Library User Manual

AnyBus Series

Carrier Board

AJINEXTEK AXT

Product Information

Full information about other AJINEXTEK products
is available by visiting our Web Site at:
www.ajinextek.com

Useful Contact Information

Customer Support Seoul
Tel : 82-31-389-1580~2 Fax: 82-31-389-1583
E-mail : marketing@ajinextek.com

Customer Support Cheonan
Tel : 82-41-555-9771~2 Fax: 82-41-555-9773
E-mail : ljh001024@ajinextek.com

Customer Support Deagu
Tel : 82-53-593-3700 Fax: 82-53-593-3703
E-mail : support@ajinextek.com

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

Contents

Revision History

Manual	PCB	Comments
Rev. 2.1 issue 1.0	Rev. 1.0	2004.03.30
Rev. 2.0 issue 1.0	Rev. 1.0	2003.06.17

1. 매뉴얼 정보

본 매뉴얼은 BIHR/BIFR/BPHR/BPFR/BV3R/BV6R/BC3R/BC6R/BPHD 캐리어보드를 Windows 98, Windows NT, Windows 2000 또는 Windows XP 의 OS 환경에서 Microsoft VC++6.0, Visual Basic, Borland C-Builder, Delphi 등의 언어에서 구동하기 위해 필요한 매뉴얼이며, 포함된 라이브러리 함수를 기능별로 분류하여 설명하였다.

1.1.헤더 파일

C

AxtLIB.h

Visual Basic

AxtLIB.bas

Delphi

AxtLIB.pas

1.2.함수 용어

1.2.1.본 매뉴얼의 함수 이름

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

라이브러리 함수 **Prefix**

Axt : 캐리어보드 전용 함수임을 나타낸다. Axt 로 시작되는 함수들은 모두 AxtLIB.h 에 정의되어있다.

get : 변수 값을 확인하거나 칩 레지스터의 상태를 읽는다.

1.2.2.본 매뉴얼의 인자 이름

본 매뉴얼에서 사용된 함수들의 공통적인 인자들은 다음과 같은 의미를 가진다.

INT16 nBoardNo : 초기화 된 캐리어보드의 첫 번째 보드부터 오름차순으로 자동 정렬된다. 보드 번호는 '0'부터 시작한다.

INT16 BusType : 버스 타입은 ISA, PCI, VME, CompactPCI 총 네 종류가 있는데 아래의 Table 을 참조하여 설정 및 확인할 수 있다.

버스 타입 종류

#define	Value	Explanation
BUSTYPE_ISA	0	ISA Type
BUSTYPE_PCI	1	PCI Type
BUSTYPE_VME	2	VME Type
BUSTYPE_CPCI	3	CompactPCI Type

2. Quick List

2.1.라이브러리 초기화

2.2.캐리어보드 초기화

2.3.인터럽트 설정

	다.	
AxtWriteInterruptMaskModule	지정한 캐리어보드의 각 모듈에 인터럽트 사용 여부를 설정한다.	25
AxtReadInterruptMaskModule	지정한 캐리어보드의 각 모듈에 인터럽트 사용 여부를 확인한다.	27
AxtReadInterruptFlagModule	지정한 캐리어보드의 각 모듈의 인터럽트 상태를 확인한다.	29

2.4.캐리어보드 정보

Function	Description	Page
AxtGetBoardID	지정한 캐리어보드의 ID를 확인한다.	32
AxtGetBoardCounts	버스 타입에 관계없이 사용 등록된 캐리어보드의 개수를 확인한다.	34
AxtGetBoardCountsBus	지정한 버스 타입의 사용 등록된 캐리어보드의 개수를 확인한다.	35
AxtGetModuleCounts	지정한 캐리어보드에 장착된 모듈 개수와 각 모듈ID를 확인한다.	36
AxtGetModelCounts	지정한 캐리어보드에 장착된 모듈 중 특정 ID를 가진 모듈의 개수를 확인한다.	38
AxtGetModelCountsAll	버스 타입에 관계없이 모든 캐리어보드에 장착된 특정 ID를 가진 모듈의 개수를 확인한다.	40
AxtGetLibVersion	라이브러리 버전을 확인한다.	41
AxtGetLibDate	라이브러리 최종 수정일을 확인한다.	42

3. Function List

3.1.라이브러리 초기화

3.1.1.AxtInitialize

Purpose

라이브러리를 초기화 한다.

Format

C

```
BOOL AxtInitialize(HANDLE hWnd, INT16 nIrqNo);
```

Visual Basic

```
Function AxtInitialize(ByVal hWnd As Byte, ByVal nIrqNo As Integer) As Byte
```

Delphi

```
function AxtInitialize(hWnd : HWND; nIrqNo : SmallInt) : Boolean; stdcall;
```

Input

hWnd	윈도우 HANDLE
nIrqNo	IRQ 번호

Output

Return	TRUE : 라이브러리 초기화 성공
	FALSE : 라이브러리 초기화 실패

Description

라이브러리를 초기화하고 사용자가 인터럽트를 사용할 경우 인터럽트를 처리할 윈도우의 HANDLE 과 IRQ 번호를 등록함으로써 인터럽트를 사용하게 해준다. 이때 PNP 기능이 지원되는 PCI 버스와 cPCI 버스인경우 IRQ 번호는 자동 할당 되므로 사용자가 설정한 IRQ 번호는 무시된다.

윈도우 HANDLE 은 인터럽트를 사용할 때 인터럽트 메시지를 받아 처리할 윈도우의 핸들을 넘겨주면 된다. 만약 MFC 를 지원하지 않는 Consol 모드일 경우에는 인터럽트 처리 방식으로 메시지 방식을 사용하지 않고 Callback 함수의 포인터를 이용해서 처리하므로 인자로 NULL 을 넣어주면 된다.

IRQ 번호는 ISA 버스나 VME 버스의 경우에는 캐리어보드에 있는 Irq 설정 Dip S/W 의 설정 값을 넣어주면 되고 PCI 버스나 cPCI 버스의 경우에는 PNP 기능이 지원되므로 nIrqNo 가 자동 할당되므로 아무 값이나 넣어주면 된다. 제품 출고 시 초기 값으로 Irq 7 로 설정되어 있다.

폐사에서 제공하는 라이브러리를 사용하기 위해서는 AxtInitialize 함수를 반드시 호출한 후 사용해야 한다. 그리고 사용이 끝난 후에는 AxtClose 함수를 호출하여 할당 받은 메모리를 해제해 주어야 한다. 그렇지 않을 경우에 메모리 누수가 생겨 시스템에 문제가 발생할 수도 있다.

Example

```
// 라이브러리를 초기화 한다.
// m_hWnd 은 윈도우 HANDLE 이고 7 은 IRQ 를 뜻한다.
if (AxtInitialize(m_hWnd, 7))
{
    AfxMessageBox("라이브러리가 초기화 되었습니다.");
}
else
{
    AfxMessageBox("라이브러리가 초기화 되지 못했습니다.");
}
```

See Also

AxtIsInitialized, AxtClose

3.1.2.AxtIsInitialized

Purpose

라이브러리가 초기화 되어 있는 지 확인한다.

Format

C

```
BOOL AxtIsInitialized();
```

Visual Basic

```
Function AxtIsInitialized() As Byte
```

Delphi

```
function AxtIsInitialized() : Boolean; stdcall;
```

Input

None

Output

Return	TRUE : 라이브러리가 초기화 되어 있음
	FALSE : 라이브러리가 초기화 되어 있지 않음

Description

라이브러리가 초기화 되어있어 사용 가능한 지 확인한다. 만약 라이브러리가 초기화 되지 않은 상태에서 함수들을 사용하게 되면 여러 가지 문제를 야기하므로 반드시 이 함수를 이용해서 라이브러리가 초기화 되었는지 검사한 후 사용해야 한다.

Example

```
// 라이브러리가 초기화 되어 있는 지 확인한다.
if (AxtIsInitialized())
{
    AfxMessageBox("라이브러리가 초기화 되어 있습니다.");
}
else
{
    AfxMessageBox("라이브러리가 초기화 되지 않았습니다.");
}
```

See Also

AxtInitialize, AxtClose

3.1.3.AxtClose

Purpose

라이브러리 사용을 종료 한다.

Format

C

```
void AxtClose();
```

Visual Basic

```
Sub AxtClose()
```

Delphi

```
procedure AxtClose(); stdcall;
```

Input

None

Output

None

Description

라이브러리 사용을 종료 한다. 할당 받은 메모리를 해제하므로 라이브러리 사용이 종료되면 반드시 호출해줘야 된다. 다시 사용하기 위해서는 초기화 하여야 한다.

Example

```
// 라이브러리 사용을 종료 한다.
AxtClose();
```

See Also

AxtInitialize, AxtIsInitialized

3.2.캐리어보드 초기화

Function	Description	Page
AxtOpenDevice	지정한 버스 타입의 베이스 어드레스를 가진 캐리어보드를 사용 가능하게 초기화 한다.	11
AxtOpenDeviceAll	지정한 버스 타입의 베이스 어드레스를 가진 여러 장의 캐리어보드를 사용 가능하게 초기화 한다.	13
AxtOpenDeviceAuto	지정한 버스 타입의 캐리어보드를 모두 자동으로 사용 가능하게 초기화 한다.	15
AxtIsInitializedBus	지정한 버스 타입의 캐리어보드가 초기화 되었는지 확인한다.	17
AxtCloseDeviceAll	초기화된 모든 캐리어보드를 등록을 해제한다.	18

3.2.1.AxtOpenDevice

Purpose

지정한 버스 타입의 베이스 어드레스를 가진 캐리어보드를 사용 가능하게 초기화 한다.

Format

C

```
INT16 AxtOpenDevice(INT16 BusType, UINT32 dwBaseAddr);
```

Visual Basic

```
Function AxtOpenDevice(ByVal BusType As Integer, ByVal dwBaseAddr As Long) As Integer
```

Delphi

```
function AxtOpenDevice(BusType : SmallInt; dwBaseAddr : DWord) : SmallInt; stdcall;
```

Input

BusType	버스 타입
dwBaseAddr	베이스 어드레스

Output

Return	TRUE : 캐리어보드 초기화 성공
	FALSE : 캐리어보드 초기화 실패

Description

사용자가 지정한 버스 타입의 베이스 어드레스를 가진 캐리어보드를 사용 가능하게 초기화 하고 라이브러리에 등록한다.

버스 타입은 초기화 하고 라이브러리에 등록할 캐리어보드 버스 타입으로 사용자는 원하는 버스 타입을 선택한 후 인자로 넣어주면 된다. 종류는 아래의 Table 을 참조하여 설정한다.

버스 타입 종류

#define	Value	Explanation
BUSTYPE_ISA	0	ISA Bus Type
BUSTYPE_PCI	1	PCI Bus Type
BUSTYPE_VME	2	VME Bus Type
BUSTYPE_CPCI	3	CompactPCI Bus Type

베이스 어드레스는 ISA 버스나 VME 버스의 경우에는 사용하고자 하는 캐리어보드에 있는 Address 설정 Dip S/W 의 설정 값을 넣어주면 되고 PCI 버스나 cPCI 버스의 경우에는 PNP 기능이 지원되어 Address 가 자동 할당되므로 아무 값이나 넣어주면 된다.

Example

```
// ISA 버스 타입의 000D8000h 베이스 어드레스를 가진 캐리어보드를 사용 가능하게 초기화
한다.
if (AxtOpenDevice(BUSTYPE_ISA, 0x000D8000))
{
    AfxMessageBox("000D8000h 베이스 어드레스를 가진 캐리어보드가 초기화
되었습니다.");
}
else
{
    AfxMessageBox("000D8000h 베이스 어드레스를 가진 캐리어보드가 초기화 되지
못했습니다.");
}
```

See Also

AxtOpenDeviceAll, AxtOpenDeviceAuto, AxtIsInitializedBus, AxtCloseDeviceAll

3.2.2.AxtOpenDeviceAll

Purpose

지정한 버스 타입의 베이스 어드레스를 가진 여러 장의 캐리어보드를 사용 가능하게 초기화 한다.

Format

C

```
INT16 AxtOpenDeviceAll(INT16 BusType, INT16 nLen, UINT32 *dwBaseAddr);
```

Visual Basic

```
Function AxtOpenDeviceAll(ByVal BusType As Integer, ByVal nLen As Integer, ByRef dwBaseAddr As Long) As Integer
```

Delphi

```
function AxtOpenDeviceAll(BusType : SmallInt; nLen : SmallInt; dwBaseAddr : PDWord) : SmallInt; stdcall;
```

Input

BusType	버스 타입
nLen	캐리어보드의 개수
*dwBaseAddr	베이스 어드레스를 담을 배열

Output

Return	성공적으로 등록된 캐리어보드의 개수

Description

동일한 버스 타입의 캐리어보드가 여러 장 장착되어 있을 경우 일일이 등록하면 번거로우므로 등록할 캐리어보드의 개수와 어드레스 배열을 인자로 넣어주면 동시에 초기화 하고 라이브러리에 등록한다.

버스 타입은 초기화 하고 라이브러리에 등록할 캐리어보드 버스 타입으로 사용자는 원하는 버스 타입을 선택한 후 인자로 넣어주면 된다. 종류는 AxtOpenDevice 의 버스 타입 종류 Table 을 참조하여 설정한다.

캐리어보드의 개수는 동일한 버스타입의 캐리어보드가 여러 장 장착되어 있을 경우 동시에 여러 장의 캐리어 보드를 초기화할 경우 초기화할 캐리어보드의 개수를 넣어주면 된다.

*dwBaseAddr 는 베이스 어드레스를 담을 배열로서 사용할 베이스 어드레스를 배열의 값으로 넣어주면 된다. 예를 들어 UINT32 dwBaseAddr[4] = {0x000D8000, 0x000DA000, 0x000DC000, 0x000DE000}; 처럼 하면 4 개의 캐링어보드를 동시에 초기화를 한다는 것이다.

이 베이스 어드레스는 ISA 버스나 VME 버스의 경우에는 사용하고자 하는 캐리어보드에 있는 Address 설정 Dip S/W 의 설정 값을 넣어주면 되고 PCI 버스나 cPCI 버스의 경우에는 PNP 기능이 지원되어 Address 가 자동 할당되므로 아무 값이나 넣어주면 된다.

Example

```
// ISA 버스 타입의 베이스 어드레스를 가진 여러 장의 캐리어보드를 사용 가능하게 초기화
한다.
UINT32  dwBaseAddr[4] = {0x000D8000, 0x000DA000, 0x000DC000, 0x000DE000};
INT16   nCount;
CString strData;

nCount  = AxtOpenDeviceAll(BUSTYPE_ISA, dwBaseAddr);

if (nCount > -1)
{
    strData.Format("%d 개의 ISA 버스 타입의 캐리어보드를 초기화 되었습니다.",
nCount);
}
else
{
    strData.Format("ISA 버스 타입의 캐리어보드를 초기화 되지 못했습니다.",
nCount);
}

AfxMessageBox(strData);
```

See Also

AxtOpenDevice, AxtOpenDeviceAuto, AxtIsInitializedBus, AxtCloseDeviceAll

3.2.3.AxtOpenDeviceAuto

Purpose

지정한 버스 타입의 캐리어보드를 모두 자동으로 사용 가능하게 초기화 한다.

Format

C

```
INT16 AxtOpenDeviceAuto(INT16 BusType);
```

Visual Basic

```
Function AxtOpenDeviceAuto(ByVal BusType As Integer) As Integer
```

Delphi

```
function AxtOpenDeviceAuto(BusType : SmallInt) : SmallInt; stdcall;
```

Input

BusType	버스 타입

Output

Return	성공적으로 등록된 캐리어보드의 개수

Description

사용자가 캐리어보드의 베이스 어드레스를 확인하지 않고 이 함수에 캐리어보드의 버스 타입만 지정해주면 동일 버스 타입의 모든 캐리어보드를 자동으로 등록해준다. 모든 캐리어보드를 등록해서 사용 할 경우 매우 편리하며 가장 많이 사용된다.

버스 타입은 초기화 하고 라이브러리에 등록할 캐리어보드 버스 타입으로 사용자는 원하는 버스 타입을 선택한 후 인자로 넣어주면 된다. 종류는 AxtOpenDevice 의 버스 타입 종류 Table 을 참조하여 설정한다.

Example

```
// PCI 버스 타입의 캐리어보드를 모두 자동으로 사용 가능하게 초기화 한다.
INT16   nCount;
CString strData;

nCount  = AxtOpenDeviceAuto(BUSTYPE_PCI);

if (nCount > 0)
{
    strData.Format("%d 개의 PCI 버스 타입의 캐리어보드를 초기화 되었습니다.",
nCount);
}
else
{
    strData.Format("PCI 버스 타입의 캐리어보드를 초기화 되지 못했습니다.",
nCount);
```

```
}

AfxMessageBox(strData);
```

See Also

AxtOpenDevice, AxtOpenDeviceAll, AxtIsInitializedBus, AxtCloseDeviceAll

3.2.4.AxtIsInitializedBus

Purpose

지정한 버스 타입의 캐리어보드가 초기화 되었는지 확인한다.

Format

C

```
INT16 AxtIsInitializedBus(INT16 BusType);
```

Visual Basic

```
Function AxtIsInitializedBus(ByVal BusType As Integer) As Integer
```

Delphi

```
function AxtIsInitializedBus(BusType : Smallint) : Smallint; stdcall;
```

Input

BusType	버스 타입

Output

Return	TRUE : 지정한 버스 타입의 캐리어보드가 초기화 되어 있음
	FALSE : 지정한 버스 타입의 캐리어보드가 초기화 되어 있지 않음

Description

사용자가 지정한 버스 타입의 캐리어보드가 초기화 되었는지 확인한다.

버스 타입은 캐리어보드 버스 타입으로 사용자는 원하는 버스 타입을 선택한 후 인자로 넣어주면 된다. 종류는 AxtOpenDevice 의 버스 타입 종류 Table 을 참조하여 설정한다.

Example

```
// PCI 버스 타입의 캐리어보드가 초기화 되었는지 확인한다.
if (AxtIsInitializedBus(BUSTYPE_PCI))
{
    AfxMessageBox("PCI 버스 타입의 캐리어보드가 초기화 되어 있습니다.");
}
else
{
    AfxMessageBox("PCI 버스 타입의 캐리어보드가 초기화 되지 않았습니다.");
}
```

See Also

AxtOpenDevice, AxtOpenDeviceAll, AxtOpenDeviceAuto, AxtCloseDeviceAll

3.2.5.AxtCloseDeviceAll

Purpose

초기화된 모든 캐리어보드를 등록을 해제한다.

Format

C

```
void AxtCloseDeviceAll();
```

Visual Basic

```
Sub AxtCloseDeviceAll()
```

Delphi

```
procedure AxtCloseDeviceAll(); stdcall;
```

Input

None

Output

None

Description

초기화하여 등록되어 있는 모든 종류의 캐리어보드를 등록 해제한다. 이 함수를 호출한 후 모듈을 제어하려면 다시 캐리어보드를 등록해야 한다. 단 이 함수는 캐리어보드 등록만 해제 하는 함수이므로 새로 라이브러리를 초기화 하지는 않아도 된다.

Example

```
// 초기화된 모든 캐리어보드를 등록을 해제한다.
AxtCloseDeviceAll();
```

See Also

AxtOpenDevice, AxtOpenDeviceAll, AxtOpenDeviceAuto, AxtIsInitializedBus

3.3.인터럽트 설정

Function	Description	Page
AxtEnableInterrupt	지정한 캐리어보드에 인터럽트를 사용 가능하게 설정한다.	20
AxtDisableInterrupt	지정한 캐리어보드에 인터럽트를 사용 불가능하게 해제한다.	21
AxtIsEnableInterrupt	지정한 캐리어보드에 인터럽트 사용 가능 여부를 확인한다.	22
AxtInterruptFlagClear	지정한 캐리어보드에 인터럽트 플래그를 지운다.	24
AxtWriteInterruptMaskModule	지정한 캐리어보드의 각 모듈에 인터럽트 사용 여부를 설정한다.	25
AxtReadInterruptMaskModule	지정한 캐리어보드의 각 모듈에 인터럽트 사용 여부를 확인한다.	27
AxtReadInterruptFlagModule	지정한 캐리어보드의 각 모듈의 인터럽트 상태를 확인한다.	29

3.3.1.AxtEnableInterrupt

Purpose

지정한 캐리어보드에 인터럽트를 사용 가능하게 설정한다.

Format

C

```
void AxtEnableInterrupt(INT16 nBoardNo);
```

Visual Basic

```
Sub AxtEnableInterrupt(ByVal nBoardNo As Integer)
```

Delphi

```
procedure AxtEnableInterrupt(nBoardNo : SmallInt); stdcall;
```

Input

nBoardNo 캐리어보드 번호

Output

None

Description

사용자가 인터럽트를 사용하고자 하는 캐리어보드의 번호를 인자로 넘겨받아 인터럽트를 사용할 수 있게 해 준다.

캐리어보드 번호는 AxtOpenDevice, AxtOpenDeviceAll, AxtOpenDeviceAuto 함수를 이용하여 캐리어보드를 초기화하고 라이브러리에 등록하게 되면 자동으로 할당되는 번호이다. 사용자는 이 번호를 이용해서 각 캐리어보드를 제어할 수 있다.

캐리어보드는 초기화하는 순서대로 캐리어보드 번호가 '0'부터 오름차순으로 정렬 되어 있다.

Example

```
// 0번 캐리어보드에 인터럽트를 사용 가능하게 설정한다.
AxtEnableInterrupt(0);
```

See Also

AxtDisableInterrupt, AxtIsEnableInterrupt

3.3.2.AxtDisableInterrupt

Purpose

지정한 캐리어보드에 인터럽트를 사용 불가능하게 해제한다.

Format

C

```
void AxtDisableInterrupt(INT16 nBoardNo);
```

Visual Basic

```
Sub AxtDisableInterrupt(ByVal nBoardNo As Integer)
```

Delphi

```
procedure AxtDisableInterrupt(nBoardNo : SmallInt); stdcall;
```

Input

nBoardNo 캐리어보드 번호

Output

None

Description

사용자가 인터럽트를 해제하고자 하는 캐리어보드의 번호를 인자로 넘겨받아 인터럽트 사용을 해제해 준다.

캐리어보드 번호는 AxtOpenDevice, AxtOpenDeviceAll, AxtOpenDeviceAuto 함수를 이용하여 캐리어보드를 초기화하고 라이브러리에 등록하게 되면 자동으로 할당되는 번호이다. 사용자는 이 번호를 이용해서 각 캐리어보드를 제어할 수 있다.

캐리어보드는 초기화하는 순서대로 캐리어보드 번호가 '0'부터 오름차순으로 정렬 되어 있다.

Example

```
// 0번 캐리어보드에 인터럽트를 사용 불가능하게 해제한다.
AxtDisableInterrupt(0);
```

See Also

AxtEnableInterrupt, AxtIsEnableInterrupt

3.3.3.AxtIsEnableInterrupt

Purpose

지정한 캐리어보드에 인터럽트 사용 가능 여부를 확인한다.

Format

C

```
BOOL AxtIsEnableInterrupt(INT16 nBoardNo);
```

Visual Basic

```
Function AxtIsEnableInterrupt(ByVal nBoardNo As Integer) As Byte
```

Delphi

```
function AxtIsEnableInterrupt(nBoardNo : SmallInt) : Boolean; stdcall;
```

Input

nBoardNo	캐리어보드 번호

Output

Return	TRUE : 인터럽트 사용 가능
	FALSE : 인터럽트 사용 불가능

Description

사용자가 지정한 캐리어보드에 인터럽트가 허용상태인지 검사해서 허용 상태이면 'TRUE'를 반환하고 금지 상태이면 'FALSE'를 반환 한다.

캐리어보드 번호는 AxtOpenDevice, AxtOpenDeviceAll, AxtOpenDeviceAuto 함수를 이용하여 캐리어보드를 초기화하고 라이브러리에 등록하게 되면 자동으로 할당되는 번호이다. 사용자는 이 번호를 이용해서 각 캐리어보드를 제어할 수 있다.

캐리어보드는 초기화하는 순서대로 캐리어보드 번호가 '0'부터 오름차순으로 정렬 되어 있다.

Example

```
// 0 번 캐리어보드에 인터럽트 사용 가능 여부를 확인한다.
if (AxtIsEnableInterrupt(0))
{
    AfxMessageBox("0 번 캐리어보드에 인터럽트 사용 가능합니다.");
}
else
{
    AfxMessageBox("0 번 캐리어보드에 인터럽트 사용 불가능합니다.");
}
```

See Also

AxtEnableInterrupt, AxtDisableInterrupt

3.3.4.AxtInterruptFlagClear

Purpose

지정한 캐리어보드에 인터럽트 플래그를 지운다.

Format

C

```
void AxtInterruptFlagClear(INT16 nBoardNo);
```

Visual Basic

```
Sub AxtInterruptFlagClear(ByVal nBoardNo As Integer)
```

Delphi

```
procedure AxtInterruptFlagClear(nBoardNo : SmallInt); stdcall;
```

Input

nBoardNo 캐리어보드 번호

Output

None

Description

사용자가 지정한 캐리어보드에 인터럽트가 걸리게 되면 Interrupt Flag 가 'HIGH'(1)로 설정된다. 이 플래그가 'HIGH' 라는 의미는 인터럽트가 걸린 상태를 의미하므로 인터럽트 처리가 끝나면 다시 인터럽트 상태를'LOW'(0)값으로 만들어 주어야 하는데 이때 사용하는 함수 이다.

캐리어보드 번호는 AxtOpenDevice, AxtOpenDeviceAll, AxtOpenDeviceAuto 함수를 이용하여 캐리어보드를 초기화하고 라이브러리에 등록하게 되면 자동으로 할당되는 번호이다. 사용자는 이 번호를 이용해서 각 캐리어보드를 제어할 수 있다.

캐리어보드는 초기화하는 순서대로 캐리어보드 번호가 '0'부터 오름차순으로 정렬 되어 있다.

Example

```
// 0 번 캐리어보드에 인터럽트 플래그를 지운다.
AxtInterruptFlagClear(0);
```

See Also

AxtReadInterruptFlagModule

3.3.5.AxtWriteInterruptMaskModule

Purpose

지정한 캐리어보드의 각 모듈에 인터럽트 사용 여부를 설정한다.

Format

C

```
void AxtWriteInterruptMaskModule(INT16 nBoardNo, UINT8 Mask);
```

Visual Basic

```
Sub AxtWriteInterruptMaskModule(ByVal nBoardNo As Integer, ByVal Mask As Byte)
```

Delphi

```
procedure AxtWriteInterruptMaskModule(nBoardNo : SmallInt; Mask : Byte); stdcall;
```

Input

nBoardNo	캐리어보드 번호
Mask	특정 모듈에 대한 인터럽트 사용 여부 설정

Output

None

Description

사용자가 지정한 캐리어보드에 장착된 각 모듈에 인터럽트 사용 여부를 설정한다.

캐리어보드 번호는 AxtOpenDevice, AxtOpenDeviceAll, AxtOpenDeviceAuto 함수를 이용하여 캐리어보드를 초기화하고 라이브러리에 등록하게 되면 자동으로 할당되는 번호이다. 사용자는 이 번호를 이용해서 각 캐리어보드를 제어할 수 있다.

캐리어보드는 초기화하는 순서대로 캐리어보드 번호가 '0'부터 오름차순으로 정렬 되어 있다.

특정 모듈에 대한 인터럽트 사용 여부 설정은 bit0 에서 bit3 까지가 각 모듈의 Mask bit 를 의미하며 bit7 은 Global bit 로 이 bit 를 'HI'(1)로 해 주어야 각 모듈의 Mask bit 가 유효하게 된다.

예를 들어 Module0 와 Module1 의 인터럽트를 허용하려면 Mask 값으로 0x83 을 넣어주면 된다.

아래의 Mask bit 종류 Table 을 참조하여 설정한다.

Mask bit 종류

Bit	Explanation
Bit 0	Module 0 (SUB1)
Bit 1	Module 1 (SUB2)
Bit 2	Module 2 (SUB3)
Bit 3	Module 3 (SUB4)

Bit 7	Global : 이 Bit를 On 해줘야 사용 가능함

Example

```
// 0 번 캐리어보드의 모든 모듈에 인터럽트 사용 가능하게 설정한다.
AxtWriteInterruptMaskModule(0, 0x8F);
```

See Also

AxtReadInterruptMaskModule

3.3.6.AxtReadInterruptMaskModule

Purpose

지정한 캐리어보드의 각 모듈에 인터럽트 사용 여부를 확인한다.

Format

C

```
UINT8 AxtReadInterruptMaskModule(INT16 nBoardNo);
```

Visual Basic

```
Function AxtReadInterruptMaskModule(ByVal nBoardNo As Integer) As Byte
```

Delphi

```
function AxtReadInterruptMaskModule(nBoardNo : SmallInt) : Byte; stdcall;
```

Input

nBoardNo	캐리어보드 번호

Output

Return	특정 모듈에 대한 인터럽트 사용 여부 설정 값

Description

사용자가 지정한 캐리어보드에 장착된 각 모듈에 인터럽트 사용 여부를 확인한다.

캐리어보드 번호는 AxtOpenDevice, AxtOpenDeviceAll, AxtOpenDeviceAuto 함수를 이용하여 캐리어보드를 초기화하고 라이브러리에 등록하게 되면 자동으로 할당되는 번호이다. 사용자는 이 번호를 이용해서 각 캐리어보드를 제어할 수 있다.

캐리어보드는 초기화하는 순서대로 캐리어보드 번호가 '0'부터 오름차순으로 정렬 되어 있다.

반환 값은 bit0 에서 bit3 까지가 각 모듈의 Mask bit 를 의미하며 bit7 은 Global bit 로 이 bit 가 'HI'(1)이면 각 모듈의 Mask bit 가 유효하며 'LOW'(0)일 경우에는 모든 모듈의 인터럽트가 금지된 상태를 의미한다.

예를 들어 반환값이 0x8F 라면 모든 모듈의 인터럽트가 허용상태를 의미한다. 만약 0x0F 라면 모든 모듈의 인터럽트가 금지됐음을 의미한다.

AxtWriteInterruptMaskModule 의 Mask bit 종류 Table 을 참조하여 확인한다.

Example

```
// 0 번 캐리어보드의 각 모듈에 인터럽트 사용 여부를 확인한다.
UINT8   Mask;

Mask    = AxtReadInterruptMaskModule(0);

if (Mask & 0x01)
    printf("모듈 0(SUB1) 인터럽트 사용 가능합니다.\n");
```

```
if (Mask & 0x02)
    printf("모듈 1(SUB2) 인터럽트 사용 가능합니다.\n");

if (Mask & 0x04)
    printf("모듈 2(SUB3) 인터럽트 사용 가능합니다.\n");

if (Mask & 0x08)
    printf("모듈 3(SUB4) 인터럽트 사용 가능합니다.\n");

if (Mask & 0x80)
    printf("Global bit On 되어 있습니다.\n");
```

See Also

AxtWriteInterruptMaskModule

3.3.7.AxtReadInterruptFlagModule

Purpose

지정한 캐리어보드의 각 모듈의 인터럽트 상태를 확인한다.

Format

C

```
UINT8 AxtReadInterruptFlagModule(INT16 nBoardNo);
```

Visual Basic

```
Function AxtReadInterruptFlagModule(ByVal nBoardNo As Integer) As Byte
```

Delphi

```
function AxtReadInterruptFlagModule(nBoardNo : SmallInt) : Byte; stdcall;
```

Input

nBoardNo	캐리어보드 번호

Output

Return	각 모듈의 인터럽트 상태

Description

사용자가 지정한 캐리어보드의 각 모듈의 인터럽트 상태를 확인한다.

인터럽트가 걸리게 되면 해당 모듈의 인터럽트 플래그가 'HI'(1)로 설정되게 된다. 따라서 이 함수를 이용해 반환 값을 분석하면 어떤 모듈에서 인터럽트가 걸렸는지 알 수 있다. 이 함수의 반환 값은 bit0 에서 bit3 까지가 각 모듈의 인터럽트 플래그를 의미한다.

캐리어보드 번호는 AxtOpenDevice, AxtOpenDeviceAll, AxtOpenDeviceAuto 함수를 이용하여 캐리어보드를 초기화하고 라이브러리에 등록하게 되면 자동으로 할당되는 번호이다. 사용자는 이 번호를 이용해서 각 캐리어보드를 제어할 수 있다.

캐리어보드는 초기화하는 순서대로 캐리어보드 번호가 '0'부터 오름차순으로 정렬 되어 있다.

반환 값은 bit0 에서 bit3 까지가 각 모듈의 인터럽트 플래그를 의미한다. 해당 Bit 가 On 이 되면 인터럽트가 걸렸음을 의미한다.

AxtWriteInterruptMaskModule 의 Mask bit 종류 Table 을 참조하여 확인한다.

Example

```
// 0 번 캐리어보드의 각 모듈의 인터럽트 상태를 확인한다.
UINT8   Flag;

Flag    = AxtReadInterruptFlagModule(0);

    printf("모듈 0(SUB1) 인터럽트가 걸렸습니다.\n");
```

```
if (Flag & 0x02)
    printf("모듈 1(SUB2) 인터럽트가 걸렸습니다.\n");

if (Flag & 0x04)
    printf("모듈 2(SUB3) 인터럽트가 걸렸습니다.\n");

if (Flag & 0x08)
    printf("모듈 3(SUB4) 인터럽트가 걸렸습니다.\n");
```

See Also

AxtInterruptFlagClear

3.4.캐리어보드 정보

3.4.1.AxtGetBoardID

Purpose

지정한 캐리어보드의 ID 를 확인한다.

Format

C

```
INT16 AxtGetBoardID(INT16 nBoardNo);
```

Visual Basic

```
Function AxtGetBoardID(ByVal nBoardNo As Integer) As Integer
```

Delphi

```
function AxtGetBoardID(nBoardNo : SmallInt) : SmallInt; stdcall;
```

Input

nBoardNo	캐리어보드 번호

Output

Return	캐리어보드의 ID

Description

사용자가 지정한 캐리어보드의 ID 를 확인한다. 이 반환한 값을 확인하면 사용자가 지정한 캐리어보드가 버스 타입과 캐리어보드 종류를 알 수 있다.

캐리어보드 번호는 AxtOpenDevice, AxtOpenDeviceAll, AxtOpenDeviceAuto 함수를 이용하여 캐리어보드를 초기화하고 라이브러리에 등록하게 되면 자동으로 할당되는 번호이다. 사용자는 이 번호를 이용해서 각 캐리어보드를 제어할 수 있다.

캐리어보드는 초기화하는 순서대로 캐리어보드 번호가 '0'부터 오름차순으로 정렬 되어 있다.

캐리어보드 종류

#define	ID	Explanation
AXT_BIHR	01h	ISA bus, Half size
AXT_BIFR	02h	ISA bus, Full size
AXT_BPHR	03h	PCI bus, Half size
AXT_BPFR	04h	PCI bus, Full size
AXT_BV3R	05h	VME bus, 3U size
AXT_BV6R	06h	VME bus, 6U size
AXT_BC3R	07h	Compact PCI bus, 3U size
AXT_BC6R	08h	Compact PCI bus, 6U size

Example

```
// 0 번 캐리어보드의 ID 를 확인한다.
switch (AxtGetBoardID(0))
{
case AXT_BIHR:
    AfxMessageBox("BIHR 캐리어보드 입니다.");
    break;

case AXT_BIFR:
    AfxMessageBox("BIFR 캐리어보드 입니다.");
    break;

case AXT_BPHR:
    AfxMessageBox("BPHR 캐리어보드 입니다.");
    break;

case AXT_BPFR:
    AfxMessageBox("BPFR 캐리어보드 입니다.");
    break;

case AXT_BV3R:
    AfxMessageBox("BV3R 캐리어보드 입니다.");
    break;

case AXT_BV6R:
    AfxMessageBox("BV6R 캐리어보드 입니다.");
    break;

case AXT_BC3R:
    AfxMessageBox("BC3R 캐리어보드 입니다.");
    break;

case AXT_BC6R:
    AfxMessageBox("BC6R 캐리어보드 입니다.");
    break;

default:
    AfxMessageBox("알 수 없는 캐리어보드 입니다.");
    break;
}
```

See Also

AxtGetBoardCounts, AxtGetBoardCountsBus

3.4.2.AxtGetBoardCounts

Purpose

버스 타입에 관계없이 사용 등록된 캐리어보드의 개수를 확인한다.

Format

C

```
INT16 AxtGetBoardCounts();
```

Visual Basic

```
Function AxtGetBoardCounts() As Integer
```

Delphi

```
function AxtGetBoardCounts() : SmallInt; stdcall;
```

Input

None

Output

Return 캐리어보드 개수

Description

버스 타입에 관계없이 사용 등록된 캐리어보드의 개수를 확인한다. 만약, 이 반환 값이 -1 이 나오면 등록이 정상적으로 이루어지지 않았다는 의미이다.

Example

```
// 버스 타입에 관계없이 사용 등록된 캐리어보드의 개수를 확인한다.
INT16   nCount;
CString strData;

nCount  = AxtGetBoardCounts();
strData.Format("전체 캐리어보드 개수는 %d 개 입니다.", nCount);

AfxMessageBox(strData);
```

See Also

AxtGetBoardID, AxtGetBoardCountsBus

3.4.3.AxtGetBoardCountsBus

Purpose

지정한 버스 타입의 사용 등록된 캐리어보드의 개수를 확인한다.

Format

C

```
INT16 AxtGetBoardCountsBus(INT16 nBusType);
```

Visual Basic

```
Function AxtGetBoardCountsBus(ByVal nBusType As Integer) As Integer
```

Delphi

```
function AxtGetBoardCountsBus(nBusType : SmallInt) : SmallInt; stdcall;
```

Input

nBoardNo	캐리어보드 번호

Output

Return	캐리어보드 개수

Description

사용자가 지정한 버스 타입의 사용 등록된 캐리어보드의 개수를 확인한다. 만약, 이 반환 값이 -1 이 나오면 등록이 정상적으로 이루어지지 않았거나 지정한 버스 타입의 캐리어보드가 장착되지 않았다는 의미이다.

버스 타입은 캐리어보드 버스 타입으로 사용자는 원하는 버스 타입을 선택한 후 인자로 넣어주면 된다. 종류는 AxtOpenDevice 의 버스 타입 종류 Table 을 참조하여 설정한다.

Example

```
// PCI 버스 타입의 사용 등록된 캐리어보드의 개수를 확인한다.
INT16   nCount;
CString strData;

nCount  = AxtGetBoardCountsBus(BUSTYPE_PCI);
strData.Format("PCI 버스 타입의 캐리어보드 개수는 %d 개 입니다.", nCount);

AfxMessageBox(strData);
```

See Also

AxtGetBoardID, AxtGetBoardCounts

3.4.4.AxtGetModuleCounts

Purpose

지정한 캐리어보드에 장착된 모듈 개수와 각 모듈 ID 를 확인한다.

Format

C

```
INT16 AxtGetModuleCounts(INT16 nBoardNo, UINT8 *ModuleID);
```

Visual Basic

```
Function AxtGetModuleCounts(ByVal nBoardNo As Integer, ByRef ModuleID As Byte) As Integer
```

Delphi

```
function AxtGetModuleCounts(nBoardNo : SmallInt; ModuleID : PByte) : SmallInt; stdcall;
```

Input

nBoardNo	캐리어보드 번호

Output

Return	모듈 개수
*ModuleID	모듈 ID 의 배열

Description

사용자가 지정한 지정한 캐리어보드에 장착된 모듈 개수와 각 모듈 ID 를 확인한다.

사용자가 지정한 캐리어보드에 장착된 모듈의 개수를 확인하고 해당 모듈의 ID 를 배열을 이용해서 확인한다. 지정한 캐리어보드에 장착된 모듈의 개수와 종류를 파악할 수 있다.

캐리어보드 번호는 AxtOpenDevice, AxtOpenDeviceAll, AxtOpenDeviceAuto 함수를 이용하여 캐리어보드를 초기화하고 라이브러리에 등록하게 되면 자동으로 할당되는 번호이다. 사용자는 이 번호를 이용해서 각 캐리어보드를 제어할 수 있다.

캐리어보드는 초기화하는 순서대로 캐리어보드 번호가 '0'부터 오름차순으로 정렬 되어 있다.

모듈 ID 의 배열은 모듈의 ID 를 넘겨 받을 변수로 UINT8 ModuleID[4];로 정의 하면 된다. 여기서 4 가 의미하는 것은 현재 제공되고 있는 캐리어보드에 최대 장착 가능한 모듈의 개수를 의미한다.

모듈 ID 는 아래의 Table 을 참조한다.

모듈 종류

#define	ID	Explanation
AXT_SMC_2V01	01h	SMC-2V01, CAMC-5M, 2 Axes
AXT_SMC_2V02	02h	SMC-2V02, CAMC-FS, 2 Axes
AXT_SMC_1V01	03h	SMC-1V01, CAMC-5M, 1 Axis

AXT_SMC_1V02	04h	SMC-1V02, CAMC-FS, 1 Axis
AXT_SMC_4V51	33h	SMC-4V51, MCX314, 4 Axes
AXT_SMC_2V53	35h	SMC-2V53, PMD, 2 Axes
AXT_SMC_2V54	36h	SMC-2V54, MCX312, 2 Axes
AXT_SIO_DI32	97h	SIO-DI32, Digital IN 32점
AXT_SIO_DO32P	98h	SIO-DO32P, Digital OUT 32점
AXT_SIO_DB32P	99h	SIO-DB32P, Digital IN 16점 / OUT 16점
AXT_SIO_DO32T	9Eh	SIO-DO32T, Digital OUT 16점, Power TR 출력
AXT_SIO_DB32T	9Fh	SIO-DB32T, Digital IN 16점 / OUT 16점, Power TR 출력
AXT_SIO_AI4R	A1h	SIO-AI4R, Analog IN 4Ch, 12 bit
AXT_SIO_AO4R	A2h	SIO-AO4R, Analog OUT 4Ch, 12 bit
AXT_SIO_AI16H	A3h	SIO-AI16H, Analog IN 16Ch, 16 bit
AXT_SIO_AO8H	A4h	SIO-AO8H, Analog OUT 8Ch, 16 bit
AXT_COM_234R	D3h	COM-234R, COM 4Ch
AXT_COM_484R	D4h	COM-484R, COM 4Ch

Example

```
// 0번 캐리어보드에 장착된 모듈 개수와 각 모듈 ID를 확인한다.
INT16   nCount;
UINT8   ModuleID[4];
CString strData;

nCount  = AxtGetModuleCounts(0, ModuleID);
strData.Format("0번 캐리어보드에 장착된 모듈 개수는 %d개이고\n모듈 ID는 0:%02X,
1:%02X, 2:%02X, 3:%02X, 입니다.",
    nCount, ModuleID[0], ModuleID[1], ModuleID[2], ModuleID[3]);

AfxMessageBox(strData);
```

See Also

AxtGetModuleCounts, AxtGetModuleCountsAll

3.4.5.AxtGetModelCounts

Purpose

지정한 캐리어보드에 장착된 모듈 중 특정 ID 를 가진 모듈의 개수를 확인한다.

Format

C

```
INT16 AxtGetModelCounts(INT16 nBoardNo, UINT8 ModuleID);
```

Visual Basic

```
Function AxtGetModelCounts(ByVal nBoardNo As Integer, ByVal ModuleID As Byte) As Integer
```

Delphi

```
function AxtGetModelCounts(nBoardNo : SmallInt; ModuleID : Byte) : SmallInt; stdcall;
```

Input

nBoardNo	캐리어보드 번호
ModuleID	모듈 ID

Output

Return	특정 모듈 개수

Description

사용자가 지정한 캐리어보드에 장착된 모듈 중 특정 ID 를 가진 모듈의 개수를 확인한다.

캐리어보드 번호는 AxtOpenDevice, AxtOpenDeviceAll, AxtOpenDeviceAuto 함수를 이용하여 캐리어보드를 초기화하고 라이브러리에 등록하게 되면 자동으로 할당되는 번호이다. 사용자는 이 번호를 이용해서 각 캐리어보드를 제어할 수 있다.

캐리어보드는 초기화하는 순서대로 캐리어보드 번호가 '0'부터 오름차순으로 정렬 되어 있다.

모듈 ID 는 특정 모듈을 구분하는 번호로써 각 모듈마다 고유의 번호를 가지고 있다. 참고로 아날로그 입력 모듈 중 모델 명 SIO-AI4R 의 경우 0xA1 이며 SIO-AI16H 의 경우 0xA3 이다.

자세한 내용은 AxtGetModuleCounts 의 모듈 종류 Table 을 참조하여 설정한다.

Example

```
// 0 번 캐리어보드에 장착된 모듈 중 SMC-2V02 를 가진 모듈의 개수를 확인한다.
INT16   nCount;
CString strData;

nCount  = AxtGetModelCounts(0, AXT_SMC_2V02);

nCount  = AxtGetModuleCounts(0, ModuleID);
strData.Format("0 번 캐리어보드에 장착된 모듈 중 SMC-2V02 를 가진 모듈 개수는 %d 개
입니다.", nCount);
```

```
AfxMessageBox(strData);
```

See Also

AxtGetModuleCounts, AxtGetModuleCountsAll

3.4.6.AxtGetModelCountsAll

Purpose

버스 타입에 관계없이 모든 캐리어보드에 장착된 특정 ID 를 가진 모듈의 개수를 확인한다.

Format

C

```
INT16 AxtGetModelCountsAll(UINT8 ModuleID);
```

Visual Basic

```
Function AxtGetModelCountsAll(ByVal ModuleID As Byte) As Integer
```

Delphi

```
function AxtGetModelCountsAll(ModuleID : Byte) : SmallInt; stdcall;
```

Input

ModuleID	모듈 ID

Output

Return	특정 모듈 개수

Description

버스 타입에 관계없이 사용 등록된 모든 캐리어보드에 장착된 특정 ID 를 가진 모듈의 개수를 확인한다. 사용자가 지정한 모듈이 등록된 캐리어보드에 몇 개가 장착되어 있는지 알 수 있다.

모듈 ID 는 특정 모듈을 구분하는 번호로써 각 모듈마다 고유의 번호를 가지고 있다. 참고로 아날로그 입력 모듈 중 모델 명 SIO-AI4R 의 경우 0xA1 이며 SIO-AI16H 의 경우 0xA3 이다.

자세한 내용은 AxtGetModuleCounts 의 모듈 종류 Table 을 참조하여 설정한다.

Example

```
// 버스 타입에 관계없이 모든 캐리어보드에 장착된 모듈 중 SMC-2V02 를 가진 모듈의 개수를
확인한다.
INT16   nCount;
CString strData;

nCount  = AxtGetModelCountsAll(AXT_SMC_2V02);

nCount  = AxtGetModuleCounts(0, ModuleID);
strData.Format("버스 타입에 관계없이 모든 캐리어보드에 장착된 모듈 중 SMC-2V02 를
가진 모듈 개수는 %d 개 입니다.", nCount);

AfxMessageBox(strData);
```

See Also

AxtGetModuleCounts, AxtGetModelCounts

3.4.7.AxtGetLibVersion

Purpose

라이브러리 버전을 확인한다.

Format

C

```
char *AxtGetLibVersion();
```

Visual Basic

```
Function AxtGetLibVersion() As String
```

Delphi

```
function AxtGetLibVersion() : PChar; stdcall;
```

Input

None

Output

Return 라이브러리 버전

Description

라이브러리 버전을 확인한다. 라이브러리가 주기적으로 업데이트 되니 버전 정보나 최종 수정일을 확인하여 업데이트 여부를 결정할 수 있다.

Example

```
// 라이브러리 버전을 확인한다.
CString    strVersion;

strVersion.Format("현재 라이브러리 버전은 %s 입니다.", AxtGetLibVersion());

AfxMessageBox(strVersion);
```

See Also

AxtGetLibDate

3.4.8.AxtGetLibDate

Purpose

라이브러리 최종 수정일을 확인한다.

Format

C

```
char *AxtGetLibDate();
```

Visual Basic

```
Function AxtGetLibDate() As String
```

Delphi

```
function AxtGetLibDate() : PChar; stdcall;
```

Input

None

Output

Return 라이브러리 최종 수정일

Description

라이브러리 최종 수정일을 확인한다. 라이브러리가 주기적으로 업데이트 되니 버전 정보나 최종 수정일을 확인하여 업데이트 여부를 결정할 수 있다.

Example

```
// 라이브러리 최종 수정일을 확인한다.
CString     strDate;

strDate.Format("현재 라이브러리 최종 수정일은 %s 입니다.", AxtGetLibDate());

AfxMessageBox(strDate);
```

See Also

AxtGetLibVersion

4. 찾아보기