxaoInfoGetModuleNoOfChannelNo(0, &lModuleNo);
strData.Format("0번째출력채널의모듈번호는 %d 입니다.", lCount);
```

```
AfxMessageBox(strData);
```

Visual Basic

```
' 0번째출력채널의모듈번호를확인합니다.
Dim lModuleNo As Long
Dim strData As String

AxaoInfoGetModuleNoOfChannelNo 0, lModuleNo
strData = "0번째출력채널의모듈번호는" + CStr(lModuleNo) + "입니다."

MsgBox strData
```

Delphi

```
{ 0번째출력채널의모듈번호를확인합니다. }
var
  lModuleNo : LongInt;
  strData : String;

begin
  AxaoInfoGetModuleNoOfChannelNo(0, @lModuleNo);
  strData  :='0번째출력채널의모듈번호는' + IntToStr(lModuleNo) + '입니다.';

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

C#

```
// 0번째출력채널의모듈번호를확인합니다.
int  nModuleNo;
String  strData;

CAXA.AxaoInfoGetModuleNoOfChannelNo(0, ref nModuleNo);
strData.Format("0번째출력채널의모듈번호는 %d 입니다.", lCount);

MessageBox.Show(strData);
```

See Also

AxaoInfoGetChannelCount

AxaoInfoGetChannelCount

Purpose

시스템에 장착된 아날로그 출력 모듈의 전체 채널 개수를 확인합니다.

Format

C++

```
DWORD AxaoInfoGetChannelCount(long *lpDacChannelCount);
```

Visual Basic

```
Function AxaoInfoGetChannelCount(ByRef lpDacChannelCount As Long) As
Long
```

Delphi

```
functionAxaoInfoGetChannelCount(lpDacChannelCount : PLongInt) : DWord;
stdcall;
```

C#

```
uintAxaoInfoGetChannelCount(ref int npDacChannelCount);
```

Parameters

[in/out] Name	[Init Value] Explanation
[out]lpDacChannelCount	출력 모듈의 전체 채널 개수

Return Values

[0000] AXT_RT_SUCCESS :함수실행성공
[1053] AXT_RT_NOT_OPEN :라이브러리초기화실패
[2051]AXT_RT_AIO_NOT_MODULE :AIO 모듈없음
* See error code Table for more information on status error codes

Description

시스템에 장착된 모든 아날로그 출력 모듈의 전체 채널 개수를 확인합니다.

Example

C++

```
// 아날로그출력모듈의전체채널개수를확인합니다.
long lCount;
CString strData;

AxaoInfoGetChannelCount(&lCount);
strData.Format("아날로그출력모듈의전체채널개수는 %d개입니다.", lCount);

AfxMessageBox(strData);
```

Visual Basic

```
'아날로그출력모듈의전체채널개수를확인합니다.
Dim lCount As Long
Dim strData As String

AxaoInfoGetChannelCount lCount
strData = "아날로그출력모듈의전체채널개수는" + CStr(lCount) + "개입니다."

MsgBox strData
```

Delphi

```
{ 아날로그출력모듈의전체채널개수를확인합니다. }
var
  lCount : LongInt;
  strData : String;

begin
  AxaoInfoGetChannelCount(@lCount);
  strData  :='아날로그출력모듈의전체채널개수는' + IntToStr(lCount) + '개입니다.';

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

C#

```
// 아날로그출력모듈의전체채널개수를확인합니다.
int nCount;
String strData;

CAXA.AxaoInfoGetChannelCount(ref nCount);
strData.Format("아날로그출력모듈의전체채널개수는 %d개입니다.", nCount);

MessageBox.Show(strData);
```

See Also

AxaoInfoGetModuleNoOfChannelNo

출력모듈설정및확인

Function	Description
AxaoSetRange	지정한 출력 채널에 출력 전압 범위를 설정한다.
AxaoGetRange	지정한 출력 채널의 출력 전압 범위를 확인한다.
AxaoSetMultiRange	지정한 여러 출력 채널에 출력 전압 범위를 설정한다.
AxaoWriteVoltage	지정한 출력 채널에 입력된 전압이 출력 된다.
AxaoWriteMultiVoltage	지정한 여러 출력 채널에 입력된 전압이 출력 된다.
AxaoReadVoltage	지정한 출력 채널에 출력되는 전압 값을 확인한다.
AxaoReadMultiVoltage	지정한 여러 출력 채널에 출력되는 전압 값을 확인한다.

AxaoSetRange

Purpose

지정한 출력 채널에 출력 전압 범위를 설정합니다.

Format

C++

```
DWORD AxaoSetRange(long lDacChannelNo, double dMinVolt, double
dMaxVolt);
```

Visual Basic

```
Function AxaoSetRange(ByVal lDacChannelNo As Long, ByVal dMinVolt As
Double, ByVal dMaxVolt As Double) As Long
```

Delphi

```
functionAxaoSetRange(lDacChannelNo : LongInt; dMinVolt : Double;
dMaxVolt : Double) : DWord; stdcall;
```

C#

```
uintAxaoSetRange(int nDacChannelNo, double dMinVolt, double dMaxVolt);
```

Parameters

[in/out] Name	[Init Value] Explanation
[in]lDacChannelNo	출력 채널 번호
[in]dMinVolt	최소 출력 전압
[in]dMaxVolt	최대 출력 전압

Return Values

[0000] AXT_RT_SUCCESS :함수실행성공
[1053] AXT_RT_NOT_OPEN :라이브러리초기화실패
[2102] AXT_RT_AIO_INVALID_CHANNEL_NO :유효하지않은 AIO 채널번호
* See error code Table for more information on status error codes

Description

출력 전압 범위는 D/A변환 성능과 관련이 있으므로 사용 범위 확인 후 설정하시기 바랍니다. 모듈별로 허용 출력 전압이 정해져 있습니다. 사용자가 출력 전압 설정범위 이외 전압을 입력하면 자동으로 그 값을 포함하는 근삿값으로 자동 설정됩니다. 자동 설정된 값은AxaoGetRange 함수를 이용해서 확인할 수 있습니다. 최소 출력 전압은 아날로그 출력 모듈의 허용전압 중 최소 전압을 의미합니다. 즉, 사용자는 이 전압보다 작은 전압을 입력할 수 없습니다. 그리고 사용자가 설정한 값은 하드웨어가 지원하는 최소 허용전압을 포함하는 가장 근사치 값으로 설정됩니다. SIO-AO4RB의 경우 최소 기준 전압으로 -10V, -5V, 0V를 제공하므로 사용자가 -8V를 입력하면 -10V로 자동 설정됩니다. 최대 출력 전압은 아날로그 출력의 허용 전압 중 최대 전압을 의미합니다. 즉, 사용자는 이 전압보다 높은 전압을 입력할 수 없다. 그리고 사용자가 설정한 값은 하드웨어가 지원하는 최대 허용전압을 포함하는 가장근사치 값으로 설정됩니다. SIO-AO4RB의 경우 최대 허용전압으로 10V, 5V를 제공하므로 사용자가 8V를 입력하면 10V로 자동 설정됩니다. 반면 분해능이 16bit 인 PCI-AI16HR의 경우 출력전압의 범위가 고정되어 있으므로 함수 수행시 AXT_RT_AIO_INVALID_USE 에러가 반환됩니다.

※참고 :PCI-AI16HR 은 출력 전압 범위가 -10V ~ 10V 로 고정되어 있습니다.

출력 전압 설정 범위

Module	최소 전압	최대 전압	최소 출력 단위
SIO-AO4RB	-10V	10V	약 19.5mV
	-5V	5V	약 9.7mV
PCI-AI16HR(SIO-AO2HB) 출전압범위고정	-10V	10V	약 1.2mV

Example

C++

```
// 0번출력채널에출력전압범위를설정합니다.
AxaoSetRange(0, -10, 10);
```

Visual Basic

```
' 0번출력채널에출력전압범위를설정합니다.
AxaoSetRange 0, -10, 10
```

Delphi

```
{ 0번출력채널에출력전압범위를설정합니다. }
AxaoSetRange(0, -10, 10);
```

C#

```
// 0번출력채널에출력전압범위를설정합니다.
CAXA.AxaoSetRange(0, -10, 10);
```

See Also

AxaoGetRange, AxaoSetMultiRange, AxaoWriteVoltage, AxaoWriteMultiVoltage, AxaoReadVoltage, AxaoReadMultiVoltage

AxaoGetRange

Purpose

지정한 출력 채널에 설정한 출력 전압 범위를 확인합니다.

Format

C++

```
DWORD AxaoGetRange(long lDacChannelNo, double *dpMinVolt, double
*dpMaxVolt);
```

Visual Basic

```
Function AxaoGetRange(ByVal lDacChannelNo As Long, ByRef dpMinVolt As
Double, ByRef dpMaxVolt As Double) As Long
```

Delphi

```
functionAxaoGetRange(lDacChannelNo : LongInt; dpMinVolt : PDouble;
dpMaxVolt : PDouble) : DWord; stdcall;
```

C#

```
uintAxaoGetRange(int nDacChannelNo, ref double dpMinVolt, ref double
dpMaxVolt);
```

Parameters

[in/out] Name	[Init Value] Explanation
[in]lDacChannelNo	출력 채널 번호(0 ~ 출력 채널 개수 -1)
[out]dpMinVolt	최소 출력 전압
[out]dpMaxVolt	최대 출력 전압

Return Values

[0000] AXT_RT_SUCCESS :함수실행성공
[1053] AXT_RT_NOT_OPEN :라이브러리초기화실패
[2102] AXT_RT_AIO_INVALID_CHANNEL_NO :유효하지않은 AIO 채널번호
* See error code Table for more information on status error codes

Description

사용자가 AxaoSetRange함수를 이용해서 허용 출력 전압을 설정한 후 실제로 설정된 값을 확인합니다. 사용자가 H/W에서 제공되지 않는 허용 출력 전압을 설정하였을 경우 실제 설정 값과 다를 수 있습니다.

※참고 :PCI-AI16HR 은 출력 전압 범위가 -10V ~ 10V 로 고정되어 있습니다.

출력 전압 설정 범위

Module	최소 전압	최대 전압	최소 출력 단위
SIO-AO4RB	-10V	10V	약 19.5mV
	-5V	5V	약 9.7mV
PCI-AI16HR(SIO-AO2HB)	-10V	10V	약 1.2mV

출전압범위고정			

Example

C++

```
// 0번출력채널의출력전압범위를확인합니다.
double  dMinVolt;
double  dMaxVolt;

AxaoGetRange(0, &dMinVolt, &dMaxVolt);
```

Visual Basic

```
' 0번출력채널의출력전압범위를확인합니다.
Dim lMask As Long

AxaoGetRange 0, dMinVolt, dMaxVolt
```

Delphi

```
{ 0번출력채널의출력전압범위를확인합니다. }
var
 dMinVolt : Double;
 dMaxVolt : Double;
 strData : String;

begin
  AxaoGetRange(0, @dMinVolt, @dMaxVolt);

  strData := 'MinVolt : ' + FloatToStr(dMinVolt) + ', MaxVolt : ' +
     FloatToStr(dMaxVolt);
  Application.MessageBox (PCHAR(strData), 'Ajinextek', MB_OK);
end;
```

C#

```
// 0번출력채널의출력전압범위를확인합니다.
double  dMinVolt;
double  dMaxVolt;

CAXA.AxaoGetRange(0, ref dMinVolt, ref dMaxVolt);
```

See Also

AxaoSetRange, AxaoSetMultiRange, AxaoWriteVoltage, AxaoWriteMultiVoltage, AxaoReadVoltage, AxaoReadMultiVoltage

AxaoSetMultiRange

Purpose

지정한 여러 출력 채널에 출력 전압 범위를 설정합니다.

Format

C++

```
DWORD AxaoSetMultiRange(long lSize, long *lpDacChannelNo, double
dMinVolt, double dMaxVolt);
```

Visual Basic

```
Function AxaoSetMultiRange(ByVal lSize As Long, ByRef lpDacChannelNo
As Long, ByVal dMinVolt As Double, ByVal dMaxVolt As Double) As Long
```

Delphi

```
functionAxaoSetMultiRange(lSize : LongInt; lpDacChannelNo : PLongInt;
dMinVolt : Double; dMaxVolt : Double) : DWord; stdcall;
```

C#

```
uintAxaoSetMultiRange(int nSize, ref int npDacChannelNo, double
dMinVolt, double dMaxVolt);
```

Parameters

[in/out] Name	[Init Value] Explanation
[in]lSize	출력 채널의 개수
[in]lpDacChannelNo	채널 번호의 배열
[in]dMinVolt	최소 출력 전압
[in]dMaxVolt	최대 출력 전압

Return Values

[0000] AXT_RT_SUCCESS :함수실행성공
[1053] AXT_RT_NOT_OPEN :라이브러리초기화실패
[2102] AXT_RT_AIO_INVALID_CHANNEL_NO :유효하지않은 AIO 채널번호
* See error code Table for more information on status error codes

Description

출력 전압 범위는 D/A변환 성능과 관련이 있으므로 사용 범위 확인 후 설정하시기 바랍니다. 모듈별로 허용 출력 전압이 정해져 있습니다. 사용자가 출력 전압 설정범위 이외 전압을 입력하면 자동으로 그 값을 포함하는 근삿값으로 자동 설정됩니다. 자동 설정된 값은AxaoGetRange 함수를 이용해서 확인할 수 있습니다. 최소 출력 전압은 아날로그 출력 모듈의 허용전압 중 최소 전압을 의미합니다. 즉, 사용자는 이 전압보다 작은 전압을 입력할 수 없습니다. 그리고 사용자가 설정한 값은 하드웨어가 지원하는 최소 허용전압을 포함하는 가장 근사치 값으로 설정됩니다. SIO-AO4RB의 경우 최소 기준 전압으로 -10V, -5V, 0V를 제공하므로 사용자가 -8V를 입력하면 -10V로 자동 설정됩니다. 최대 출력 전압은 아날로그 출력의 허용 전압 중 최대 전압을 의미합니다. 즉, 사용자는 이 전압보다 높은 전압을 입력할 수 없다. 그리고 사용자가 설정한 값은 하드웨어가 지원하는 최대 허용전압을 포함하는 가장근사치 값으로 설정됩니다. SIO-AO4RB의 경우 최대 허용전압으로 10V, 5V를 제공하므로 사용자가

8V를 입력하면 10V로 자동 설정됩니다. 반면 분해능이 16bit 인 PCI-AI16HR의 경우 출력전압의 범위가 고정되어 있으므로 함수 수행시 AXT_RT_AIO_INVALID_USE 에러가 반환됩니다.

※참고 :PCI-AI16HR 은 출력 전압 범위가 -10V ~ 10V 로 고정되어 있습니다.

출력 전압 설정 범위

Module	최소 전압	최대 전압	최소 출력 단위
SIO-AO4RB	-10V	10V	약 19.5mV
	-5V	5V	약 9.7mV
PCI-AI16HR(SIO-AO2HB) 출전압범위고정	-10V	10V	약 1.2mV

Example

C++

```
// 여러출력채널에채널에출력전압범위를설정합니다.
long lDacChannelNo[4] = {0, 1, 2, 3};

AxaoSetMultiRange(4, lDacChannelNo, -10, 10);
```

Visual Basic

```
'여러출력채널에채널에출력전압범위를설정합니다.
Dim lDacChannelNo(0 To 3) As Long

lDacChannelNo(0) = 0
lDacChannelNo(1) = 1
lDacChannelNo(2) = 2
lDacChannelNo(3) = 3

AxaoSetMultiRange 4, lDacChannelNo(0), -10, 10
```

Delphi

```
{ 여러출력채널에채널에출력전압범위를설정합니다. }
var
  lDacChannelNo: array [0..3] of LongInt;

begin
  lDacChannelNo[0]:= 0;
  lDacChannelNo[1]:= 1;
  lDacChannelNo[2]:= 2;
  lDacChannelNo[3]:= 3;

  AxaoSetMultiRange(4, @lDacChannelNo, -10, 10);
end;
```

C#

```
// 여러출력채널에채널에출력전압범위를설정합니다.
Int[] nDacChannelNo = new int[4];
```

```
nDacChannelNo = 0;
nDacChannelNo = 1;
nDacChannelNo = 2;
nDacChannelNo = 3;

AxaoSetMultiRange(4, ref nDacChannelNo[0], -10, 10);
```

See Also

AxaoSetRange, AxaoGetRange, AxaoWriteVoltage, AxaoWriteMultiVoltage, AxaoReadVoltage, AxaoReadMultiVoltage

AxaoWriteVoltage

Purpose

지정한 출력 채널에 입력한값을 전압으로 출력 합니다.

Format

C++

```
DWORD AxaoWriteVoltage(long lDacChannelNo, double dVolt);
```

Visual Basic

```
Function AxaoWriteVoltage(ByVal lDacChannelNo As Long, ByVal dVolt As
Double) As Long
```

Delphi

```
functionAxaoWriteVoltage(lDacChannelNo : LongInt; dVolt : Double) :
DWord; stdcall;
```

C#

```
uintAxaoWriteVoltage(int nDacChannelNo, double dVolt);
```

Parameters

[in/out] Name	[Init Value] Explanation
[in]lDacChannelNo	출력 채널 번호
[in]dVolt	출력 전압 값(V)

Return Values

[0000] AXT_RT_SUCCESS :함수실행성공
[1053] AXT_RT_NOT_OPEN :라이브러리초기화실패
[2102] AXT_RT_AIO_INVALID_CHANNEL_NO :유효하지않은 AIO 채널번호
* See error code Table for more information on status error codes

Description

출력한 전압은 사용자가 dVolt 값을 변경할 때까지 유지됩니다. 이 출력 전압은 AxaoSetRange에서 설정한 최대 범위보다 클 수 없습니다.

Example

C++

```
// 0번출력채널에출력전압범위를 -10 ~ 10으로설정합니다.
AxaoSetRange(0, -10, 10);
// 0번출력채널에 5.7V 전압이출력됩니다.
AxaoWriteVoltage(0, 5.7);
```

Visual Basic

```
' 0번출력채널에출력전압범위를 -10 ~ 10으로설정합니다.
AxaoSetRange 0, -10, 10
'0번출력채널에 5.7V 전압이출력됩니다.
```

```
AxaoWriteVoltage 0, 5.7
```

Delphi

```
{ 0번출력채널에출력전압범위를 -10 ~ 10으로설정합니다. }
AxaoSetRange(0, -10, 10);
{ 0번출력채널에 5.7V 전압이출력됩니다. }
AxaoWriteVoltage(0, 5.7);
```

C#

```
// 0번출력채널에출력전압범위를 -10 ~ 10으로설정합니다.
AxaoSetRange(0, -10, 10);
// 0번출력채널에 5.7V 전압이출력됩니다.
AxaoWriteVoltage(0, 5.7);
```

See Also

AxaoSetRange, AxaoGetRange, AxaoSetMultiRange, AxaoWriteMultiVoltage, AxaoReadVoltage, AxaoReadMultiVoltage

AxaoWriteMultiVoltage

Purpose

지정한 여러 출력 채널에 입력한 값을 전압으로 출력 합니다.

Format

C++

```
DWORD AxaoWriteMultiVoltage(long lSize, long *lpDacChannelNo, double
*dpVolt);
```

Visual Basic

```
Function AxaoWriteMultiVoltage(ByVal lSize As Long, ByRef
lpDacChannelNo As Long, ByRef dpVolt As Double) As Long
```

Delphi

```
functionAxaoWriteMultiVoltage(lSize : LongInt; lpDacChannelNo :
PLongInt; dpVolt : PDouble) : DWord; stdcall;
```

C#

```
uintAxaoWriteMultiVoltage(int nSize, ref int npDacChannelNo, ref
double dpVolt);
```

Parameters

[in/out] Name	[Init Value] Explanation
[in]lSize	사용할 출력 채널의 개수
[in]lpDacChannelNo	사용할 채널 번호의 배열
[in]dpVolt	출력 전압의 배열

Return Values

[0000] AXT_RT_SUCCESS :함수실행성공
[1053] AXT_RT_NOT_OPEN :라이브러리초기화실패
[2102] AXT_RT_AIO_INVALID_CHANNEL_NO :유효하지않은 AIO 채널번호
* See error code Table for more information on status error codes

Description

출력한 전압은 사용자가 dVolt 값을 변경할 때까지 유지됩니다. 이 출력 전압은 AxaoSetRange에서 설정한 최대 범위보다 클 수 없습니다.

Example

C++

```
// 여러출력채널에각각다른전압이출력됩니다.
long lDacChannelNo[4] = {0, 1, 2, 3};
double dVolt[4] = {3.2, 5,3, 6,2, 8.7};

AxaoSetMultiRange(4, lDacChannelNo, -10, 10);
```

```
AxaoWriteMultiVoltage(4, lDacChannelNo, dVolt);
```

Visual Basic

```
'여러출력채널에각각다른전압이출력됩니다.
Dim lDacChannelNo(0 To 3) As Long
Dim dVolt(0 To 3) As Double

lDacChannelNo(0) = 0
lDacChannelNo(1) = 1
lDacChannelNo(2) = 2
lDacChannelNo(3) = 3

dVolt(0) = 3.2
dVolt(1) = 5.3
dVolt(2) = 6.2
dVolt(3) = 8.7

AxaoSetMultiRange4, lDacChannelNo, -10, 10
AxaoWriteMultiVoltage 4, lDacChannelNo(0), dVolt(0)
```

Delphi

```
{ 여러출력채널에각각다른전압이출력됩니다. }
var
  DacChannelNo: array [0..3] of LongInt;
  Volt : array [0..3] of Double;

begin
  DacChannelNo[0]:= 0;
  DacChannelNo[1]:= 1;
  DacChannelNo[2]:= 2;
  DacChannelNo[3]:= 3;
  Volt[0] := 3.2;
  Volt[1] := 5.3;
  Volt[2] := 6.2;
  Volt[3] := 8.7;

  AxaoSetMultiRange(4, lDacChannelNo, -10, 10);
  AxaoWriteMultiVoltage(4, @lDacChannelNo, @dVolt);
end;
```

C#

```
// 여러출력채널에각각다른전압이출력됩니다.
int[] nDacChannelNo = new int[4];
double[] dVolt = new double[4];

nDacChannelNo[0] = 0;
nDacChannelNo[1] = 1;
```

```
nDacChannelNo[2] = 2;
nDacChannelNo[3] = 3;

dVolt[0] = 3.2;
dVolt[1] = 5.3;
dVolt[2] = 6.2;
dVolt[3] = 8.7;

AxaoSetMultiRange(4, ref lDacChannelNo[0], -10, 10);
AxaoWriteMultiVoltage(4, ref lDacChannelNo[0], ref dVolt[0]);
```

See Also

AxaoSetRange, AxaoGetRange, AxaoSetMultiRange, AxaoWriteVoltage, AxaoReadVoltage, AxaoReadMultiVoltage

AxaoReadVoltage

Purpose

지정한 출력 채널에 출력하고 있는 전압 값을 확인합니다.

Format

C++

```
DWORD AxaoReadVoltage(long lDacChannelNo, double *dpVolt);
```

Visual Basic

```
Function AxaoReadVoltage(ByVal lDacChannelNo As Long, ByRef dpVolt As
Double) As Long
```

Delphi

```
functionAxaoReadVoltage(lDacChannelNo : LongInt; dpVolt : PDouble) :
DWord; stdcall;
```

C#

```
uintAxaoReadVoltage(int nDacChannelNo, ref double dpVolt);
```

Parameters

[in/out] Name	[Init Value] Explanation
[in]lDacChannelNo	출력 채널 번호
[out]dpVolt	출력 전압 값(V)

Return Values

[0000] AXT_RT_SUCCESS :함수실행성공
[1053] AXT_RT_NOT_OPEN :라이브러리초기화실패
[2102] AXT_RT_AIO_INVALID_CHANNEL_NO :유효하지않은 AIO 채널번호
* See error code Table for more information on status error codes

Description

AxaoWriteVoltage 또는 AxaoWriteMultiVoltage 함수로 출력하고 있는 전압 값을 확인합니다.

Example

C++

```
// 0번출력채널에아날로그출력값을전압값으로반환합니다.
double  dVolt;

AxaoReadVoltage(0, &dVolt);
```

Visual Basic

```
' 0번입력채널에아날로그입력값을전압값으로반환합니다.
Dim dVolt As Double

AxaoReadVoltage 0, dVolt
```

Delphi

```
{ 0번입력채널에아날로그입력값을전압값으로반환합니다. }
var
  dVolt : Double;

begin
  AxaoReadVoltage(0, @dVolt);
  Application.MessageBox (PCHAR(FloatToStr(dVolt)), 'Ajinextek', MB_OK);
end;
```

C#

```
// 0번출력채널에아날로그출력값을전압값으로반환합니다.
double  dVolt;

AxaoReadVoltage(0, ref dVolt);
```

See Also

AxaoSetRange, AxaoGetRange, AxaoSetMultiRange, AxaoWriteVoltage, AxaoWriteMultiVoltage, AxaoReadMultiVoltage

AxaoReadMultiVoltage

Purpose

지정한 여러 출력 채널에 출력하고 있는 전압 값을 확인합니다.

Format

C++

```
DWORD AxaoReadMultiVoltage(long lSize, long *lpDacChannelNo, double
*dpVolt);
```

Visual Basic

```
Function AxaoReadMultiVoltage(ByVal lSize As Long, ByRef
lpDacChannelNo As Long, ByRef dpVolt As Double) As Long
```

Delphi

```
functionAxaoReadMultiVoltage(lSize : LongInt; lpDacChannelNo:
PLongInt; dpVolt : PDouble) : DWord; stdcall;
```

C#

```
uintAxaoReadMultiVoltage(int nSize, ref int npDacChannelNo, ref
double dpVolt);
```

Parameters

[in/out] Name	[Init Value] Explanation
[in]lSize	출력 채널의 개수
[in]lpDacChannelNo	채널 번호의 배열
[out]dpVolt	출력 전압 값의 배열

Return Values

[0000] AXT_RT_SUCCESS :함수실행성공
[1053] AXT_RT_NOT_OPEN :라이브러리초기화실패
[2102] AXT_RT_AIO_INVALID_CHANNEL_NO :유효하지않은 AIO 채널번호
* See error code Table for more information on status error codes

Description

AxaoWriteVoltage 또는 AxaoWriteMultiVoltage 함수로 출력하고 있는 전압 값을 확인합니다.

Example

C++

```
// 여러출력채널에각각출력되는전압값을확인합니다.
long lDacChannelNo[4] = {0, 1, 2, 3};
double dVolt[4]

AxaoReadMultiVoltage(4, lDacChannelNo, dVolt);
```

Visual Basic

```
'여러출력채널에각각출력되는전압값을확인합니다.
Dim lDacChannelNo(0 To 3) As Long
Dim dVolt(0 To 3) As Double

lDacChannelNo(0) = 0
lDacChannelNo(1) = 1
lDacChannelNo(2) = 2
lDacChannelNo(3) = 3

AxaoReadMultiVoltage 4, lDacChannelNo(0), dVolt(0)
```

Delphi

```
{ 여러출력채널에각각출력되는전압값을확인합니다. }
var
  lDacChannelNo: array [0..3] of LongInt;
  dVolt : array [0..3] of Double;

begin
  lDacChannelNo[0]:= 0;
  lDacChannelNo[1]:= 1;
  lDacChannelNo[2]:= 2;
  lDacChannelNo[3]:= 3;

  AxaoReadMultiVoltage(4, @lDacChannelNo, @dVolt);
end;
```

C#

```
// 여러출력채널에각각출력되는전압값을확인합니다.
int[] nDacChannelNo = new int[4];
double[] dVolt = new double[4];

nDacChannelNo[0]= 0;
nDacChannelNo[1]= 1;
nDacChannelNo[2]= 2;
nDacChannelNo[3]= 3;

AxaoReadMultiVoltage(4, ref nDacChannelNo[0], ref dVolt[0]);
```

See Also

AxaoSetRange, AxaoGetRange, AxaoSetMultiRange, AxaoWriteVoltage, AxaoWriteMultiVoltage, AxaoReadVoltage

Extern Trigger 함수

Function	Description
AxaiExternalStartADC	지정한 입력모듈의 선택된 채널들의 외부 트리거 모드를 시작한다.
AxaiExternalStopADC	지정한 입력모듈의 외부트리거 모드를 정지한다.
AxaiExternalReadFifoStatus	지정한 입력모듈의 Fifo상태를 반환한다.
AxaiExternalReadVoltage	지정한 입력모듈의 외부신호에 의해 변환된 A/D값을 읽어온다.

AxaiExternalStartADC

Purpose

지정한 입력모듈의 선택된 채널들의 외부 트리거 모드를 시작 합니다.

Format

C++

```
DWORD AxaiExternalStartADC (long lModuleNo, long lSize, long
*lpChannelPos);
```

Visual Basic

```
Function AxaiExternalStartADC (ByVal lModuleNo As Long, ByVal lSize
As Long, ByRef lpChannelPos As Long) As Long
```

Delphi

```
Function AxaiExternalStartADC (lModuleNo: LongInt; lSize: LongInt;
lpModuleNo : PLongInt) : DWord; stdcall;
```

C#

```
Uint AxaiExternalStartADC (Int lModuleNo ,lSize ,ref Int
lpChannelPos);
```

Parameters

[in/out] Name	[Init Value] Explanation
[in] lModuleNo	모듈 번호
[in] lSize	지정한 입력 모듈에서 외부트리거를 사용 할 채널갯수
[in] lpChannelPos	지정한 입력 모듈에서 외부트리거를 사용 할 채널의 Index

Return Values

[0000] AXT_RT_SUCCESS :함수실행성공
[1053] AXT_RT_NOT_OPEN :라이브러리초기화실패
[2101] AXT_RT_AIO_INVALID_MODULE_NO :유효하지않은 AIO 모듈번호
* See error code Table for more information on status error codes

Description

사용자가 지정한 아날로그 입력 모듈 번호로 모듈의 선택된 채널들의 외부 트리거 모드를 시작 한다.

Example

C++

```
// 0번째 모듈의 0, 1번 채널 외부 트리거 모드를 시작.
long lpChannel[2];

lpChannel[0] = 0;
lpChannel[1] = 1;
```

```
AxaiExternalStartADC (0, 2, &lpChannel);
```

See Also

AxaiExternalStartADC, AxaiExternalStopADC, AxaiExternalReadFifoStatus, AxaiExternalReadVoltage

AxaiExternalStopADC

Purpose

지정한 입력모듈의 선택된 채널들의 외부 트리거 모드를 정지 합니다.

Format

C++

```
DWORD AxaiExternalStopADC (long lModuleNo);
```

Visual Basic

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

Delphi

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

C#

```
Uint AxaiExternalStopADC (Int lModuleNo);
```

Parameters

[in/out] Name	[Init Value] Explanation
[in] lModuleNo	모듈 번호

Return Values

[0000] AXT_RT_SUCCESS :함수실행성공
[1053] AXT_RT_NOT_OPEN :라이브러리초기화실패
[2101] AXT_RT_AIO_INVALID_MODULE_NO :유효하지않은 AIO 모듈번호
* See error code Table for more information on status error codes

Description

사용자가 지정한 아날로그 입력 모듈 번호로 모듈의 선택된 채널들의 외부 트리거 모드를 시작 한다.

Example

C++

```
// 0번째 모듈의 0, 1번 채널 외부 트리거 모드를 정지.

AxaiExternalStopADC (0);
```

See Also

AxaiExternalStartADC, AxaiExternalStopADC, AxaiExternalReadFifoStatus, AxaiExternalReadVoltage

AxaiExternalReadFifoStatus

Purpose

지정한 입력모듈의 Fifo 상태를 반환합니다.

Format

C++

```
DWORD AxaiExternalReadFifoStatus (long lModuleNo, DWORD *dwStatus);
```

Visual Basic

```
Function AxaiExternalReadFifoStatus (ByVal lModuleNo As Long, ByRef
lpStatus As Long) As Long
```

Delphi

```
Function AxaiExternalReadFifoStatus (lModuleNo: LongInt; lSize:
LongInt; lpStatus : PLongInt) : DWord; stdcall;
```

C#

```
Uint AxaiExternalReadFifoStatus (Int lModuleNo, ref Int lpStatus);
```

Parameters

[in/out] Name	[Init Value] Explanation
[in] lModuleNo	모듈 번호
[out] lpStatus	FIFO 상태 -[00] FIFO_DATA_EXIST -[01] FIFO_DATA_EMPTY -[02] FIFO_DATA_HALF -[06] FIFO_DATA_FULL

Return Values

[0000] AXT_RT_SUCCESS :함수실행성공
[1053] AXT_RT_NOT_OPEN :라이브러리초기화실패
[2101] AXT_RT_AIO_INVALID_MODULE_NO :유효하지않은 AIO 모듈번호
* See error code Table for more information on status error codes

Description

사용자가 지정한 아날로그 입력 모듈 번호로 모듈의 FIFO 상태를 반환 한다.

Example

C++

```
// 0번째 모듈의 FIFO 상태를 읽어 온다.
DWORD dwStatus;
CString strData;

AxaiExternalReadFifoStatus (0, & dwStatus);
```

```
strData.Format("0번째 아날로그 모듈의 FIFO 상태는 %d 입니다." dwStatus);

AfxMessageBos(StrData);
```

See Also

AxaiExternalStartADC, AxaiExternalStopADC, AxaiExternalReadFifoStatus, AxaiExternalReadVoltage

AxaiExternalReadVoltage

Purpose

지정한 입력모듈의 외부신호에 의해 변환된 A/D 값을 읽어옵니다.

Format

C++

```
DWORD AxaiExternalReadVoltage (long lModuleNo, long lSize, long
*lpChannelPos, long lDataSize, long lBuffSize, long lStartDataPos,
double *dpVolt, long *lpRetDataSize, DWORD *dwpStatus);
```

Visual Basic

```
Function AxaiExternalReadVoltage (ByVal lModuleNo As Long, ByVal
lSize As Long, ByRef lpChannelPos As Long, ByVal lDataSize As Long,
ByVal lBuffSize As Long, ByVal lStartDataPos As Long, ByRef dpVolt As
Double, ByRef lpRetDataSize As Long, ByRef dwpStatus As Long) As Long
```

Delphi

```
Function AxaiExternalReadVoltage (lModuleNo : LongInt, lSize :
LongInt, lpChannelPos : PLongInt, lDataSize : LongInt, lBuffSize :
LongInt, lStartDataPos : LongInt, dpVolt : PDouble, lpRetDataSize :
LongInt, dwpStatus : LongInt) : DWord; stdcall;
```

C#

```
Uint AxaiExternalReadVoltage (Int lModuleNo, Int lSize, ref Int
lpChannelPos, Int lDataSize, Int lBuffSize, Int lStartDataPos, ref
double *dpVolt, ref Int *lpRetDataSize, ref Int *dwpStatus);
```

Parameters

[in/out] Name	[Init Value] Explanation
[in] lModuleNo	모듈 번호
[in] lSize	지정한 입력 모듈에서 변환된 A/D값을 읽어올 채널의 갯수 (AxaiExternalStartADC에 사용한 채널갯수와 동일 해야됨)
[in] lpChannelPos	지정한 입력 모듈에서 변환된 A/D값을 읽어올 채널의 Index(AxaiExternalStartADC에 사용한 채널의 Index와 동일 해야됨)
[in] lDataSize	외부트리거에 의해 A/D변환된 값을 한번에 읽어 올 최대 데이타의 갯수
[in] lBuffSize	외부에서(사용자 Program) 할당한 Data Buffer의 Size
[in] lStartDataPos	외부에서(사용자 Program) 할당한 Data Buffer에 저장 시작 할 위치
[out] dpVolt	A/D변환된 값을 할당 받을 2차원 배열 포인트(dpVlot[Channel][Count])
[in] lpRetDataSize	A/D변환된 값이 Data Buffer에 실제 할당된 갯수
[in] dwpStatus	A/D변환된 값을 Fifo(H/W Buffer)로 부터 읽을 때 Fifo상태를 반환함.

Return Values

[0000] AXT_RT_SUCCESS :함수실행성공
[1053] AXT_RT_NOT_OPEN :라이브러리초기화실패
[2101] AXT_RT_AIO_INVALID_MODULE_NO :유효하지않은 AIO 모듈번호
* See error code Table for more information on status error codes

Description

사용자가 지정한 아날로그 입력 모듈 번호로 외부 신호에 의해 변환된 A/D 값을 읽어 온다. 읽어올 Data 의 Size 는 AxaiExternalStartADC 에 사용한 채널갯수와 동일 해야 하며 Index 역시 AxaiExternalStartADC 에 사용한 채널의 Index 와 동일 해야 한다.

Example

C++

```
// 0번째 모듈의 0번째 채널
long lpChannel[2];
long lpData[2][100];
long lDataSize;
DWORD dwStatus;

lpChannel[0] = 0;
lpChannel[1] = 1;

AxaiExternalStartADC (0, 2, &lpChannel);
Sleep(1000);
AxaiExternalStopADC(0);

AxaiExternalReadVoltage (0, 2, &lpChannel, 100, 100, 0, &lpData, &lDataSize,
&dwStatus);
```

See Also

AxaiExternalStartADC, AxaiExternalStopADC, AxaiExternalReadFifoStatus, AxaiExternalReadVoltage

에러코드테이블 (Error Code Table) 확인

라이브러리 사용 중 오 동작 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 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
[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. EzConfig 에서 해당 보드의 Lock 정보와 실제 연결상태가 일치 하지 않음. 슬레이브 재구성시 기존의 Lock 정보는 Ezconfig 에서 Unlock 으로 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	디섯번째 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 축이 마스터채널
[4156] AXT_RT_MOTION_ERROR_MASTER_SERVOON	마스터 축 서보온이 안되어있을 때
[4157] AXT_RT_MOTION_ERROR_SLAVE_SERVOON	슬레이브 축 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 축이 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 축이 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 멥핑 번호가 비워 있을 때
[4180] AXT_ERROR_SPLINE_INVALID_AXIS_NO	스플라인 축 맵핑 시 유효한 축이 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_OUTOFBOUND	설정한 위치값이 설정 최대값보다 크거나 최소값보다 작은값임
[4252] AXT_RT_MOTION_PROFILE_INVALID	구동 속도 프로파일 설정이 잘못됨
[4253] AXT_RT_MOTION_VELOCITY_OUTOFBOUND	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	순차구동함수실행중자원할당실패
[4401] AXT_ERROR_SEQ_INVALID_MAP_NO	순차구동함수실행중맵핑번호이상.
[4402] AXT_ERROR_INVALID_AXIS_NO	함수설정인자중축번호이상.
[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

이설명서의내용은예고없이변경될수있습니다. 용례에사용된회사, 기관, 제품, 인물및사건등은실제데이터가아닙니다. 어떠한실제회사, 기관, 제품, 인물또는사건과도연관시킬의도가없으며그렇게유추해서도안됩니다. 해당저작권법을준수하는것은사용자의책임입니다. 저작권에서의권리와는별도로, 이설명서의어떠한부분도 (주) 아진엑스텍의명시적인서면승인없이는어떠한형식이나수단(전기적, 기계적, 복사기에의한복사, 디스크복사또는다른방법) 또는다른목적으로도복제되거나, 검색시스템에저장또는도입되거나, 전송될수없습니다.

(주) 아진엑스텍은이설명서본안에관련된특허권, 상표권, 저작권또는기타지적소유권등을보유할수있습니다. 서면사용권계약에따라 (주) 아진엑스텍으로부터귀하에게명시적으로제공된권리이외에, 이설명서의제공은귀하에게이러한특허권, 저작권또는기타지적소유권등에대한어떠한사용권도허용하지않습니다.