Library User Manual

AXL library family

AXM

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 2008 AJINEXTEK co.ltd. All rights reserved.

Contents

CONTENTS **3**

MOTION 라이브러리 사용법 **12**

모션라이브러리 개요 **14**

모션제어 개요 14

모션라이브러리 개요 15

초기화 및 종료 **17**

라이브러리의 초기화 17

라이브러리의 종료 21

모션 파라미터 **23**

모션 파라미터의 설정 24

모션신호 **37**

모션신호 설정 38

범용 입출력 신호의 읽기 및 출력 쓰기 47

설정 정보의 저장 및 불러오기 51

모션 구동 **54**

모션구동 개요 55

1) One Axis 55

2) Multi Axis 56

모션구동의 용어해설 56

1) Servo On 56

2) 속도 프로파일 58

3) 시점탈출과 종점 탈출 59

4) 상대위치구동과 절대 위치구동 60

단축 구동 61
1) 위치구동 61
2) 속도구동 66
3) 신호검색 구동 72
4) 원점 검색 구동 80
5) 모션 구동의 정지 86
6) 모션 구동상태 확인 92
다축 구동 100
1) 다축위치구동 100
2) 보간 구동 106

확장 모션구동 122
CRC 신호 설정 123
오버라이드 구동 129
MPG 구동 (Manual Pulse Generation) 137
동기구동 149
갠트리 구동 155
1) 겐트리 구동 설정 및 해제 155
2) 갠트리 구동에서의 홈 검색 157
가상축 매핑 169
트리거 신호의 사용 170

네트워크 상태에 대한 모션 구동 에러 처리 182

MOTION COMMAND 매뉴얼 정보 185

헤더 파일 185

본 매뉴얼의 함수 이름 185

MOTION COMMAND QUICK LIST 188

함수리스트 189
노드 및 모듈 확인함수(Info) – Information 189
가상축 함수(Virtual) 190
모션파라메타 설정함수(Mot) -> Motor Parameter 190
입출력 설정함수(Signal) 193
모션 구동 후에 상태 확인함수(Status) 195
홈(원점) 검색 함수(Home) 195
위치 구동 함수(Move) 198
위치 및 속도 오버라이드 구동 함수(Override) 199
MASTER, SLAVE 구동 함수(Link) 199
겐트리 구동 함수(Gantry) 200
보간 구동 함수(Line, Circle) 200
트리거 함수 (Trigger) 201
CRC(잔여 펄스 클리어) 구동 함수 202
MPG(Manul pulse Generator) 구동 함수 202
네트워크 상태에 대한 모션 구동 에러 처리 203

Define 문 204

MOTION COMMAND FUNCTION LIST 215

노드 및 모듈 초기화 216
AxmInfoGetAxis 217
AxmInfoIsMotionModule 219
AxmInfoIsInvalidAxisNo 221
AxmInfoGetAxisCount 223
AxmInfoGetFirstAxisNo 225

가상축 함수 227
AxmVirtualSetAxisNoMap 228
AxmVirtualGetAxisNoMap 230

AxmVirtualSetMultiAxisNoMap 232
AxmVirtualGetMultiAxisNoMap 235
AxmVirtualResetAxisMap 238

모션 파라메타 설정함수 240
AxmMotLoadParaAll 243
AxmMotSaveParaAll 247
AxmMotSetParaLoad 251
AxmMotGetParaLoad 255
AxmMotSetPulseOutMethod 259
AxmMotGetPulseOutMethod 263
AxmMotSetEncInputMethod 266
AxmMotGetEncInputMethod 269
AxmMotSetMoveUnitPerPulse 271
AxmMotGetMoveUnitPerPulse 275
AxmMotSetDecelPos 277
AxmMotGetDecelPos 279
AxmMotSetRemainPulse 281
AxmMotGetRemainPulse 284
AxmMotSetMaxVel 286
AxmMotGetMaxVel 288
AxmMotSetAbsRelMode 290
AxmMotGetAbsRelMode 292
AxmMotSetProfileMode 294
AxmMotGetProfileMode 297
AxmMotSetAccelUnit 299
AxmMotGetAccelUnit 301
AxmMotSetMinVel 303
AxmMotGetMinVel 305
AxmMotSetAccelJerk 307

AxmMotGetAccelJerk 309
AxmMotSetDecelJerk 311
AxmMotGetDecelJerk 313

입출력 레벨 설정 함수 315
AxmSignalSetZphaseLevel 317
AxmSignalGetZphaseLevel 319
AxmSignalSetServoOnLevel 321
AxmSignalGetServoOnLevel 323
AxmSignalSetServoAlarmResetLevel 325
AxmSignalGetServoAlarmResetLevel 327
AxmSignalSetInpos 329
AxmSignalGetInpos 332
AxmSignalReadInpos 334
AxmSignalSetServoAlarm 336
AxmSignalGetServoAlarm 338
AxmSignalReadServoAlarm 340
AxmSignalSetLimit 342
AxmSignalGetLimit 345
AxmSignalReadLimit 348
AxmSignalSetSoftLimit 350
AxmSignalGetSoftLimit 353
AxmSignalSetStop 356
AxmSignalGetStop 358
AxmSignalReadStop 360
AxmSignalServoOn 362
AxmSignalIsServoOn 365
AxmSignalServoAlarmReset 367
AxmSignalWriteOutput 369
AxmSignalReadOutput 372

AxmSignalWriteOutputBit 374
AxmSignalReadOutputBit 377
AxmSignalReadInput 379
AxmSignalReadInputBit 381

모션 구동 후에 상태 확인 함수 384
AxmStatusReadInMotion 385
AxmStatusReadDrivePulseCount 388
AxmStatusReadMotion 390
AxmStatusReadStop 397
AxmStatusReadMechanical 406
AxmStatusReadVel 410
AxmStatusReadPosError 412
AxmStatusReadDriveDistance 414
AxmStatusSetActPos 416
AxmStatusGetActPos 418
AxmStatusSetCmdPos 420
AxmStatusGetCmdPos 422

원점 검색 함수 424
AxmHomeSetSignalLevel 431
AxmHomeGetSignalLevel 433
AxmHomeReadSignal 435
AxmHomeSetMethod 437
AxmHomeGetMethod 441
AxmHomeSetVel 445
AxmHomeGetVel 448
AxmHomeSetStart 451
AxmHomeSetResult 453
AxmHomeGetResult 458
AxmHomeGetRate 463

위치구동 함수 468
AxmMoveStartPos 470
AxmMovePos 473
AxmMoveVel 475
AxmMoveStartMultiVel 477
AxmMoveSignalSearch 480
AxmMoveSignalCapture 483
AxmMoveGetCapturePos 486
AxmMoveStartMultiPos 488
AxmMoveMultiPos 491
AxmMoveStop 494
AxmMoveEStop 496
AxmMoveSStop 498

위치 및 속도 오버라이드 구동 함수 500
AxmOverridePos 501
AxmOverrideSetMaxVel 503
AxmOverrideVel 505
AxmOverrideAccelVelDecel 507
AxmOverrideVelAtPos 509

MASTER, SLAVE 구동 함수 510
AxmLinkSetMode 512
AxmLinkGetMode 514
AxmLinkResetMode 516

보간 구동 함수(Line, Circle) 518
AxmLineMove 520
AxmCircleCenterMove 524
AxmCirclePointMove 529
AxmCircleRadiusMove 534

AxmCircleAngleMove 539
AxmContiSetAxisMap 544
AxmContiGetAxisMap 548
AxmContiSetAbsRelMode 551
AxmContiGetAbsRelMode 554
AxmContiIsMotion 557

Trigger 함수 561
AxmTriggerSetTimeLevel 562
AxmTriggerGetTimeLevel 566
AxmTriggerSetAbsPeriod 570
AxmTriggerGetAbsPeriod 573
AxmTriggerSetBlock 575
AxmTriggerGetBlock 578
AxmTriggerOneShot 580
AxmTriggerOnlyAbs 582
AxmTriggerSetReset 585

겐트리 구동 함수 587
AxmGantrySetEnable 597
AxmGantryGetEnable 601
AxmGantrySetDisable 605

CRC(잔여 펄스 클리어) 함수 607
AxmCrcSetMaskLevel 608
AxmCrcGetMaskLevel 611
AxmCrcSetOutput 614
AxmCrcGetOutput 616

MPG 구동 함수 618
AxmMPGSetEnable 622

AxmMPGGetEnable 625
AxmMPGSetRatio 628
AxmMPGGetRatio 631
AxmMPGReset 634

네트웍크 상태에 대한 모션 구동 에러 처리 636
AxmNetWorkErrorSetAction 637
AxmNetWorkErrorGetAction 639

에러코드 테이블(ERROR CODE TABLE) 확인 641

Motion 라이브러리 사용법

모션라이브러리 개요

모션제어 개요

모션라이브러리 개요

초기화 및 종료

라이브러리의 초기화

라이브러리의 종료

모션 파라미터

모션 파라미터의 설정

모션신호

모션신호 설정

범용 입출력 신호의 읽기 및 출력 쓰기

설정 정보의 저장 및 불러오기

모션 구동

모션구동 개요

모션구동의 용어해설

단축 구동

다축구동

확장 모션구동

CRC 신호 설정

오버라이드 구동

MPG 구동 (Manual Pulse Generation)

동기구동

갠트리 구동

1) 겐트리 구동 설정 및 해제

2) 갠트리 구동에서의 홈 검색
가상축 매핑
트리거 신호의 사용

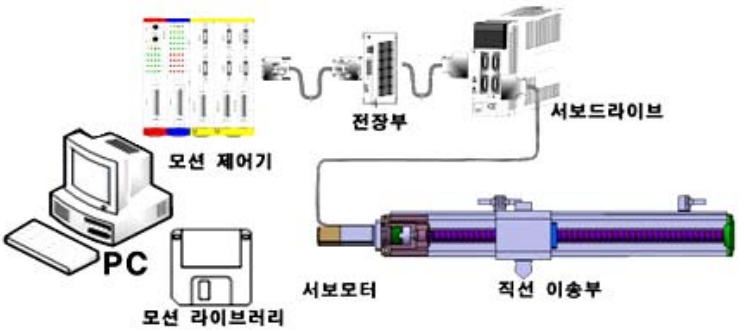
모션라이브러리 개요

모션제어 개요
모션라이브러리 개요

모션제어 개요

반도체 등 정밀한 부품을 조립 또는 검사하는 장비나 공작물을 가공하는 공작기계 등의 주요구성 요소를 살펴보면 실제 운동을 일어나게 하는 모션 제어부 (이송시스템)이라고 말 할 수 있다. 여기서 모션이란 우리말로 풀이하자면 운동을 뜻한다. 어떠한 물체가 한 위치에서 다른 위치로 이동하는 것을 모션이라고 설명할 수 있을 것이다. 그리고, 제어란 한마디로 조종의 개념이다. 우리가 원하는 방향과 위치로 대상물을 움직이게 하는 것을 뜻하는 것이다. 결국 모션제어란 앞에서 설명한 모션과 제어의 조합으로서 어떠한 물체를 우리가 움직이고자 하는 방향으로 이동시키는 것이라고 정의할 수 있다. 모션제어를 하기 위해서는 아래 그림에서 보는 바와 같은 구성품이 필요하다.

- 이송부 : 운동이 발생하는 기구부
- 서보모터 : 동력을 발생하는 장치
- 서보드라이브 : 서보모터에 전류를 흘려주는 장치
- 모션제어기 : 모션을 제어하기 위한 각종 신호를 발생시키는 장치
- 전장부 : 모션제어기와 서보드라이브등을 연결하기 위한 단자대외 기타 연결장치
- PC : PC프로그래밍을 통해 모션제어기를 제어하는 최상위 장치
- 모션 라이브러리 : 모션제어기를 구동하는 함수



모션라이브러리 개요

최근 장비프로그래밍은 MS Windows 상에서 프로그래밍 하는 윈도우 프로그래밍이 주종을 이루고 있다. 프로그래밍 툴(Tool)에는 Visual C++, C++ Builder, Delphi, Visual Basic 등의 다양한 툴들이 제공되고 있으며, AXL 에서는 이러한 모든 툴들에서 활용이 가능한 API 함수들을 제공하고 있다. 특히 모션구동에 대한 라이브러리 함수들은 일반적인 기구부 의 기계적인 신호 및 범용입출력 신호에 대한 세부적인 제어에서부터 단축 및 다축 보간구동기능에 관한 함수까지 매우 세분화된 함수들을 제공하고 있어 사용자가 필요로 하는 모든 모션제어에 활용할 수 있도록 되어 있다.

본 User Guide 에서는 여러 모션제어용 모듈 중 단축 및 다축구동 뿐만 아니라 보간 기능까지 지원하는 4 Axis Motor Control LSI 가 들어있는 EIP-M2Q, EIP-M4Q 모션제어을 구동하기 위한 라이브러리 함수들을 사용하여 설명하고 있다. EIP-M2Q, EIP-M4Q 모션 라이브러리에서는 Stepping 및 Servo Motor 등을 정밀 제어할 수 있다. 따라서 이 칩을 이용하여 CPU의 부하를 감소시키면서 연속적인 모션제어를 구현할 수 있으며, 원점검출, 속도 오버라이드 및 이동량 오버라이드, 직선 보간, 원호 보간을 직선 또는 S-curve 가감속 프로파일로 구동할 수 있다.

기초 모션구동

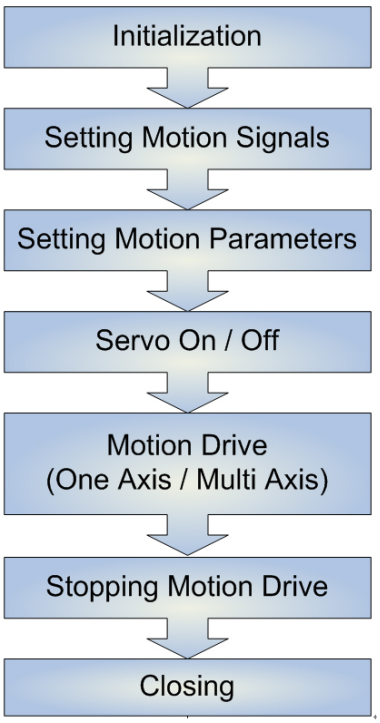
- 라이브러리 초기화 및 종료
- 모션 파라미터의 설정
- 모션 신호의 설정
- 범용 입출력 신호의 활용
- 단축구동
  - 위치구동 (Position Drive)
  - 속도구동 (Velocity Drive)
  - 신호검색구동 (Signal Search Drive)
  - 홈서치구동(Home serarch)
- 다축구동
  - 다축 위치구동 (Multi Axes Position Drive)
  - 다축 보간구동 (Interpolation Drive)

고급 모션구동

- CRC 신호 설정
- 오버라이드구동 (Override Drive)
- MPG 구동
- 동기구동
- Dynamic Link
- 갠트리 구동
- 가상축 매핑

• 트리거 신호의 사용

기본적인 모션구동은 초기화 및 종료, 입출력신호, 파라미터 설정 및 확인, 서보 On/Off, 신호 검색구동, 단축 위치구동, 단축 속도구동, 구동상태 및 정지상태 모니터링, 단축구동의 정지로 크게 나눌 수 있으며 아래와 같은 순서로 이루어 진다. 각 장마다 그 장에 필요한 하드웨어의 개념과 기본함수, 그리고 기본함수의 예제를 기초로 하여 설명한다.



초기화 및 종료

AXL 라이브러리의 초기화 작업은 AxlOpen함수를 실행함으로써 이루어진다. AxlOpen함수를 실행하면, 내부적으로 라이브러리의 초기화뿐만 아니라 설치되어 있는 현재 시스템과 연결되어 있는 모든 노드를 초기화 하게 된다. 그러므로 사용자는 모션 라이브러리를 사용하기 전에 반드시 라이브러리 초기화 작업을 수행하여야 한다.

라이브러리의 초기화
라이브러리의 종료

라이브러리의 초기화

라이브러리의 초기화

AxlOpen : AxlOpen 함수는 라이브러리를 초기화 하는 함수이다.
이 함수는 리턴값으로 에러코드를 리턴하므로 아래에서처럼 사용할 수 있다.

```
// 라이브러리를 초기화 한다.
if (AxlOpen() == AXT_RT_SUCCESS)
    printf("라이브러리가 초기화 되었습니다.");
```

노드에 장착된 모듈 초기화

AxlInit : AxlInit 함수는 지정하는 노드에 장착된 모듈을 초기화하는 함수이다.

```
// 라이브러리가 초기화 되어 있는지 확인한다.
if (AxlInit(lNodeNum) == AXT_RT_SUCCESS)
    printf("지정 노드의 모듈들이 초기화 되었습니다.");
else
    printf("지정 노드의 모듈 초기화에 실패하였습니다.");
```

라이브러리 초기화 여부의 확인

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

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

라이브러리 및 노드 정보 확인

AxlGetNodeCount: AxlGetNodeCount함수는 등록된 노드 개수를 확인하는 함수이다.

```
//등록된 노드 개수 확인
long lpNodeCount;
AxlGetNodeCount(&lpNodeCount);
printf("등록된 노드 개수는 : %d", lpNodeCount);
```

AxlGetNodeInfo : 함수는 지정 노드에 장착된 기능 모듈 정보를 확인하는 함수이다.
한 노드에 장착될 수 있는 최대 모듈 개수는 17개로 제한 된다. 주의) 장착된 모듈 개수보다 큰 Arrary로 변수를 선언하여야 한다.

```
// 노드 번호에 따른 모듈 정보를 확인
long NodeNum=0;
DWORD dwpModuleId[17];
 AxlGetNodeInfo(NodeNum, dwpModuleId);
printf("0축의 모듈 ID는 %x", dwpModuleId[0], dwpModuleId[1], dwpModuleId[2],
dwpModuleId[3], dwpModuleId[4]);
```

AxlGetNetInfo : AxlGetNetInfo함수는 등록된 노드 정보를 확인하는 함수이다.
등록된 노드 정보는 구조체로 제공된다.

```
// 등록된 노드 정보를 구조체로 제공
AXT_EipNetInfo* tagEioNetInfo;
 AxlGetNetInfo(AXT_EipNetInfo* tagEioNetInfo);
prinff("노드 카운터:%s, 노드 ID: %s", tagEioNetInfo.ucNodeCount, tagEipNetInfo.uNodeID);
```

AxlGetLibVersion : AxlGetLibVersion함수는 라이브러리 버전을 확인하는 함수이다

```
// 라이브러리 버전 확인
char szVersion; AxlGetLibVersion(&szVersion);
   printf("라이브러리 버전은: %s", szVersion);
```

모션모듈의 존재 여부 확인

AxmInfoIsMotionModule : AxmInfoIsMotionModule 함수는 실제 모션모듈을 장착한 노드가 존재하는가 여부를 확인하는 함수이다. 모션관련 함수를 사용하기 전에 모션모듈이 있는지를 먼저 확인하기 위해 사용한다. 모듈이 존재하는지를 확인하기 위해 아래와 같이 한다.

```
// 모션모듈이 있는지 확인한다.
DWORD dwStatus;
AxmInfoIsMotionModule(&dwStatus);
if(dwStatus == STATUS_EXIST)
   printf("모션모듈이 존재합니다.");
else
   printf ("모션모듈이 존재하지 않는다.");
```

시스템에 장착된 축의 수 및 정보 확인

AxmInfoGetAxisCount : AxmInfoGetAxisCount 함수는 현재 시스템과 연결된 노드들에 장착되어 있는 모든 축의 개수를 확인하는 함수이다.

```
//시스템에 장착된 축 개수를 확인
long lAxisCount;
AxmInfoGetAxisCount(&lAxisCount);
printf("시스템에 장착된 축 개수 : %d",lAxisCount);
```

AxmInfoGetFirstAxisNo : AxmInfoGetFirstAxisNo 함수는 지정한 노드의 지정 모듈에서 첫 번째 축의 라이브러리내 할당 축번호를 확인하는 함수이다.

```
//0번 노드의 1번 모듈에서 시작 축 번호를 확인한다.
long lNodeNo = 0;
long lModulePos = 1;
long lFirstAxisNo;

AxmInfoGetFirstAxisNo(lNodeNo, lModulePos, &lFirstAxisNo);
printf("0번 노드의 Module ID설정이 0인 모듈의 시작 축 번호 : %d", lFirstAxisNo);
```

AxmInfoGetAxis : AxmInfoGetAxis 함수는 지정한 축의 노드번호, 모듈위치, 모듈 ID를 확인하는 함수이다.

```
AxmInfoGetAxis : AxmInfoGetAxis 함수는 지정한 축의 노드번호, 모듈위치, 모듈 ID를 확인하는 함수
이다. //0번 축의 노드번호, 모듈위치, 모듈 ID를  확인한다
long lAxisNo = 0;
long lpNodeNo, lModulePos;
DWORD dwModuleID;

AxmInfoGetAxis(lAxisNo, &lpNodeNo, &lModulePos, &dwModuleID);
printf(("0번 축의 노드번호 : %x, 모듈위치 : %x, 모듈 ID : %x",
lpNodeNo, lModulePos, dwModuleID);
```

라이브러리의 종료

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.

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

```
// Ex1_AXM_InitAndClose.cpp : Defines the entry point for the console application.
// 라이브러리를 초기화하고, 축 정보를 확인한 후 종료한다.

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

void main(void)
{
   // 라이브러리를 초기화 한다.
   DWORD Code = AxlOpen();
   if (Code == AXT_RT_SUCCESS)
   {
         printf("라이브러리가 초기화 되었습니다.₩n");

         //모션모듈이 있는지 검사
         DWORD dwStatus;
```

```
Code = AxmInfoIsMotionModule(&dwStatus);
if(Code == AXT_RT_SUCCESS)
{
    if(dwStatus == STATUS_EXIST)
    {
        printf("모션모듈이 존재 합니다.₩n");

        //시스템에 장착된 축 개수를 확인
        long lpAxisCount;
        AxmInfoGetAxisCount(&lpAxisCount);
        printf("시스템에 장착된 축 개수 : %d ₩n",lpAxisCount);
        //0번 축의 노드번호, 모듈위치, 모듈 ID를  확인한다
        long lNodeNo, lModulePos;
        long lAxisNo = 0;
        DWORD dwModuleID;
        AxmInfoGetAxis(lAxisNo, &lpNodeNo, &lModulePos, &dwModuleID);
        printf("0번 축의 노드번호 : %x, 모듈위치 : %x, 모듈 ID : %x₩n",
            lpNodeNo,lModulePos,dwModuleID);

        //0번 노드의 1번 모듈에서 시작 축 번호를 확인한다.
        long lFirstAxisNo;
        AxmInfoGetFirstAxisNo(lNodeNum, lModulePos, &lFirstAxisNo);
        printf("%d번 노드의 %d번째 모듈의 시작 축번호 : %d₩n",
            lNodeNum,lModulePos,lFirstAxisNo);

    }
    else
        printf ("AxmInfoIsMotionModule() : ERROR ( NOT STATUS_EXIST )
            code 0x%x₩n",Code);
```

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

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

모션 파라미터

모션 파라미터라고 함은 모션 시스템을 이루는 각 부분의 올바른 설정을 위한 설정값이라고 할 수 있다. 모션 시스템은 앞에서도 언급했듯이 기구부와 서보드라이버 및 서보모터, 전장부, 모션제어기등 여러 가지 장치들이 조합되어 있다. 당연히 각 구성요소들은 모두 각각의 설정 파라미터들을 가지고 있다. 그러므로 하나만 설정이 잘못되어도 전체 시스템은 오동작을 하거나 에러를 발생시키게 된다.

모션제어기에서 설정해주는 모션 파라미터는 다음과 같으며, 각각의 의미를 간단하게 살펴보면 다음과 같다.

Unit per Pulse

- 명령에서 내려주는 지령 값의 단위를 설정해주는 값이다.

Start / Stop Speed

- 구동부의 시작 및 종료시의 속도를 설정한다.

Pulse Out Method

- 모션제어기에서 서보드라이브로 출력하는 펄스의 형태를 설정한다.

Encoder Input Method

- 서보드라이브에서 모션제어기로 보내주는 엔코더신호를 받아들이는 방식을 설정한다.

모션 파라미터의 설정

모션 파라미터의 설정

모션제어기의 파라미터 또한 전체 시스템에서의 각 요소들이 가지는 설정 및 특성들과 연관하여 설정하게 된다. 그러므로 본문에서 설명하는 각 파라미터의 값을 설정하기 위한 전제조건들을 주의 깊게 살펴보고, 사용자가 가지고 있는 시스템과 비교하여 알맞게 조절하여 설정하여야 한다.

1) Unit per Pulse

기본적으로 사용자가 라이브러리의 명령어에서 지정해주는 이동거리 지령값은 아래의 그림에서와 같이 두 번의 단위변환 과정을 거치게 된다.

소프트웨어 상에서 이동량을 입력하면, 설정된 Unit/Pulse라는 파라미터에 의해 모션 제어기의 출력펄스 수가 결정된다. 그리고 이 출력펄스는 서보드라이브에 설정되어있는 전자기어비라는 파라미터에 의해 또 한번 변환되어 실제 모터를 구동시키는 구동펄스 수가 결정된다.
여기서 Unit/Pulse는 모션 구동함수를 실행함에 있어서 주어지는 값들의 단위를 지정해주기 위한 파라미터이며, 이 값을 설정하는 함수가 AxmMotSetMoveUnitPerPulse 이다. 쉽게 말해서

Unit에 10을 설정하고, Pulse에 100을 설정한다면, 모션구동함수에서 10을 명령하면 100개의 펄스가 서보드라이브로 출력된다는 뜻이다. 예를 들어 서보모터가 한 바퀴 회전할 때 10mm 움직이는 기구부에서 모션제어기는 서보모터를 1회전 시키기 위해 10,000펄스를 출력한다고 할 때, 프로그램상에서 주어지는 지령값을 mm단위로 설정하고자 한다면 다음과 같이 한다.

```
//0축 지령값을 mm단위가 되도록 한다.
long lAxisNo = 0;
long lPulse = 10000;      // 1회전을 위한 펄스 수
double dUnit = 10;        // 1회전시 이동거리
AxmMotSetMoveUnitPerPulse(lAxisNo, dUnit, lPulse);
```

프로그램상의 지령값을 Pulse단위로 설정하고자 한다면 다음과 같이 한다.

```
//0축 지령값을 Pulse단위가 되도록 한다.
long lAxisNo = 0;
long lPulse = 1;
double dUnit = 1;
AxmMotSetMoveUnitPerPulse(lAxisNo, dUnit, lPulse);
```

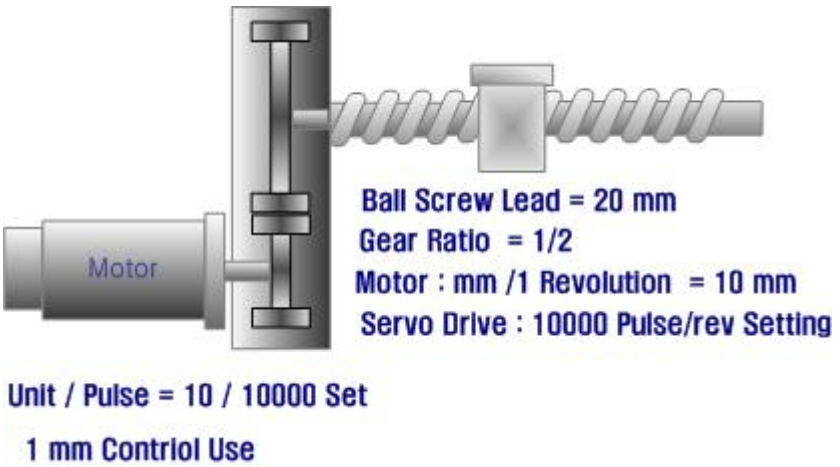
※ Electronic gear Ratio (전자기어비)

Unit/Pulse는 모션제어기에서 설정하는 파라미터인데 반하여 전자기어비는 서보드라이브에서 설정하는 파라미터이다.

$$[\text{전자 기어비}] = \frac{[\text{서보모터가 1회전 할 때 발생하는 엔코더 펄스 수}]}{[\text{서보모터가 일회전 할 때 모션제어기의 출력지령펄스 수}]}$$

17비트의 엔코더를 사용한다고 하면, 위의 식으로 131,072개의 엔코더의 펄스 수는 모션제어기에서 10,000로 줄일 수 있다. 즉 서보모터 엔코더의 회전 정밀도를 기구부의 정밀도 만큼으로 낮추는 것이 전자 기어비 이다.

예를 들어 20mm 볼 스크루, 기어비 : 1/20이고 모터 1회전에 필요한 펄스 수가 10000 일때, lPulse = 10000과 dUnit = 10으로 설정하면 1mm 움직이는 거리 당 출력되는 펄스 수 = 1000펄스가 설정된다



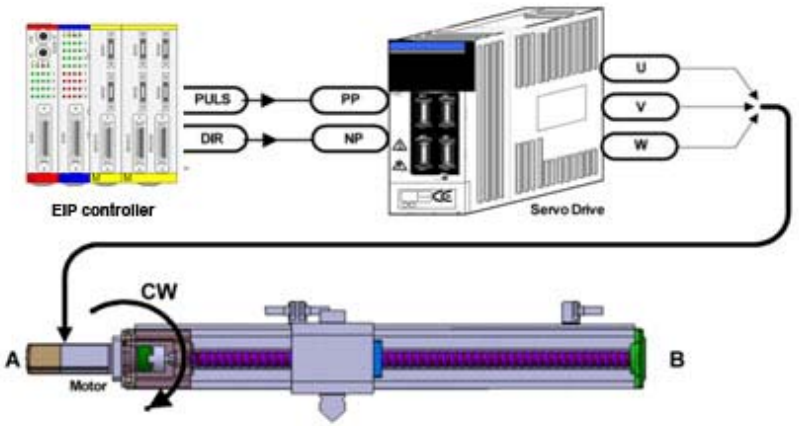
2) Start Speed

모션이 시작될 때의 초기 속도를 지정한다. 초기속도는 모션구동의 기준점이 되므로 반드시 설정하여야 하며, 기본값은 1로 설정되어 있다. AxmMotSetMinVel 라는 함수를 이용하여 다음과 같이 설정한다.

```
//0축에 초기 속도를 1로 설정한다.    Default : 1
long lAxisNo = 0;
double dMinVelocity = 1;
AxmMotSetMinVel(lAxisNo, dMinVelocity);
```

3) Pulse Out Method

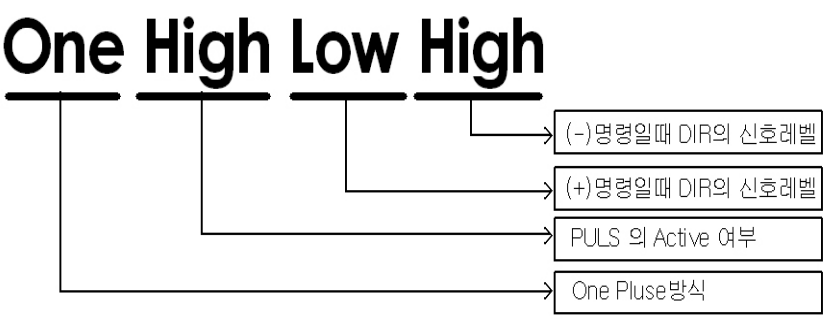
사용자가 지령하는 구동명령은 아래의 그림에서와 같이 모션제어기가 지령값에 의해 서보드라이브로 펄스를 전송하고, 서보드라이브는 다시 서보모터를 구동시키는 구조로 되어 있다.
모션제어기는 PULS핀과 DIR핀을 통해서 서보드라이브로 펄스를 출력하고, 서보드라이브의 PP핀과 NP핀에서 그 펄스를 받아들인다고 하면, 두 핀에서 들어온 펄스신호를 바탕으로 해당 펄스만큼 서보모터를 구동시킨다. 여기서 서보모터를 어느 방향으로 어느 핀에서 들어온 펄스 수만큼 회전시켜야 하는지를 결정해야 하는데, 모션제어기에서 설정한 방식대로 서보드라이브 쪽도 동일하게 설정해야 정상 작동한다.

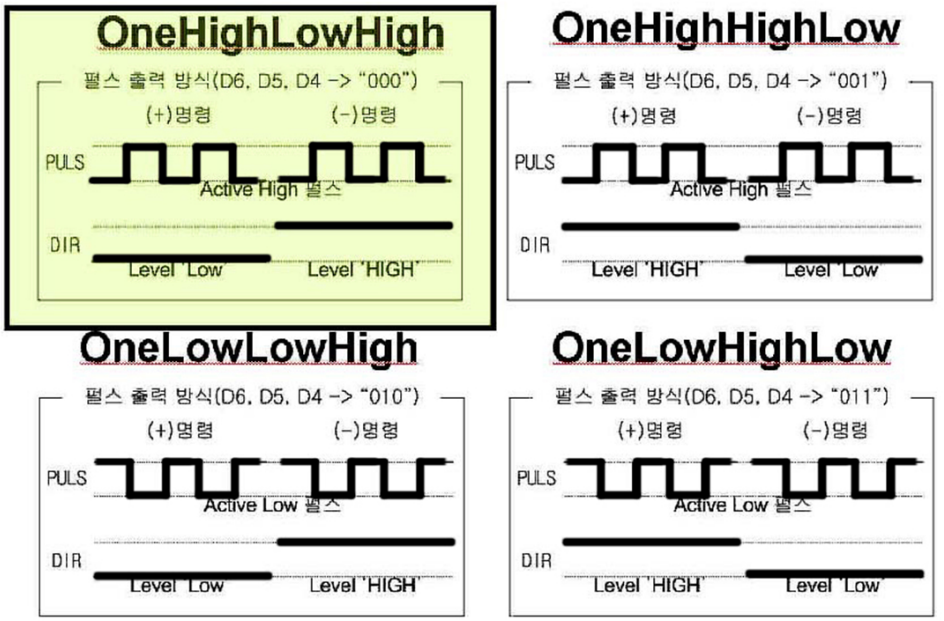


서보드라이브로 출력되는 모션제어기의 PULS, DIR의 펄스 출력방식은 One Pulse방식과 Two Pulse방식으로 구분한다.

One Pulse 방식

One Pulse방식에서는 모션제어기의 PULS핀에서 회전량에 해당하는 개수의 펄스를 출력하고 DIR핀에서는 회전방향에 해당하는 신호레벨을 출력한다. 아래 그림에서 보는 바와 같이 1펄스 출력방식은 PULS신호의 Active여부와 (+) 방향의 명령을 주었을 때 DIR의 신호레벨과 (-) 명령을 주었을 때 DIR의 신호레벨을 설정하는 부분으로 구성되어 있다. 서보 드라이브와 서보 모터에 따라 조금씩의 차이가 있으므로, 해당 서보드라이브에서 어떠한 방식으로 설정되어 있는지에 주의하여 모션제어기의 펄스출력방식을 설정하여야 한다.

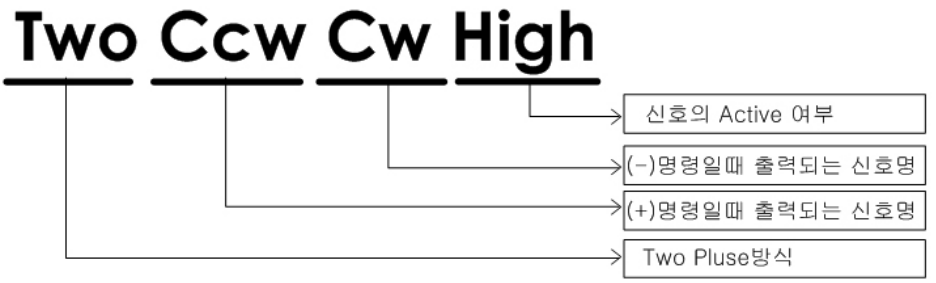


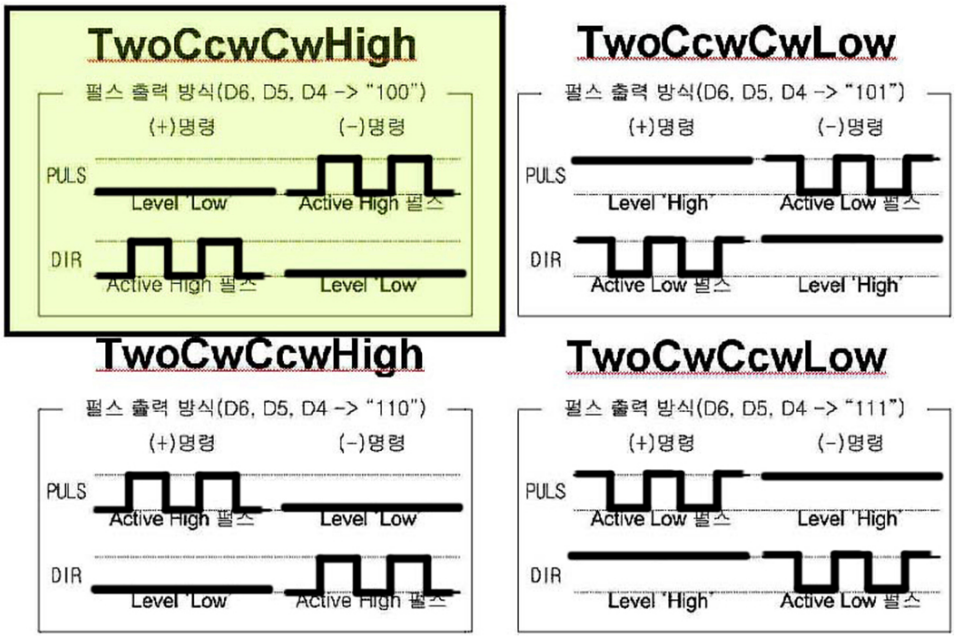


종 류	설 명
OneHighLowHigh	1펄스 방식, PULSE(Active High) +방향(DIR=Low) / -방향(DIR=High)
OneHighHighLow	1펄스 방식, PULSE(Active High) +방향(DIR=High) / -방향(DIR=Low)
OneLowLowHigh	1펄스 방식, PULSE(Active Low) +방향(DIR=Low) / -방향(DIR=High)
OneLowHighLow	1펄스 방식, PULSE(Active Low) +방향(DIR=High)/ -방향(DIR=Low)

Two Pulse 방식

Two pulse방식은 PULS핀에서 나오는 펄스 신호와 DIR핀에서 나오는 펄스신호가 각각 회전량과 회전방향을 동시에 표현하도록 구성된 출력방식이다. 아래 그림에서 보는 바와 같이 2펄스 출력방식은 (+) 방향의 명령을 주었을 때 신호가 출력될 핀을 설정하는 부분과, (-) 방향의 명령을 주었을 때 신호가 출력될 핀을 설정하는 부분 및 신호의 Active 여부로 구성되어 있다. 여기서 Cw는 PULS핀을, Ccw는 DIR핀을 의미한다.

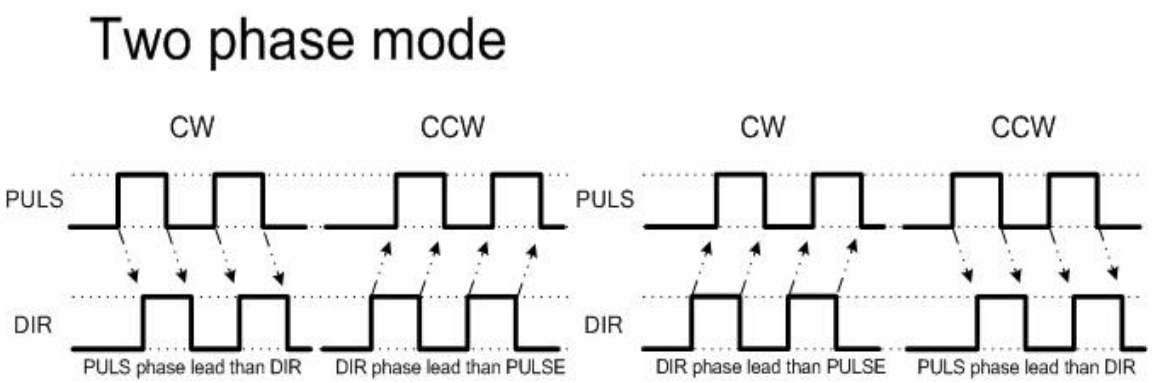




종 류	설 명
TwoCcwCwHigh	2펄스 방식, PULSE(CW:-방향) DIR(CCW:+방향), Active High

TwoCcwCwLow	2펄스 방식, PULSE(CW:-방향) DIR(CCW:+방향), Active Low
TwoCwCcwHigh	2펄스 방식, PULSE(CW:+방향) DIR(CCW:-방향), Active High
TwoCwCcwLow	2펄스 방식, PULSE(CW:+방향) DIR(CCW:-방향), Active Low

Two Phase 방식



출력펄스를 2상 4체배 방식으로 출력방식을 설정할 수 있는데 제어펄스 최대주파수를 1/4로 출력할수있다.

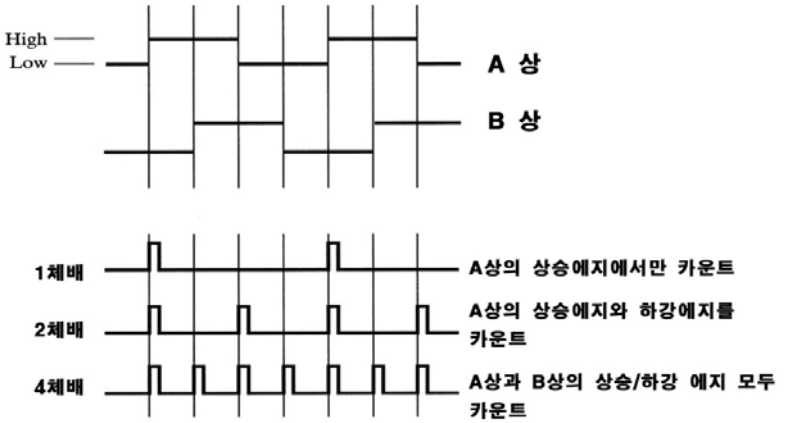
펄스 출력방식의 설정

사용자는 자신의 기구부와 서보드라이브의 성격에 맞도록 펄스 출력 방식을 선택하고, AxmMotSetPulseOutMethod 함수를 이용하여 지정 축에 펄스 출력방식을 설정할 수 있다. 예를 들어, 서보드라이브는 PP핀에 펄스가 입력되면 서보모터를 정 방향으로 회전하고 기구부는 (-)방향으로 테이블이 움직이고, NP에 펄스가 입력되면 그 반대라고 하자. 프로그램상에서 명령을 내릴 때 (+)명령 시 모션제어기의 DIR핀에서 펄스가 출력되어 서보드라이브의 NP핀으로 입력되고, (-)명령 시는 반대로 적용되도록 펄스출력방식을 설정해야 할 것이다. 각각의 핀에 입력되는 펄스에 따라 작동하므로 2펄스 방식이고, (+)명령 시 DIR핀에서 펄스가 출력되고, (-)명령 시 PULS핀에서 펄스가 출력되어야 하므로 CcwCw, 그리고 High Active로 설정하게 되므로, TwoCcwCwHigh로 설정한다.

```
//0축의 펄스 출력 방식을 TwoCwCcwHigh로 한다.          Default : OneLowHighLow
long lAxisNo = 0;
DWORD uMethod = TwoCwCcwHigh;
AxmMotSetPulseOutMethod(lAxisNo, uMethod);
```

4) Encoder Input Method

서보모터가 회전을 하면, 구동축에 연결되어 있는 엔코더에서 회전량에 따른 펄스가 출력된다. 엔코더는 일반적으로 A상과 B상, 그리고 Z상의 세 종류의 펄스가 출력되는데, A상과 B상은 90°의 위상차를 가진다. 즉 모터가 정 방향으로 회전할 때에 A 상이 먼저 발생하고, 이어서 90° 지연되어 B상의 펄스가 나타나는 것이다. 또한 Z상은 모터가 한 바퀴 회전할 때마다 발생하는 펄스이다. 이러한 펄스를 카운트하여 모터의 회전량을 피드백 받을 수 있다.
엔코더의 A상과 B상을 이용하여 기존의 주파수의 배수에 해당하는 신호를 만들어낼 수 있는데, 이러한 신호 획득 방법을 "체배"라고 한다. 그림에서 A는 2체배를 B는 4체배를 보여주고 있다.



Define	의미
ObverseUpDownMode	정방향 Up/Down
ObverseSqr1Mode	정방향 1체배
ObverseSqr2Mode	정방향 2체배
ObverseSqr4Mode	정방향 4체배
ReverseUpDownMode	역방향 Up/Down
ReverseSqr1Mode	역방향 1체배
ReverseSqr2Mode	역방향 2체배
ReverseSqr4Mode	역방향 4체배

```
//지정 축에 Encoder 입력 방식을 4체배로 설정한다.
long lAxisNo = 0;
DWORD uMethod = ObverseSqr4Mode;
AxmMotSetEncInputMethod(lAxisNo, uMethod);
```

```
// Ex2_AXM_SettingParameter.cpp : Defines the entry point for the console application.
// 라이브러리를 초기화하고, 축 정보를 확인한 후 종료한다.

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

void main(void)
{
    // 라이브러리를 초기화 한다.
    DWORD Code = AxlOpen();
    if (Code == AXT_RT_SUCCESS)
    {
        printf("라이브러리가 초기화 되었습니다.₩n");

        //모션모듈이 있는지 검사
        DWORD dwStatus;
        Code = AxmInfoIsMotionModule(&dwStatus);
        if(Code == AXT_RT_SUCCESS)
        {
            if(dwStatus == STATUS_EXIST)
            {
```

```
printf("모션모듈이 존재 합니다.₩n");

//0축 지령값을 mm단위가 되도록 한다.
long lAxisNo = 0;
long lPulse = 10000;
double dUnit = 10;
AxmMotSetMoveUnitPerPulse(lAxisNo, dUnit, lPulse);
//설정된 값을 확인한다.
AxmMotGetMoveUnitPerPulse(lAxisNo, &dUnit, &lPulse);
printf("UnitPerPulse 설정 => Unit : %d, Pulse : %f ₩n", dUnit, lPulse);

//0축에 초기 속도를 1로 설정한다. Default : 1
double dMinVelocity = 1;
AxmMotSetMinVel(lAxisNo, dMinVelocity);
//설정된 값을 확인한다.
AxmMotGetMinVel(lAxisNo, &dMinVelocity);
printf("MinVelocity 설정 => %2f ₩n", dMinVelocity);

//0축의 펄스 출력 방식을 TwoCwCcwHigh로 한다.
//   Default : OneLowHighLow
DWORD uMethod = TwoCcwCwHigh;
AxmMotSetPulseOutMethod(lAxisNo, uMethod);

//설정된 값을 확인한다.
AxmMotGetPulseOutMethod(lAxisNo, &uMethod);
printf("PulseOutMethod 설정 => ");
switch(uMethod)
{
    case OneHighLowHigh:
```

```
            printf("OneHighLowHigh ₩n");   break;
            case OneHighHighLow:
            printf("OneHighHighLow ₩n");   break;
            case OneLowLowHigh:
            printf("OneLowHighLow ₩n");    break;
            case OneLowHighLow:
            printf("OneLowHighLow ₩n");    break;
            case TwoCcwCwHigh:
            printf("TwoCcwCwHigh ₩n");    break;
            case TwoCcwCwLow:
            printf("TwoCcwCwLow ₩n");    break;
            case TwoCwCcwHigh:
            printf("TwoCwCcwHigh ₩n");    break;
            case TwoCwCcwLow:
            printf("TwoCwCcwLow ₩n")      break;
        }

        //0축에 Encoder 입력 방식을 4체배로 설정한다.
        uMethod = ObverseSqr4Mode;
        AxmMotSetEncInputMethod(lAxisNo, uMethod);
        //설정된 값을 확인한다.
        AxmMotGetEncInputMethod(lAxisNo, &uMethod);
        printf("PulseOutMethod 설정 => ");
        switch(uMethod)
        {
            case ObverseUpDownMode:
            printf("ObverseUpDownMode ₩n");   break;
            case ObverseSqr1Mode:
            printf("ObverseSqr1Mode ₩n");        break;
```

```
                    case ObverseSqr2Mode:
                     printf("ObverseSqr2Mode ₩n");        break;
                    case ObverseSqr4Mode:
                     printf("ObverseSqr4Mode ₩n");        break;
                    case ReverseUpDownMode:
                     printf("ReverseUpDownMode ₩n");   break;
                    case ReverseSqr1Mode:
                     printf("ReverseSqr1Mode ₩n");        break;
                    case ReverseSqr2Mode:
                     printf("ReverseSqr2Mode ₩n");        break;
                    case ReverseSqr4Mode:
                     printf("ReverseSqr4Mode ₩n");      break;
                }
            }
            else
                printf ("AxmInfoIsMotionModule() : ERROR ( NOT STATUS_EXIST )
                     code 0x%x₩n",Code);
        }
        else
            printf ("AxmInfoIsMotionModule() : ERROR ( Return FALSE )
                 code 0x%x₩n",Code);
    }
        else
            printf ("AxlOpen() : ERROR    code 0x%x₩n",Code);

    // 라이브러리를 종료 한다.
    if (AxlClose())
        printf("라이브러리가 종료 되었습니다.₩n");
    else
```

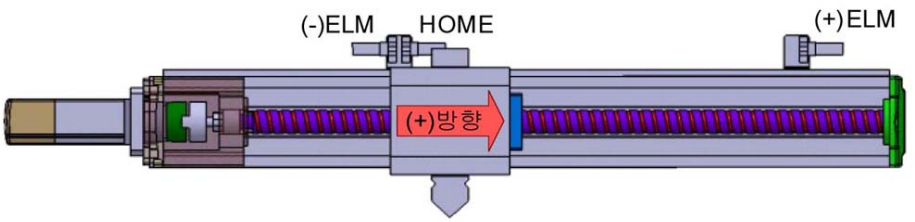
```
        printf("라이브러리가 정상적으로 종료되지 않았습니다.₩n");

}
```

모션신호

Signal은 크게 Motion Signal, Universal Input Signal, Universal Output Signal로 나눌 수 있다. Motion Signal은 기구부의 구동에 대한 상태를 알려주는 각종 센서와 서보드라이브의 상태를 알려주는 신호들로 구성되어 있고, Universal Input Signal은 사용자가 임의로 사용 가능한 입력신호들이며, Universal Output Signal은 사용자가 임의로 사용이 가능한 출력 신호들이라고 할 수 있다. 여기서 범용 입/출력 신호(Universal Input/Output Signal)중 상위 2비트는 이미 모션제어기에서 특별한 기능으로 사용되고 있으므로, 실제 사용자가 임의로 사용 가능한 비트는 Input, Output 각각 3개씩이다.

1) Motion Signal

Motion Signal은 기구부의 구동에 대한 상태를 알려주는 각종 센서와 서보드라이브의 상태를 알려주는 신호들을 일컫는다.



End Limit

기구부의 이송구역을 지정하는 센서에서 발생되어 모션제어기로 전달되는 신호로서 아래의 그림에서와 같이+,-로 구분되어져 기구부 양쪽에 설치되어 있다. 일반적으로 기구부의 이송 중 리미트 신호가 발생하면 모션제어기는 펄스 출력을 바로 중단하고 기구부는 급정지 하게 된다.

Inposition

서보드라이브의 출력신호로 서보모터의 위치가 서보드라이브에 설정해둔 인포지션 범위값 내에 들어오면 서보드라이브는 Inposition 신호를 모션제어기에 전달한다. 즉 목표위치 도달여부 확인용 신호라고 할 수 있다.

Alarm

서보드라이브의 출력신호로 서보모터에 이상전류가 발생하거나 작동오류가 생겼을 경우 서보드라이브에서 모션제어기에 Alarm 신호를 전달한다.

Emergency Stop

모션제어기에 이 신호가 입력되면 모션제어기는 펄스 출력을 바로 중단하고, 기구부는 급정지한다. 이 신호는 보통 장비에 설치된 비상 스위치와 연결되어 있다.

2) Universal Input Signal

사용자가 임의로 활용 가능한 6비트의 범용 입력을 지원한다. 사용자는 이러한 범용 입력비트를 사용하여 추가적으로 센서등의 신호를 입력 받아 활용할 수 있다.

3) Universal Output Signal

사용자가 임의로 활용 가능한 6비트의 범용 출력 또한 지원한다. 사용자는 이러한 범용 출력비트를 사용하여 신호출력 또는 엑츄에이터의 작동스위치 등으로 활용할 수 있다.

모션신호 설정
범용 입출력 신호의 읽기 및 출력 쓰기
설정 정보의 저장 및 불러오기

모션신호 설정

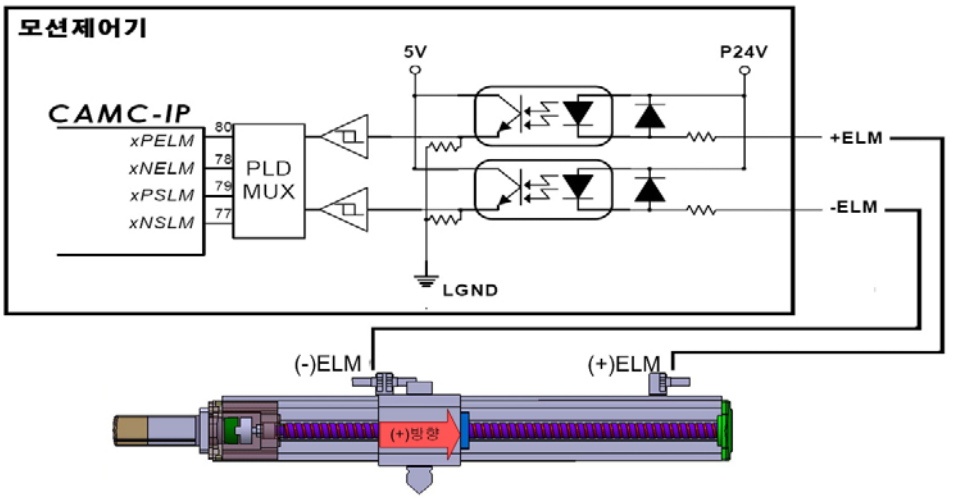
1) 신호레벨 및 사용유무 설정

각 기구부 센서의 특성에 따라 다른 Signal이 발생하는데, 그 신호의 특성에 맞게 Active Level을 설정하여야 한다. 설정 값으로는 LOW(0), HIGH(1), UNUSED(2), USED(3)가 있고, LOW또는 HIGH의 레벨을 설정하면 자동으로 USED 라고 판단한다. 예를 들어 센서가 물체를 감지할 때 모션제어기 내부신호가 HIGH가 된다면, ActiveHIGH로 설정하게 된다. 각각의 신호를 사용하지 않음으로 설정하면, 신호가 입력되더라도 관여하지 않게 된다. 사용자가 기구부의 센서의 설치유무를 파악하여 설정하면 된다.

● *End Limit 신호*

모션제어기는 내부 회로를 보호하기 위해 포토커플러를 이용하여 외부에서 들어오는 신호(24V)

와 내부신호(5V)가 전기적으로 절연되어 있다.
아래의 그림에서 보는 바와 같이 Limit 센서가 B접점 센서로 평소에는 HIGH신호를 내보내고 이송부를 감지하여 LOW신호를 발생시키면, 이는 모션제어기의 포토커플러에 의해 반전되어 내부신호는 HIGH가 된다. 그러므로 이 센서는 Active High로 설정 되어져야 한다.



End Limit센서는 기구부의 +방향(Positive)과 -방향(Negative)의 두 개가 있는데, AxmSignalSetLimitSensor함수를 이용하여 설정한다. End LimitLevel을 둘 다 Active High로 설정하고, 리밋센서 검출 시 급정지하도록 설정하여 사용하고자 한다면 다음과 같이 한다.

```
// 0축의 -End limit 과 +End limit 의 Active level 을 HIGH로 설정한다.          Default : HIGH
long lAxisNo = 0;
DWORD uStopMode = 0            // [0]급정지, [1]감속정지
DWORD uPositiveLevel = HIGH;
DWORD uNegativeLevel = HIGH;
AxmSignalSetLimit(lAxisNo, uStopMode, uPositiveLevel, uNegativeLevel);
```

● *Inposition 신호*

인포지션 신호의 사용유무 및 Active Level을 설정할 때에는 AxmSignalSetInpos 함수를 사용하며, Inposition 신호의 Active Level을 Active HIGH로 설정하여 사용하려면 아래와 같이 한다.

```
// 0축의 Inpositon 신호의 입력 레벨을 설정한다.                    Default : HIGH
long lAxisNo = 0;
DWORD uUse = HIGH;
AxmSignalSetInpos(lAxisNo, uUse);
```

● *Alarm 신호*

알람신호의 사용유무 및 Active Level를 설정할 때에는 AxmSignalSetServoAlarm 함수를 사용하며, Alarm 신호의 Active Level을 Active LOW로 설정하려면 아래와 같이 한다.

```
// 0축의 알람 신호의 입력 레벨을 설정한다.                         Default : HIGH
long lAxisNo = 0;
DWORD uUse = LOW;
AxmSignalSetServoAlarm(lAxisNo, uUse);
```

● *Emergency Stop 신호*

급정지 신호의 사용유무 및 Active Level를 설정할 때에는 AxmSignalSetStop 함수를 사용하며, Emergency Stop 신호의 Active Level을 Active HIGH로 설정하여 급정지 하도록 하려면 아래와 같이 한다.

```
//  0축의 비상정지신호(ESTOP)의 입력레벨을 설정한다.
long lAxisNo = 0;
DWORD uStopMode = 0;        //[0] Emergency stop, [1] Slow-Down stop
DWORD uLevel = HIGH;
AxmSignalSetStop(lAxisNo, uStopMode, uLevel);
```

2) 신호 읽기

● *End Limit 신호의 입력 확인*

기구부가 End Limit 센서를 통과할 때 +/-End Limit 신호가 입력되었는지를 확인하고자 한다면, AxmSignalReadLimit 함수를 사용하며, 현재 End Limit신호의 상태를 0또는 1로 반환한다.

```
// 0축의 End Limit 신호를 확인한다.
long lAxisNo = 0;
DWORD uPositiveStatus, uNegativeStatus;
AxmSignalReadLimit(lAxisNo, &uPositiveStatus, &uNegativeStatus);
printf("+ELM : %x , -ELM : %x ₩n", uPositiveStatus, uNegativeStatus);
```

● *Inposition Signal 입력 확인*

이송부가 이동하여 모션제어기에서 명령한 위치까지 이동하였을 때 발생하는 Inposition 신호가 입력되었는지를 확인할 때 AxmSignalReadInpos 함수를 사용한다.

```
// 0축의 Inosition 신호의 입력여부를 확인할 때 사용한다.
long lAxisNo = 0;
DWORD uStatus;
AxmSignalReadInpos(lAxisNo, &uStatus);
printf("0축의 Inposition 여부 : %x ₩n", uStatus);
```

● *Alarm Signal 입력여부 확인*

기구부가 원하는 위치로 구동하지 않고 이상구동을 할 때 서보드라이브는 모션제어기로 Alarm 신호를 출력 하는데 이 신호가 입력되었는지를 확인할 때 AxmSignalReadServoAlarm 함수를 사용한다.

```
// Alarm 신호의 입력여부를 확인할 때 사용한다
long lAxisNo = 0;
DWORD uStatus;
AxmSignalReadServoAlarm(lAxisNo, &uStatus);
printf("0축의 Alarm 여부 : %x ₩n", uStatus);
```

● *Emergency Stop Signal 입력여부 확인*

기구부의 오작동이나 이상작동에 의해 긴급히 기구부의 작동을 멈출 때 Emergency Stop 신호가 발생하는데, Emergency Stop 신호가 입력되었는지를 확인할 때 AxmSignalReadStop 함수를 사용한다.

```
// Emergency Stop 신호의 입력여부를 확인할 때 사용한다.
long lAxisNo = 0;
DWORD uStatus;
AxmSignalReadStop(lAxisNo, &uStatus);
printf("0축의 Emergency Stop 여부 : %x ₩n", uStatus);
```

```
// Ex3_AXM_SettingMotionSignal.cpp : Defines the entry point for the console application.
// 리미트, 알람등 각종 신호를 세팅하고, 입력상태를  확인

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

void main(void)
{
   // 라이브러리를 초기화 한다.
   DWORD Code = AxlOpen();
   if (Code == AXT_RT_SUCCESS)
   {
         printf("라이브러리가 초기화 되었습니다.₩n");

         //모션모듈이 있는지 검사
         DWORD dwStatus;
         Code = AxmInfoIsMotionModule(&dwStatus);
         if(Code == AXT_RT_SUCCESS)
         {
               if(dwStatus == STATUS_EXIST)
               {
                     printf("모션모듈이 존재 합니다.₩n");

                     long lAxisNo = 0;

                     //0축의 펄스 출력 방식을 TwoCwCcwHigh로 한다.
```

```
// Default : OneLowHighLow
DWORD uMethod = TwoCcwCwHigh;
AxmMotSetPulseOutMethod(lAxisNo, uMethod);

// 0축의 -End limit 과 +End limit 의 Active level 을 HIGH로 설정한다.
// Default : HIGH
DWORD uStopMode = 0;            // [0]급정지, [1]감속정지
DWORD uPositiveLevel = HIGH;
DWORD uNegativeLevel = HIGH;
AxmSignalSetLimit(lAxisNo, uStopMode, uPositiveLevel, uNegativeLevel);

// 0축의 Inpositon 신호의 입력 레벨을 설정한다.
//Default : HIGH
DWORD uUse = HIGH;
AxmSignalSetInpos(lAxisNo, uUse);

// 0축의 알람 신호의 입력 레벨을 설정한다.
//Default : HIGH
uUse = LOW;
AxmSignalSetServoAlarm(lAxisNo, uUse);

//  0축의 비상정지신호(ESTOP)의 active 입력레벨을 설정한다.
uStopMode = 0;        //[0] Emergency stop, [1] Slow-Down stop
DWORD uLevel = HIGH;
AxmSignalSetStop(lAxisNo, uStopMode, uLevel);

// help Message
printf("[INFORMATION]*************** ₩n");
```

```
printf("[ESC] : Exit ₩n");
printf("*************************** ₩n₩n");
printf("각종 신호의 입력상태 확인₩n₩n");

BOOL fExit = FALSE;
DWORD uPositiveStatus, uNegativeStatus;
DWORD uInositionStatus, uAlarmStatus, uEStopStatus;

while (!fExit)                    // 무한 루프
{
    if (kbhit())            // 아무키나 누르면 루프를 빠져 나간다.
    fExit = TRUE;

    // 0축의 End Limit 신호를 확인한다.
    AxmSignalReadLimit(lAxisNo, &uPositiveStatus, &uNegativeStatus);

    // 0축의 Inosition 신호의 입력여부를 확인할 때 사용한다.
    AxmSignalReadInpos(lAxisNo, &uInositionStatus);

    // Alarm 신호의 입력여부를 확인할 때 사용한다
    AxmSignalReeadServoAlarm(lAxisNo, &uAlarmStatus);

    // Emergency Stop 신호의 입력여부를 확인할 때 사용한다.
    AxmSignalReadStop(lAxisNo, &uEStopStatus);

    printf("₩rPEnd_Limit(%x), NEnd_Limit(%x), Inposition(%x), Alarm(%x),
        EStop(%x)",
uPositiveStatus, uNegativeStatus, uInositionStatus, uAlarmStatus, uEStopStatus);
```

```
                    } //End of While()
                }
                else
                    printf ("AxmInfoIsMotionModule() : ERROR ( NOT STATUS_EXIST )
                            code 0x%x₩n",Code);
            }
            else
                printf ("AxmInfoIsMotionModule() : ERROR ( Return FALSE )    code 0x%x₩n",Code);

    }
            else
                printf ("AxlOpen() : ERROR    code 0x%x₩n",Code);

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

}
```

범용 입출력 신호의 읽기 및 출력 쓰기

앞에서도 언급한 바와 같이 5비트의 범용 입출력을 제공한다. 사용자 임의로 활용 가능한 범용 입력 신호(Universal Input Signal)중 UIN0, UIN1은 Home sensor, Encoder의 Z상의 신호로 이미 사용되어 있고 나머지 UIN2~4의 3비트는 사용자가 임의로 설정이 가능하게 되어 있다. 또한 범용 출력 신호(Universal Output Signal) 역시 UOUT0, OUT1 2비트가 기본적으로 Servo On, Alarm Clear와 연결되어 있고, 나머지 UOUT2~4의 3비트는 사용자가 임의로 설정이 가능하게 되어 있다.
범용 입출력 비트의 상태확인은 AxmSignalReadInput함수 또는 AxmReadOutput를 이용하여 한번에 읽어오는 방법 이외에 AxmSignalReadInputBit함수 또는 AxmSignalReadOutputBit 함수를 이용하여 개별적으로 0~4번까지 5개 Bit의 상태를 확인하는 방법이 있으며, 해당 비트의 신호 상태에 따라 0또는 1을 반환 한다.

```
// 0번 축의 0번 범용 입력 비트 신호의 상태를 확인한다.
long lAxisNo = 0;
long lBitNo = 0;
DWORD upOn;
AxmSignalReadInputBit(lAxisNo, lBitNo, &upOn);
printf("0번 축의 0번 범용 입력 비트 신호의 상태 : %x",upOn);

// 0번 축의 0번 범용 출력 비트 신호의 상태를 확인한다.
AxmSignalReadOutputBit(lAxisNo, lBitNo, &upOn);
printf("0번 축의 0번 범용 출력 비트 신호의 상태 : %x",upOn);
```

범용 출력 신호의 On/Off를 쓰기 위해서는 AxmWriteOutput함수를 이용하여 한꺼번에 설정할 수 도 있고, AxmSignalWriteOutputBit함수를 이용하여 Bit단위로 설정할 수도 있다. 범용 출력 신호를 HIGH로 할 때에는 Bit 값을 1로 하고, LOW로 할 때에는 Bit 값을 0으로 한다. 이러한 형식으로 0~4번 신호의 Bit를 설정한다.

```
// 범용 출력 0번 비트 신호를 1로 설정한다.(Servo On)
long lAxisNo = 0;
```

```
long lBitNo = 0;
DWORD uOnOff= 1;
AxmSignalWriteOutputBit(long lAxisNo, long lBitNo, DWORD uOnOff);

// 범용 출력 0번 비트 신호를 0으로 설정한다.(Servo Off)
uOn = 0;
AxmSignalWriteOutputBit(long lAxisNo, long lBitNo, DWORD uOnOff);
```

```
// Ex4_AXM_SettingUniversalIO.cpp : Defines the entry point for the console application.
// 범용 입출력을 설정 및 확인한다.

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

void main(void)
{
   // 라이브러리를 초기화 한다.
   DWORD Code = AxlOpen();
   if (Code == AXT_RT_SUCCESS)
   {
         printf("라이브러리가 초기화 되었습니다.₩n");

         //모션모듈이 있는지 검사
         DWORD dwStatus;
         Code = AxmInfoIsMotionModule(&dwStatus);
         if(Code == AXT_RT_SUCCESS)
```

```
{
    if(dwStatus == STATUS_EXIST)
    {
        printf("모션모듈이 존재 합니다.₩n");

        // help Message
        printf("[INFORMATION]*************** ₩n");

        printf("[ESC] : Exit ₩n");
        printf("[0 ~ 4] : 범용 출력 신호 On / Off  ₩n");
        printf("**************************** ₩n");


        long lAxisNo = 0;
        long lBitNo;
        DWORD uOnOff, uStatusInput[5], uStatusOutput[5];
        BOOL fExit = FALSE;
        while (!fExit)                  // 무한 루프
        {
            if (kbhit())            // 아무키나 누르면
            {
                int ch = getch();
                switch (ch)
                {
                    case 27:  fExit = TRUE;     break;     // Esc key
                    case '0':  lBitNo = 0;      break;
                    case '1':  lBitNo = 1;      break;
                    case '2':  lBitNo = 2;      break;
                    case '3':  lBitNo = 3;      break;
```

```
                    case '4':  lBitNo = 4;          break;
                }
                AxmSignalReadOutputBit(lAxisNo, lBitNo, &uOnOff);
                if(uOn == 1) {AxmSignalWriteOutputBit(lAxisNo, lBitNo, 0); }
                else { AxmSignalWriteOutputBit(lAxisNo, lBitNo, 1); }
            }

            //개별 입출력 비트의 정보를 확인하는 함수를 이용하여 상태 확인
            for(int nBitNo =0; nBitNo <5; nBitNo ++)
            {
                AxmSignalReadInputBit(lAxisNo, nBitNo, &uStatusInput[nBitNo]);
                AxmSignalReadOutputBit(lAxisNo, nBitNo, &uStatusOutput[nBitNo]);
            }
            printf("₩r입력0(%x),1(%x),2(%x),3(%x),4(%x)/출력
                 0(%x),1(%x),2(%x),3(%x),4(%x)",
                uStatusInput[0] , uStatusInput[1] , uStatusInput[2] ,
uStatusInput[3],
                uStatusInput[4], utatusOutput[0], uStatusOutput[1],
                uStatusOutput[2], uStatusOutput[3], uStatusOutput[4]);

        } //End of While()
    }
    else
        printf ("AxmInfoIsMotionModule() : ERROR ( NOT STATUS_EXIST )
            code 0x%x₩n",Code);
}
else
    printf ("AxmInfoIsMotionModule() : ERROR ( Return FALSE )
        code 0x%x₩n",Code);
```

```
    }
        else
            printf ("AxlOpen() : ERROR    code 0x%x₩n",Code);

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

설정 정보의 저장 및 불러오기

매번 프로그램을 작성할 때마다 모든 모션파라미터와 모션신호들을 설정해주지 않고, 저장되어 있는 설정 파일을 불러옴으로써 간단하게 초기화 작업을 수행할 수 있다.
모션 구동 기능 모듈의 레지스터에 설정되어있는 정보를 파일로 저장 하기 위해서는 AxmMotSaveParaAll 함수를 사용한다. AxmMotSaveParaAll 함수는 모든 축의 정보를 파일로 저장하는 함수이다. 모든 축에 대한 설정 정보를 CURRENT_CONF.MOT 파일로 저장하고자 한다면 다음과 같이 한다.

```
// 모든 축에 대한 설정정보를 CURRENT_CONF.mot 파일에 저장한다.
char *pFilename = " CURRENT_CONF.MOT";
DWORD uOption = 0;
AxmMotSaveParaAll (pFilename);
```

파일에 저장되어 있는 설정 정보를 모션 모듈 레지스터에 설정하기 위해서는 AxmMotLoadParaAll 함수를 사용한다. AxmMotLoadParaAll 함수는 모든 축의 정보를 읽어오는 함수이다. CURRENT_CONF.MOT 파일에 저장되어 있는 모든 축에 대한 설정 정보를 읽어오고자 한다면 다음과 같이 한다.

```
// . CURRENT_CONF.mot 파일에 저장되어 있는 모든 축의 모션 정보를 읽어와서 설정한다.
char *pFilename = " . CURRENT_CONF.mot";
AxmMotLoadParaAll(pFilename);
```

파일 설정되는 내부변수 리스트 및 초기값

No	신호관련 변수	파라메타 설명	설정 값
00	AXIS_NO	축 번호	설정 축번호
01	PULSE_OUT_METHOD	모션노드에서 서보팩 지령으로 사용될 펄스의 출력방식을 설정	TwoCwCcwHigh(6)
02	Enc[Encoder]	Encoder신호의 입력방식을 설정	ObverseSqr4Mode (3)
03	InP[InPos]	서보팩 위치결정 완료신호의 사용여부 및 Active Level을 설정	2
04	Alm[Alarm]	서보팩 알람 신호의 사용여부 및 Active level을 설정	1
05	NLimit	기구 (-)리미트 센서의 사용여부 및 Active level을 설정	1
06	PLimit	기구 (+)리미트 센서의 사용여부 및 Active level을 설정	1
07	MinSpd	모터 구동 시 초기 시작구동 속도 설정	1.0
08	MaxSpd	모터 구동 시 최대구동 속도 설정	700000.0
09	HmSig	원점센서로 사용할 신호를 설정	4
10	HmLev	원점센서의 Active Level을 설정	1
11	HmDir	원점센서 검출 시 초기 검색진행 방향을 설정	0
12	Zphas_Lev	엔코더 Z상 Active level	1
13	Zphas_Use	원점센서 검출 후 엔코더 Z상을 검출 여부 설정	0
14	Stop Signal Mode	ESTOP, SSTOP 사용시 모드	0
15	Stop Signal level	ESTOP, SSTOP 사용 레벨	0

16	VelFirst	원점검색 시 초기 고속검출 속도(속도의 단위는 Unit/pulse를 1/1로 한경우에 PPS[Pulses/sec])	10000.0
17	VelSecond	원점검색 시 1차 센서 검출 후 반대 방향으로 빠져 나오는 속도	10000.0
18	VelThird	원점검색 시 1차 센서 검출 후 재 검색 진행하는 속도	2000.0
19	VelLast	원점검색 시 최종 검출 속도 설정 [원점검색의 정밀도 결정]	100.0
20	AccFirst	원점검색 시 초기 고속검출 가속도(가속도의 단위는 Unit/pulse를 1/1로 한경우에 PPS[Pulses/sec^2])	40000.0
21	AccSecond	원점검색 시 1차 센서 검출 후 반대 방향으로 빠져 나오는 가속도(가속도의 단위는 Unit/pulse를 1/1로 한경우에 PPS[Pulses/sec^2])	40000.0
22	HClrTim	원점검색 완료 후 Cmd, Act위치를 Clear전 대기 시간	1000.0
23	HOffset	원점검색 완료 후 기구원점 재 설정 시 이동 Offset 값	0.0
24	NSWLimit	원점검색 완료 후 적용할 (-) Software Limit 값 설정	0.0
25	PSWLimit	원점검색 완료 후 적용할 (+) Software Limit 값 설정	0.0
26	OenPulse	모터 1회전에 필요한 펄스 수 설정 [서보팩 전자기어 비 참조]	1.0
27	OneUnit	모터 1회전 시 이동되는 이동 량 [mm, 도, 등등]	1

매번 프로그램을 작성할 때마다 모든 모션파라미터와 모션신호들을 설정해주지 않고, 저장되어 있는 설정 파일을 불러옴으로써 간단하게 초기화 작업을 수행할 수 있다.

모션 구동

모션구동 개요

1) One Axis

2) Multi Axis

모션구동의 용어해설

1) Servo On

2) 속도 프로파일

3) 시점탈출과 종점 탈출

4) 상대위치구동과 절대 위치구동

단축 구동

1) 위치구동

2) 속도구동

3) 신호검색 구동

4) 원점검색 구동

5) 모션 구동의 정지

6) 모션 구동상태 확인

다축구동

1) 다축위치구동

2) 보간 구동

모션구동 개요

모션구동은 멈추어있거나 움직이는 특정한 기구부를 프로그램이라는 매개체를 통해 제어하는 것이다. 모션 구동은 하나의 축을 구동하는 단축 구동과 두 가지 축 이상을 구동하는 다축 구동으로 나눌 수 있다.

1) One Axis
2) Multi Axis

1) One Axis

● *Position Drive*

위치구동은 가장 일반적인 구동으로 이동할 거리가 주어지면 이동 속도, 가(감)속도 또는 가(감)속시간 등의 이동에 필요한 인자들을 입력 받아, 모션모듈에 있는 모션칩에서 설정된 속도 프로파일에 따라 계산한 값만큼 펄스를 출력하여 이동하는 구동을 뜻한다.
이러한 위치구동은 일반적인 구동 시스템들이 이미 알고 있는 위치들을 반복적으로 이동하도록 설계되기 때문에 가장 많이 쓰여지는 구동함수라고 할 수 있다.

● *Velocity Drive*

속도구동은 미리 정해진 거리에 의해 이동하는 위치구동과는 달리 정지명령이나 Limit 센서등에 의한 정지신호가 발생할 때까지 계속 일정한 속도로 움직이는 구동을 뜻한다. 이러한 속도구동 함수는 조그버튼의 (+)방향을 누르면 이송부가 (+)방향으로 이동하다가 버튼을 떼면 정지하는 작업등을 하는데 이용될 수 있다.

● *Signal Search Drive*

신호검색구동이란 Limit 신호나 범용입출력 신호등의 검출대상을 미리 설정하고, 일정 속도로 움직이다가 설정한 신호가 감지될 경우 정지하는 구동을 뜻한다. 기구부의 End Limit이나 홈센서의 위치에 정확히 정지하여 필요한 작업을 수행하도록 하는데 사용되어진다.

● *Home Search Drive*

홈검색 구동은 좌표계의 기준이 되는 원점을 찾는 구동을 뜻한다. 홈검색을 위한 기본설정을 한 후 하나의 함수를 실행함으로써 원점검색 과정을 편리하게 하기 위해 제공되는 구동함수이다.

2) Multi Axis

일반적으로 구동시스템은 여러 축을 한꺼번에 일정한 위치로 이동시켜야 하는 경우가 많이 있고, 이러한 구동을 위해서 다축구동 함수를 제공한다.

● *Position Drive*

하나의 모듈에 포함된 2개 의 축에서 이동거리, 이동 속도, 가(감)속도 또는 가(감)속시간 등의 이동에 필요한 인자들을 입력 받아 모션모듈에 있는 모션칩 에서 계산한 값만큼 펄스를 출력하여 이동하는 구동을 뜻한다. 이 구동은 두 축이 동시에 움직임을 시작한다는 점이 단축구동함수를 이용한 구동과의 가장 큰 차이라고 할 수 있다.

● *Interpolation Drive*

하나의 모듈에 포함된 2개의 축을 이용하여 현재위치에서 원하는 위치로 직선 또는 원호의 일정한 경로에 따라 보간하여 움직이도록 하는 구동을 뜻한다. 보간 구동에는 크게 직선보간구동과 원호보간 구동이 있으며, 한 점에서 출발하여 가속-정속-감속의 속도변화로 계산된 경로를 따라가는 일반적인 보간 구동이 있다.

모션구동의 용어해설

1) Servo On
2) 속도 프로파일
3) 시점탈출과 종점 탈출
4) 상대위치구동과 절대 위치구동

1) Servo On

서보드라이브를 On시킨다는 의미로 서보드라이브가 On되어 있지 않으면 모션카드에서 나오는 출력 펄스가 실제 구동에 영향을 주지 못하므로 구동하기 전에 반드시 서보드라이브를 On으로

해주어야 한다.
서보드라이브의 ON/OFF를 위해서는 먼저 AxmSignalSetServoOnLevel 함수를 이용하여 ServoOn신호의 레벨을 설정해주어야 한다. 많은 서보드라이브가 Active HIGH지만, 그렇지 않을 수도 있다.

```
//ServoOn 레벨을 HIGH로 한다.
long lAxisNo = 0;
DWORD uLevel = HIGH;
AxmSignalSetServoOnLevel(lAxisNo, uLevel);
```

다음으로 서보드라이브에 On 신호를 주어야 하는데, 이러한 기능을 하는 함수가 Axm ServoOn 이며, 서보드라이브가 On되어 있는지 여부를 확인하는 함수는 AxmSignalIsServoOn 이다.

```
//0축의 서보드라이브를 On시킨다.
long lAxisNo = 0;
DWORD uUse = ENABLE;
AxmSignalServoOn(lAxisNo, uUse);
```

```
//0축의 서보드라이브를 OFF시킨다.
long lAxisNo = 0;
DWORD uUse = DISABLE;
AxmSignalServoOn(lAxisNo, uUse);
```

```
//서보드라이브가 On되어 있는지를 확인한다.
long lAxisNo = 0;
DWORD uUse;
AxmSignalIsServoOn(lAxisNo, &uUse);
if(uUse == ENABLE) { printf("서보드라이브가 ON되어 있습니다."); }
else if(uUse == DISABLE) { printf("서보드라이브가 OFF되어 있습니다."); }
```

2) 속도 프로파일

사용자가 모션 구동 함수를 통해 움직이라는 명령을 보내면, 기구부는 미리 세팅해둔 Start Stop Speed로 구동을 시작하고 입력받은 가속도로 가속하여 입력받은 속도가 되면 정속을 유지하다가 입력받은 가속도로 감속하여 정지하는 과정을 거치게 된다. 속도 프로파일은 가속 및 감속시 어떠한 형태로 속도가 변하면서 가감속 하는지를 나타낸다.
AXL에서는 사다리꼴 가감속, S자 가감속등을 지원하며 가감속의 형태가 가속시와 감속시의 속도가 다른 비대칭 프로파일도 만들 수 있다. 따라서, 아래의 다섯 가지 형태로 속도 프로파일을 제공하며 각각 0~4까지의 설정값을 갖는다.

가감속 프로파일	설정값 (ProfileMode)
대칭 사다리꼴 가감속	0
비대칭 사다리꼴 가감속	1
대칭 Quasi-S Curve	2(지원 안함)
대칭 S자 가감속	3
비대칭 S자 가감속	4

라이브러리 함수에서는 AxmMotSetProfileMode 라는 함수를 이용하여 다음과 같이 설정한다.

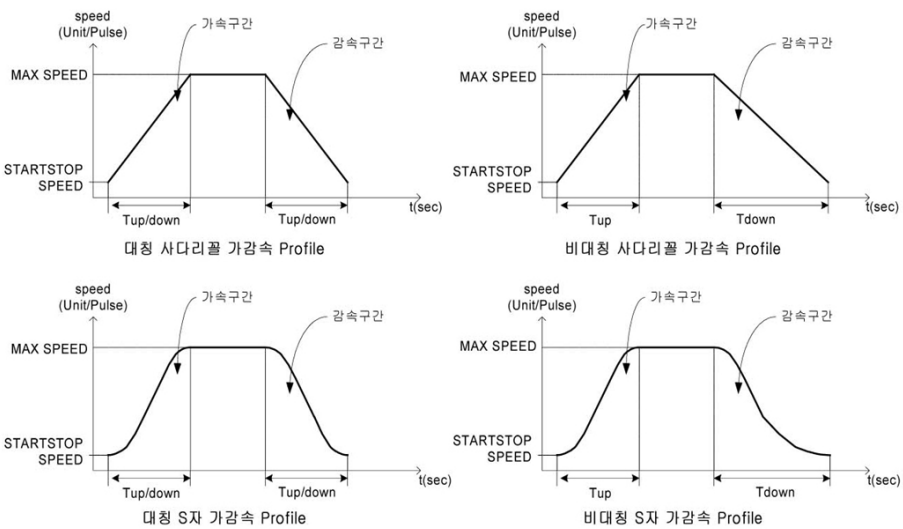
```
//0축의 속도프로파일을 대칭S자 가감속으로 한다.
long lAxisNo = 0;
DWORD uProfileMode = 4;
AxmMotSetProfileMode(lAxisNo, uProfileMode);
```

현재 설정되어 있는 값을 확인하고자 한다면 다음과 같이 하면 된다.

```
// 설정된 가감속 프로파일을 읽는 함수
long lAxisNo=0;
DWORD uProfileMode;
AxmMotGetProfileMode(lAxisNo, &upProfileMode);
```

```
switch(upProfileMode) {
    case 0 :   printf("현재 속도 프로파일 : 대칭 사다리꼴 가감속");      break;
    case 1 :   printf("현재 속도 프로파일 : 비대칭 사다리꼴 가감속");    break;
    case 3 :   printf("현재 속도 프로파일 : 대칭 S자 가감속");           break;
    case 4 :   printf("현재 속도 프로파일 : 비대칭 S자 가감속");         break;
}
```

다음 그림은 대칭사다리꼴, 비대칭사다리꼴, 대칭 S자형, 비대칭 S자형의 속도 프로파일을 도시하고 있다. 그림에서 사다리꼴 속도프로파일보다 S자형 속도 프로파일의 속도의 변화율(가속도)이 S자를 이루어 운동이 부드러워진다.



3) 시점탈출과 종점 탈출

사용자가 프로그램에서 함수를 호출하였을 때, 그 함수를 언제 빠져나가서 프로그램상의 다음 명령을 실행시키는가를 의미하는 것이 시점탈출과 종점탈출이다.

● *시점 탈출*

모션칩에서 첫 펄스를 출력하자마자 프로그램 상의 다음 명령어로 진행하는 것을 말한다. 즉, 펄스 출력이 시작되자마자 함수를 탈출한다는 뜻이다. 이 방식은 PC의 CPU를 점유하지 않고 바로 프로그램상의 다음 명령어로 간다는 장점이 있다. 즉, 모션칩에게 모션제어를 위한 펄스출력을 맡겨 놓고 PC는 다른 일을 할 수 있다.

● *종점 탈출*

모션칩에서 명령에 해당하는 펄스를 모두 출력한 후에 이 함수를 벗어나 프로그램 상의 다음 명령어로 진행되는 방식을 말한다. 즉 축 구동이 종료될 때까지 PC의 CPU를 점유하여 프로그램상의 다음 명령으로 넘어가지 않눈다.
펄스 종점 탈출에서 주의할 사항은 서보드라이브로 부터 모션제어기로 오는 Inposition 신호를 받고 벗어나느냐의 여부를 설정하는 것인데, 받지 않고 벗어나는 것이 디폴트로 되어 있다. 만약 사용자가 더욱 정밀한 제어를 필요로 한다면 Inposition 신호를 사용하는 것을 권장한다.

4) 상대위치구동과 절대 위치구동

주어진 구동명령이 상대 좌표로 이동할 것인지, 절대좌표로 이동할 것인지를 구분할 필요가 있다. 상대 좌표계에서는 함수의 인자로 주어진 거리에 대해서 현재 위치를 기준으로 그 거리만큼 움직이게 되고, 절대 좌표계에서는 주어진 거리에 대해서 현재 위치를 뺀 거리만큼 움직이게 된다. AxmMotSetAbsRelMode함수를 이용하여 설정하는데, 절대 위치구동에서는 0점이 존재해야 절대좌표를 구성할 수 있기 때문에 원점검색을 반드시 행하여 0점을 잡은 후에 사용하여야 한다.
상대좌표 또는 절대좌표를 설정하고자 한다면 다음과 같이 하면 된다.

```
// 절대위치 구동으로 설정
long lAxisNo = 0;
DWORD uAbsRelMode = 0;        // [0] 절대좌표, [1] 상대좌표
AxmMotSetAbsRelMode (lAxisNo, uAbsRelMode);
```

현재 설정되어 있는 값을 확인하고자 한다면 다음과 같이 하면 된다.

```
// 설정된 이동 모드를 읽는 함수
long lAxisNo=0;
```

```
DWORD uAbsRelMode;
AxmMotGetAbsRelMode (lAxisNo, &uAbsRelMode);
if(uAbsRelMode == 0) printf("절대 위치구동중");
elseif(uAbsRelMode == 1) printf("상대 위치구동중");
```

단축 구동

일반적으로 모션 구동을 할 때 가장 기본이 되는 구동을 단축 구동이라고 한다. 단축구동은 크게 위치구동(Position Drive), 속도구동(Velocity Drive), 신호 검색구동(Signal Search Drive), 원점검색 구동(Home Search Drive)으로 이루어져 있다.

1) 위치구동
2) 속도구동
3) 신호검색 구동
4) 원점검색 구동
5) 모션 구동의 정지
6) 모션 구동상태 확인

1) 위치구동

위치구동은 기구부를 원하는 위치로 이동시키기 위한 명령이다. 위치구동을 위해 Axm PositionMove 함수와 AxmMoveStartPos함수가 있다. 이 두 함수는 설정한 거리 또는 위치로 입력해준 속도 및 가속도로 이동하도록 하는데, AxmMovePos 함수는 종점 탈출 함수로 구동이 종료되어야 프로그램상의 다음 명령을 실행시키고, AxmMoveStartPos 함수는 시점탈출 함수로서 동작이 시작되자 마자 프로그램상의 다음 명령으로 넘어간다.
종점탈출 함수인 AxmMovePos 함수를 사용하고자 한다면 다음과 같이 한다. 앞에서도 언급한 바와 같이 좌표계와 속도 프로파일을 따로 설정하는 함수를 사용하고, 이동량, 속도, 가속도를 설정한다. 아래와 같이 대칭 프로파일 구동일 경우 입력해주는 감속도는 무시되며 입력해준 가속도가 가감속시 적용된다.

```
//0축을 절대좌표로 100위치까지 대칭 S자 구동모드로 이동 (속도 100, 가속도 200)
```

```
long lAxisNo = 0;
DWORD uAbsRelMode = 0;
DWORD uProfileMode = 3;
AxmMotSetAbsRelMode(lAxisNo, uAbsRelMode);
AxmMotSetProfileMode(lAxisNo, uProfileMode);

double dPosition = 100;
double dMaxVelocity = 100;
double dMaxAccel = 200;
double dMaxDecel = 200;
AxmMovePos(lAxisNo, dPosition, dMaxVelocity, dMaxAccel, dMaxDecel);
```

시점탈출 함수인 AxmMoveStartPos 함수를 사용하고자 한다면 다음과 같이 한다. 시점탈출 이므로 구동이 시작되고 바로 다음 함수가 실행되는데, 만약 다음 함수가 구동함수라면 반드시 AxmStatusReadInMotion 함수를 이용하여 구동중인지 아닌지를 확인하여 구동이 종료된 후 다음 구동 명령을 내려야 한다. 구동중일때는 모든 모션 구동명령이 에러를 반환하고 실행되지 않기 때문이다. 여기서 AxmStatusReadInMotion함수는 모션구동 중이라면 TRUE를, 구동 중이 아니라면 FALSE를 반환한다.

```
//0축을 상대좌표로 100위치까지 비대칭 S자 구동모드로 이동 (속도 100, 가속도 200, 감속도 100)
long lAxisNo = 0;
DWORD uAbsRelMode = 1;
DWORD uProfileMode = 4;
AxmMotSetAbsRelMode(lAxisNo, uAbsRelMode);
AxmMotSetProfileMode(lAxisNo, uProfileMode);

double dPosition = 100;
double dMaxVelocity = 100;
double dMaxAccel = 200;
double dMaxDecel = 100;
```

```
AxmMoveStartPos(lAxisNo, dPosition, dMaxVelocity, dMaxAccel, dMaxDecel);

DWORD uStatus;
AxmStatusReadInMotion(lAxisNo, &uStatus);
while(uStatus) {
        AxmStatusReadInMotion(lAxisNo, &uStatus);
}
```

```
// Ex5_AXM_PositionMove.cpp : Defines the entry point for the console application.
// 위치구동함수로 100만큼 이동 후 다시 -100만큼 이동 후 종료한다.

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

#define AXIS_0 0

void main(void)
{
   // 라이브러리를 초기화 한다.
   DWORD Code = AxlOpen();
   if (Code == AXT_RT_SUCCESS)
   {
        printf("라이브러리가 초기화 되었습니다.₩n");

        //모션모듈이 있는지 검사
```

```
DWORD dwStatus;
Code = AxmInfoIsMotionModule(&dwStatus);
if(Code == AXT_RT_SUCCESS)
{
    if(dwStatus == STATUS_EXIST)
    {
        printf("모션모듈이 존재 합니다.₩n");

        // 0축의 +End limit 과 -End limit 의 Active level 을 HIGH로 설정한다.
        AxmSignalSetLimit(AXIS_0, 0, HIGH, HIGH);
        // 0축의 Inpositon 신호의 입력 레벨을 HIGH로 설정한다.
        AxmSignalSetInpos(AXIS_0, HIGH);
        // 0축의 알람 신호의 입력 레벨을 LOW로 설정한다.
        AxmSignalSetServoAlarm(AXIS_0, LOW);
        // 0축의 비상정지신호(ESTOP)의 Active 입력레벨을 HIGH로 설정한다.
        AxmSignalSetStop(AXIS_0, 0, HIGH);

        //0축 지령값을 mm단위가 되도록 한다.
        AxmMotSetMoveUnitPerPulse(AXIS_0, 10, 10000);
        //0축에 초기 속도를 1로 설정한다. Default : 1
        AxmMotSetMinVel(AXIS_0, 1);
        //0축의 펄스 출력 방식을 TwoCwCcwHigh로 한다.
        AxmMotSetPulseOutMethod(AXIS_0, TwoCcwCwHigh);
        //지정 축에 Encoder 입력 방식을 4체배로 설정한다.
        AxmMotSetEncInputMethod(AXIS_0, ObverseSqr4Mode);

        //현재위치를 원점으로 설정한다.
        AxmStatusSetActPos(AXIS_0, 0.0);
        AxmStatusSetCmdPos(AXIS_0, 0.0);
```

```
        //Servo On
        AxmSignalServoOn(AXIS_0, ENABLE);

        printf("0축을 절대좌표로 100위치까지 대칭S자 구동
            (속도 100, 가속도 200, 종점 탈출)₩n");
        printf("아무 키나 누르면 구동을 시작합니다.₩n");
        getch();
        AxmMotSetAbsRelMode(AXIS_0, 0);
        AxmMotSetProfileMode(AXIS_0, 3);
        AxmMovePos(AXIS_0, 100, 100, 200, 200);

        printf("0축을 상대좌표로 -100위치까지 비대칭S자 구동(속도 100, 가속도 200,
            시점탈출)₩n");
        printf("아무 키나 누르면 구동을 시작합니다.₩n");
        getch();
        AxmMotSetAbsRelMode(AXIS_0, 1);
        AxmMotSetProfileMode(AXIS_0, 4);
        AxmMoveStartPos(AXIS_0, -100, 100, 200, 200);

        //현재 구동 중이면 대기한다.
        DWORD uStatus;
        AxmStatusReadInMotion(AXIS_0, &uStatus);
        while(uStatus) {
            AxmStatusReadInMotion(AXIS_0, &uStatus);
        }

    }
    else
```

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

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

}
```

2) 속도구동

속도구동은 속도, 가속도, 가속시간을 정하여 종료 명령이 나올 때까지 계속 움직이는 형태의 구동이며, 속도구동 함수는 모두 펄스가 시작되는 시점에서 함수를 벗어나는 펄스 시점탈출함수이다. 속도구동을 하기 위해서는 AxmMoveVel 함수 사용하는데, AxmMoveVel 함수는 일정한 직선속도로 이동하도록 하는 명령이다. 구동방향은 속도 값의 부호로 결정된다.

```
//0축을 100의 속도로 (+)방향으로 대칭S자 속도 프로파일로 구동 시작한다.
long lAxisNo = 0;
DWORD uProfileMode = 3;
AxmMotSetProfileMode(lAxisNo, uProfileMode);
```

```
double dMaxVelocity = 100;
double dMaxAccel = 200;
double dMaxDecel = 200;
AxmMoveVel(lAxisNo, dMaxVelocity, dMaxAccel, dMaxDecel);
```

```
//0축을 100의 속도로 (-)방향으로 대칭사다리꼴 속도 프로파일로 구동 시작한다.
long lAxisNo = 0;
DWORD uProfileMode = 0;
AxmMotSetProfileMode(lAxisNo, uProfileMode);

double dMaxVelocity = -100;
double dMaxAccel = 200;
double dMaxDecel = 200;
AxmMoveVel(lAxisNo, dMaxVelocity, dMaxAccel, dMaxDecel);
```

이렇게 속도구동 함수를 시작하면 정지 명령을 내릴 때까지 계속 구동하게 된다. 구동의 정지는 뒤에서 상세히 언급할 것이며, 아래와 같이 하면 된다.

```
//0축의 구동을 설정된 감속도로 감속하면서 정지 하도록 한다.
long lAxisNo = 0;
AxmMoveSStop(lAxisNo);
```

```
// Ex6_AXM_VelocityMove.cpp : Defines the entry point for the console application.
// 속도구동함수를 이용하여 구동하고, 정지명령에 의해 구동을 종료한다.

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

#define AXIS_0 0

void main(void)
{
   // 라이브러리를 초기화 한다.
   DWORD Code = AxlOpen();
   if (Code == AXT_RT_SUCCESS)
   {
         printf("라이브러리가 초기화 되었습니다.₩n");
               //모션모듈이 있는지 검사
         DWORD dwStatus;
         Code = AxmInfoIsMotionModule(&dwStatus);
         if(Code == AXT_RT_SUCCESS)
         {
               if(dwStatus == STATUS_EXIST)
               {
                     printf("모션모듈이 존재 합니다.₩n");

                     // 0축의 +End limit 과 -End limit 의 Active level 을 HIGH로 설정한다.
                     AxmSignalSetLimit(AXIS_0, 0, HIGH, HIGH);
```

```
// 0축의 Inpositon 신호의 입력 레벨을 HIGH로 설정한다.
AxmSignalSetInpos(AXIS_0, HIGH);
// 0축의 알람 신호의 입력 레벨을 LOW로 설정한다.
AxmSignalSetServoAlarm(AXIS_0, LOW);
// 0축의 비상정지신호(ESTOP)의 Active 입력레벨을 HIGH로 설정한다.
AxmSignalSetStop(AXIS_0, 0, HIGH);

//0축 지령값을 mm단위가 되도록 한다.
AxmMotSetMoveUnitPerPulse(AXIS_0, 10, 10000);
//0축에 초기 속도를 1로 설정한다. Default : 1
AxmMotSetMinVel(AXIS_0, 1);
//0축의 펄스 출력 방식을 TwoCcwCwHigh로 한다.
AxmMotSetPulseOutMethod(AXIS_0, TwoCcwCwHigh);
//지정 축에 Encoder 입력 방식을 4체배로 설정한다.
AxmMotSetEncInputMethod(AXIS_0, ObverseSqr4Mode);


//현재위치를 원점으로 설정한다.
AxmStatusSetActPos(AXIS_0, 0.0);
AxmStatusSetCmdPos(AXIS_0, 0.0);

//Servo On
AxmSignalServoOn(AXIS_0, ENABLE);

// help Message
printf("[INFORMATION]*************** ₩n");

printf("[ESC] : Exit after Servo Off ₩n");
```

```
printf("[1] : (+)방향으로 속도구동 시작 ₩n");
printf("[2] : (-)방향으로 속도구동 시작 ₩n");
printf("[3] : STOP  ₩n");
printf("**************************** ₩n");


DWORD uStatus;
BOOL fExit = FALSE;
while (!fExit)                    // 무한 루프
{
    if (kbhit())            // 아무 키나 누르면
    {
        int ch = getch();
        switch (ch)
        {
            case 27:        // Esc key
                fExit = TRUE;
                AxmSignalServoOn(AXIS_0, DISABLE); //Servo Off
            break;

            case '1':
// 구동 중이 아니라면, (+)방향으로 대칭사다리꼴 속도프로파일로 연속 구동한다.
                AxmStatusReadInMotion(AXIS_0, &uStatus);
                if(!uStatus)
                {
                    AxmMotSetProfileMode(AXIS_0, 0);
                    AxmMoveVel(AXIS_0, 100, 200, 200);
                }
            break;
```

```
                    case '2':
            // 구동 중이 아니라면, (-)방향으로 대칭S자 속도프로파일로 연속 구동한다.
                        AxmStatusReadInMotion(AXIS_0, &uStatus);
                        if(!uStatus)
                        {
                            AxmMotSetProfileMode(AXIS_0, 3);
                            AxmMoveVel(AXIS_0, -100, 200, 200);
                        }
                    break;

                    case '3':  //구동 중이라면, 감속 정지한다.
                        AxmStatusReadInMotion(AXIS_0, &uStatus);
                        if(uStatus)
                            AxmMoveSStop(AXIS_0);
                    break;
                }
            }
        } //End of While()
    }
    else
        printf ("AxmInfoIsMotionModule() : ERROR ( NOT STATUS_EXIST )
                code 0x%x₩n",Code);
}
else
    printf ("AxmInfoIsMotionModule() : ERROR ( Return FALSE )
            code 0x%x₩n",Code);
}
else
```

```
                printf ("AxlOpen() : ERROR    code 0x%x₩n",Code);




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

}
```

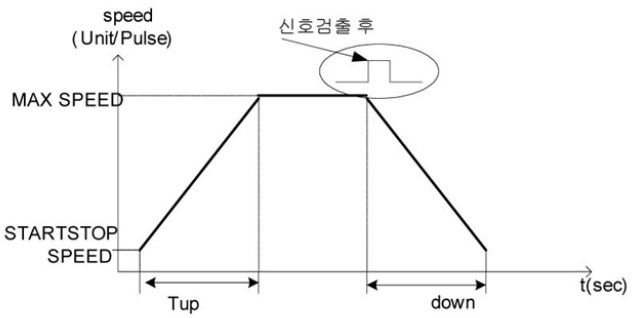
3) 신호검색 구동

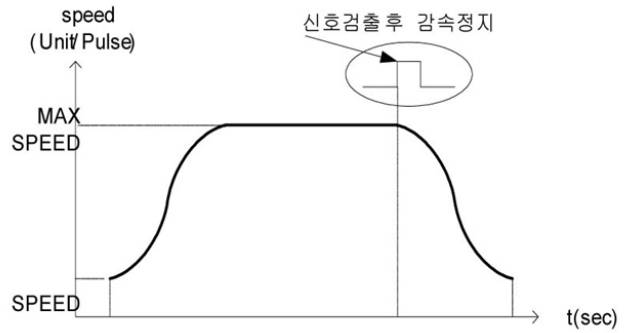
신호검색구동이란 Limit 신호나 범용입출력 신호등의 검출대상을 미리 설정하고, 일정 속도로 움직이다가 설정한 신호가 감지될 경우 정지하는 구동을 뜻한다. 신호검색구동은 특정 신호가 검색되면 감속 또는 급정지하는 AxmMoveSignalSearch 함수와 신호가 검색되면 그 위치를 저장하는 AxmMoveSignalCapture함수가 있다. 검색할 수 있는 신호는 아래의 표에서와 같이 8개 중 6개의 신호를 사용 하는데, 각각 Up Edge와 Down Edge가 있으므로 총 12개의 신호가 있다.

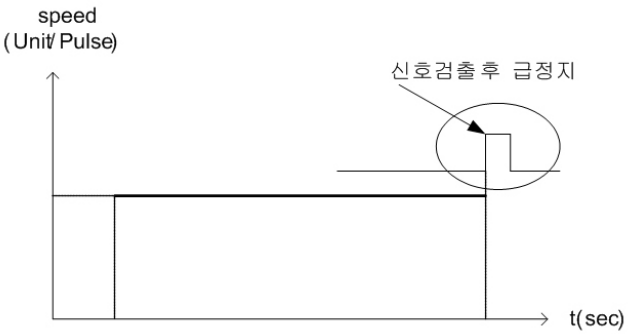
Define	value	검출 신호
PosEndLimit	0x0	+Elm(End limit) +방향 리미트 센서 신호
NegEndLimit	0x1	-Elm(End limit) -방향 리미트 센서 신호
PosSloLimit	0x2	+Slm(Slow Down limit) 신호 - 사용하지 않음
NegSloLimit	0x3	-Slm(Slow Down limit) 신호 - 사용하지 않음
HomeSensor	0x4	IN0(ORG) 원점 센서 신호
EncodZPhase	0x5	IN1(Z상) Encoder Z상 신호
UniInput02	0x6	IN2(범용) 범용 입력 2번 신호
UniInput03	0x7	IN3(범용) 범용 입력 3번 신호

신호검색구동을 하기 위해서는 AxmMoveSignalSearch 함수를 사용하는데, 이 함수는 특정 입력신호의 Edge를 검출하여 급정지 또는 감속정지하는 함수로 검출할 입력 신호와 선택한 입력신호의 Edge 방향, 그리고 구동방향, 속도 및 가속도, 정지방법를 입력받는다. 여기서 구동방향은 속도 값의 부호에 따라 결정된다. 특히 이 함수는 다음 그림에서와 같이 해당신호가 검출된 후에 감속을 시작하므로 정확한 신호위치의 검색을 위해서는 신호검색구동에서 감속정지 한 후에 반대방향으로 저속으로 신호 검색하면서 해당신호에서 급정지 하는 방법으로 사용하여야 한다.

```
// -방향으로 이동하면서 홈센서의 Up Edge를 찾는다.
long lAxisNo = 0;
double dMaxVelocity = -100;
double dMaxAccel = 200;
long lDetectSignal = HomeSensor;
long lSignalEdge = UP_EDGE;
long lSignalMethod = 0; // [0]감속정지, [1]급정지
AxmMoveSignalSearch(lAxisNo, dMaxVelocity, dMaxAccel, lDetectSignal, lSignalEdge,
lSignalMethod);

// +방향으로 천천히 이동하면서 홈센서의 Down Edge를 검색한 후 급정지 한다.
double dMaxVelocity = 20;
double dMaxAccel = 0;
long lDetectSignal = HomeSensor;
long lSignalEdge = DOWN_EDGE;
long lSignalMethod = 1; // [0]감속정지, [1]급정지
AxmMoveSignalSearch(lAxisNo, dMaxVelocity, dMaxAccel, lDetectSignal,
lSignalEdge ,lSignalMethod);
```







단순히 신호가 검색되는 위치에 정지시키는 것 뿐만 아니라 해당신호의 위치를 확인하고자 한다면 AxmMoveSignalCapture함수를 사용하면 된다. 이 함수를 호출하면 해당 신호를 검색하기 시작하고, 신호가 검색되면 해당위치를 저장하게 되는데, AxmMoveGetCapturePos 함수를 사용하여 저장된 위치를 확인할 수 있다. 저장되는 위치는 Command Position 또는 Actual Position 값 중 하나이며, 이를 통해 원점으로부터의 거리를 확인할 수 있다.

```
// +방향으로 이동하면서 +ELM의 Up Edge를 찾는다.
long lAxisNo = 0;
double dMaxVelocity = 100;
double dMaxAccel = 200;
long lDetectSignal = PosEndLimit;
long lSignalEdge = UP_EDGE
long lTarget = 0;          //[0] Command Position, [1] Actual Position
long lSignalMethod = 0   //[0]감속정지,    [1]급정지
AxmMoveSignalCapture(lAxisNo, dMaxVelocity, dMaxAccel, lDetectSignal, lSignalEdge, lTarget,
lSignalMethod);

//+ELM센서의 위치를 확인한다.
double dCapPotition;
AxmMoveGetCapturePos(lAxisNo, &dCapPotition);
printf("+ELM센서의 위치 : %f", dCapPotition);
```

```
// Ex7_AXM_SignalSearchDrive.cpp : Defines the entry point for the console application.
// 신호검색 구동함수를 이용해서 리미트 센서와 홈센서로 이동한다.

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

#define AXIS_0 0

void main(void)
```

```
{
    // 라이브러리를 초기화 한다.
    DWORD Code = AxlOpen();
    if (Code == AXT_RT_SUCCESS)
    {
        printf("라이브러리가 초기화 되었습니다.₩n");

        //모션모듈이 있는지 검사
        DWORD dwStatus;
        Code = AxmInfoIsMotionModule(&dwStatus);
        if(Code == AXT_RT_SUCCESS)
        {
            if(dwStatus == STATUS_EXIST)
            {
                printf("모션모듈이 존재 합니다.₩n");
                // 0축의 +End limit 과 -End limit 의 Active level 을 HIGH로 설정한다.
                AxmSignalSetLimit(AXIS_0, 0, HIGH, HIGH);
                // 0축의 Inpositon 신호의 입력 레벨을 HIGH로 설정한다.
                AxmSignalSetInpos(AXIS_0, HIGH);
                // 0축의 알람 신호의 입력 레벨을 LOW로 설정한다.
                AxmSignalSetServoAlarm(AXIS_0, LOW);
                // 0축의 비상정지신호(ESTOP)의 Active 입력레벨을 HIGH로 설정한다.
                AxmSignalSetStop(AXIS_0, 0, HIGH);

                //0축 지령값을 mm단위가 되도록 한다.
                AxmMotSetMoveUnitPerPulse(AXIS_0, 10, 10000);
                //0축에 초기 속도를 1로 설정한다. Default : 1
                AxmMotSetMinVel(AXIS_0, 1);
                //0축의 펄스 출력 방식을 TwoCwCcwHigh로 한다.
```

```
AxmMotSetPulseOutMethod(AXIS_0, TwoCcwCwHigh);
//지정 축에 Encoder 입력 방식을 4체배로 설정한다.
AxmMotSetEncInputMethod(AXIS_0, ObverseSqr4Mode);

//현재위치를 원점으로 설정한다.
AxmStatusSetActPos(AXIS_0, 0.0);
AxmStatusSetCmdPos(AXIS_0, 0.0);
//Servo On
AxmSignalServoOn(AXIS_0, ENABLE);

// help Message
printf("[INFORMATION]*************** ₩n");

printf("[ESC] : Exit ₩n");
printf("[1]:+ELM센서의 Up Edge를 감지할 때까지 +방향으로 이동 후
   사다리꼴 감속정지 ₩n");
printf("[2] : -ELM센서의 Up Edge 를 감지할 때까지 -방향으로 이동 후
   S-커브 감속정지 ₩n");
printf("[3] : IN/OUT5(ORG) DownEdge를 감지할 때까지 +방향으로 이동 후
   급정지 ₩n");
printf("**************************** ₩n");

DWORD uPositiveStatus, uNegativeStatus, uHomeStatus;
BOOL fExit = FALSE;
while (!fExit)                    // 무한 루프
{
    if (kbhit())                  // 아무 키나 누르면
    {
        int ch = getch();
```

```
switch (ch)
{
    case 27:        // Esc key
        fExit = TRUE;
        AxmSignalServoOn(AXIS_0, DISABLE);   //Servo Off
    break;

    case '1':
        // 0축에 100의 속도와 200의 가감속도로 사다리꼴 구동,
        // +ELM센서의 Up Edge를 감지할 때까지 +방향으로 구동
        // 후 감속정지 한다.
        AxmMotSetProfileMode(AXIS_0, 0);
        AxmSigSearchMove(AXIS_0, 100, 200, PosEndLimit,
         UP_EDGE,0);
    break;

    case '2':
        // 0축에 100의 속도와 200의 가감속도로 S자 구동,
        // -ELM센서의 Up Edge를 감지할 때까지 -방향으로 구동
        //후 감속정지 한다.
        AxmMotSetProfileMode(AXIS_0, 3);
        AxmSigSearchMove(AXIS_0, -100, 200, NegEndLimit,
         UP_EDGE,0);
    break;

    case '3':
    // 0축에 20의 속도로 홈센서의 Down Edge를 감지할 때까지
    //+방향으로 구동 후 급정지
        AxmSigSearchMove(AXIS_0, 20, 0, HomeSensor,
```

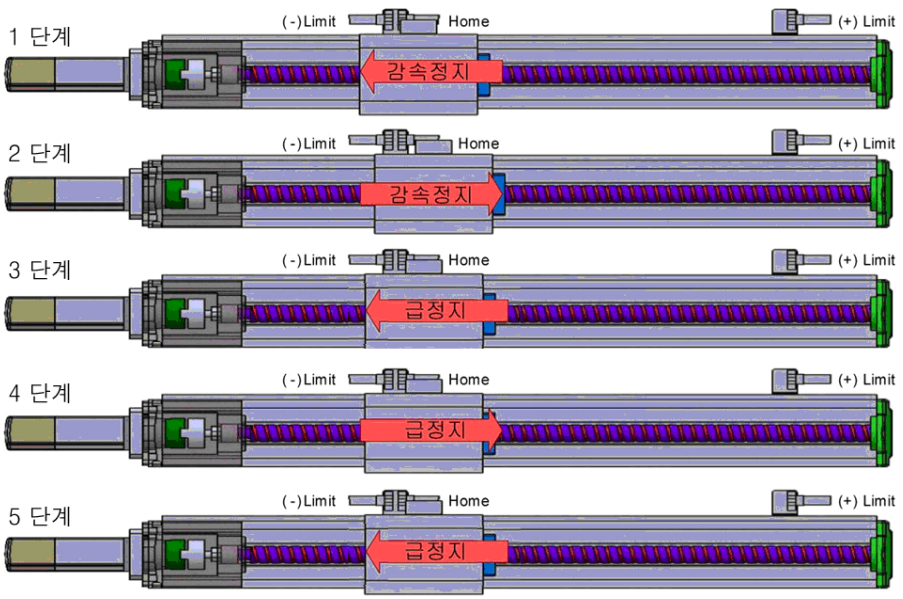
```
                                        DOWN_EDGE,1);
                                break;
                            }
                        }
                    // 각 센서의 신호상태를 확인한다.
                        AxmSignalReadLimit(AXIS_0, &uPositiveStatus, &uNegativeStatus);
                        AxmSignalReadInputBit(AXIS_0, 0, &uHomeStatus);
                        printf("₩r신호상태 : P_Elm(%d),N_Elm(%d),Home(%d)",
                                uPositiveStatus, uNegativeStatus, uHomeStatus);

                } //End of While()
            }
            else
                printf ("AxmInfoIsMotionModule() : ERROR ( NOT STATUS_EXIST )
                        code 0x%x₩n",Code);
        }
        else
            printf ("AxmInfoIsMotionModule() : ERROR ( Return FALSE )
                    code 0x%x₩n",Code);
    }
        else
            printf ("AxlOpen() : ERROR    code 0x%x₩n",Code);

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

4) 원점 검색 구동

하드웨어 초기화, 파라미터 설정, 서보온의 과정으로 모션구동에 작업이 준비되어 있다 하더라도 좌표계의 방향만 설정되어 있을 뿐 좌표계의 기준이 되는 원점은 설정이 되어 있지 않은 상태이다. 기구부가 사용자가 원하는 일정한 위치로 이송되기 위해서는 정확한 원점이 보장되어야 한다. 원점검색을 하는 방법에는 크게 신호검색구동함수를 이용하는 방법과 원점검색함수를 이용하는 방법이 있다. 신호검색을 이용하여 홈검색을 수행하기 위해서는 상당히 번거로운 잡업들이 이루어지게 되기 때문에 이러한 과정을 더 편하게 하기 위해 홈검색 함수를 제공한다. 원점검색함수를 이용하기 위해서는 AxmHomeSetMethod 함수와 AxmHomeSetVel 함수를 이용하여 홈검색에 필요한 데이터들을 설정해주고 AxmHomeSetStart 함수를 호출하여 홈검색 을 시작하면 된다.
원점검색함수는 개별적인 쓰레드를 생성하여 작동하므로, 만약 여러 축의 원점검색을 해야 하는 경우라면, 각각의 원점검색명령을 열거해서 사용하면 된다.

● *원점검색 방법 및 속도 설정*

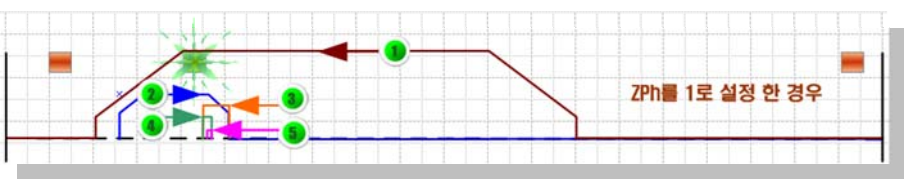
원점검색은 기구부에 따라 여러 가지 신호검색 순서가 발생할 수 있지만, AXL의 원점검색 함수는 아래의 7단계 원점검색 시퀀스로 이루어지며, 원점검색을 위한 함수의 인자와 비교하여 원점검색시퀀스를 보면 다음과 같다.



1단계 : -방향으로 이동하면서 홈센서의 Rising Edge신호를 검색하여 감속정지
(홈센서가 이미 HIGH 상태라면 1단계를 생략)
2단계 : +방향으로 이동하면서 홈센서의 Falling Edge신호를 검색하여 감속정지
3단계 : -방향으로 이동하면서 홈센서의 Rising Edge신호를 검색하여 급정지
< Z 상 검색을 사용하지 않은 경우 >
4단계 : +방향으로 이동하면서 홈센서의 Falling Edge신호를 검색하여 급정지
5단계 : -방향으로 이동하면서 홈센서의 Rising Edge신호를 검색하여 급정지
<Z 상 검색을 사용한 경우 >
4단계 : +방향으로 이동하면서 Z 상의 Falling Edge신호를 검색하여 급정지
5단계 : -방향으로 이동하면서 Z 상의 Rising Edge신호를 검색하여 급정지

6단계 : Offset값이 있다면 Offset값 만큼 추가 이동
7단계 : 사용자 설정 시간동안 대기 후 위치값을 0으로 설정

● *Step별 원점검출 속도설정의 이해*



① VelFirst, AccFirst사용
HmDir 방향으로 원점센서를 고속 검출하고 HAccF 감속도로 감속정지

② VelSecond, AccSecond사용
HmDir 반대 방향으로 원점센서의 Down Edge를 검출하고 HAccF 감속도로 감속정지

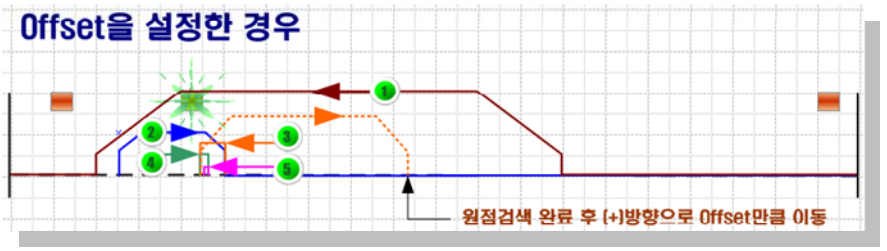
③ VelThird사용
HmDir 방향으로 원점센서의 Up Edge를 검출하고 급정지

④ VelLast 사용
HmDir 반대 방향으로 Z상의 Down Edge를 검출하고 급정지

⑤ VelLast 사용
HmDir 방향으로 Z상의 Up Edge를 검출하고 급정지

● *Offset 값 설정의 이해*

기구적인 원점센서 검출이 완료된 후 특정 위치로 옮겨 원점을 설정해야 될 경우 HOffset에 값을 지정하면 원점 센서 검출이 완료된 후 자동으로 해당 Offset만큼 이동 완료 후 원점을 재설정하게 된다. 이때 주의할 것은 (+)나 (-)Limit 을 벗어나지 않는 범위에서 Offset을 지정해야 된다. 그렇지 않을 경우 원점검색을 정상적으로 수행할 수 없습니다.



Offset 이동 완료 후 자동으로 0점으로 설정됩니다.

● *Offset 값 설정의 이해*

원점검색이 정상적으로 완료된 후 Soft Limit을 설정할 수 있습니다. Soft Limit은 기구적으로 Limit센서가 장착되어 있지 않거나 상황에 따라서 Limit을 변경해야 될 경우 사용하면 된다. 이때 한가지 주의하실 것은 NLimit(-)보다 PLimit(+)값이 커야 정상적으로 Soft Limit이 설정된다. 만약 NSwLimit값이 PSwLimit 값보다 같거나 클 경우 Soft Limit는 설정되지 않는다.

● *원점 검색 함수의 변수설명*

원점검색관련 설정변수	변수 설명
HmSig	원점센서로 사용할 신호를 설정 예) 원점센서(4), +Limit센서(0) 등
HmLev	원점센서의 Active Level을 설정 예) 0: A접점, 1: B접점
HmDir	원점센서 검출시 초기 검색진행 방향을 설정 예) 1: (+)방향, 0: (-)방향
Zphas	원점센서 검출 후 엔코더 Z상을 검출 여부 설정 예) 1:사용함, 0: 사용안함 4번 Step에서 Z상의 Down Edge를 검출하고 5번 Step에서 Z상의 Up Edge를 재검출함으로 원점검색 속도와 정밀도를 높이기 위해 사용
VelFirst	원점검색 시 초기 고속검출 속도 (초기 원점센서가 감지되지 않았을 경우)
VelSecond	원점검색 시 1차 센서 검출 후 반대 방향으로 빠져나오는 속도
VelThird	원점검색 시 1차 센서 검출 후 재 검색 진행하는 속도
VelLast	원점검색 시 최종 검출 속도 설정 [원점검색의 정밀도 결정]
AccFirst	원점검색 시 초기 고속검출 가속도
AccSecond	원점검색 시 1차 센서 검출 후 반대 방향으로 빠져나오는 가속도
HClrTim	원점검색 완료 후 Command, Actual(Encoder)위치를 Clear전 대기시간
HOffset	원점검색 완료 후 기구원점 재 설정 시 이동 Offset 값
NSWLimit	원점검색 완료 후 적용할 (-) Software Limit 값 설정
PSWLimit	원점검색 완료 후 적용할 (+) Software Limit 값 설정 (NSWLimit값이 PSWLimit값보다 같거나 클 경우 S/W Limit 설정되지 않음)

주의사항 : 레벨을 잘못 설정시 -방향으로 설정해도 +방향으로 동작할수 있으며, 홈을 찾는데 있어 문제가 될수있다.

```
// 원점검색 방법을 설정하는 함수
long lAxisNo = 0;                          //원점 검색이 이루어질 축 번호
long nHmDir = -1;                          //원점검색 1단계의 이동방향,(-)방향을 입력
DWORD uHomeSignal = HomeSensor;            //신호검색구동에서의 신호들을 사용
```

```
DWORD uZphas = 1;                    // Z상 검출 유무, 1이면 검출 하고, 0이면 안함
double dHomeClrTime = 2000.0;        // 원점 검색 Enc 값 Set하기 위한 대기시간
double dHomeOffset = 0.0;            // 원점검색후 Offset값만큼 추가적으로 이동
AxmHomeSetMethod(lAxisNo,nHmDir,uHomeSignal,uZphas,dHomeClrTime,dHomeOffset);

// 원점검색시 사용될 속도를 설정
double dVelFirst = 100;          //1단계에서의 속도
double dVelSecond = 20;          //2단계에서의 속도
double dVelThird = 10;           //3단계에서의 속도
double dvelLast = 10;            // index검색및 정밀하게 검색하기위한 속도 (Offset값 이동시 적용)
double dAccFirst = 200;          //1단계에서의 가속도
double dAccSecond = 40;          //2단계에서의 가속도
AxmHomeSetVel(lAxisNo, dVelFirst, dVelSecond, dVelThird, dvelLast, dAccFirst, dAccSecond);
```

여기서 원점 검색속도는 1단계에서는 고속으로 하여도 되나, 2단계와 3단계는 정밀한 작업이 이루어지므로 매우 작은 값으로 하여야 한다. 또한 가속도도 1단계는 적당한 가속도를 선택하면 되지만, 2단계는 상당히 큰 값을 설정하는 것이 바람직하다.
때로는 기구부의 상태에 따라 위에서 제공하는 원점검색 시퀀스를 사용할 수 없는 경우가 있는데, 그럴 때는 신호검색구동 함수를 이용하여 자신의 기구부에 맞는 원점검색 시퀀스를 작성하여 사용해야 한다.

● *원점검색 구동 시작*

지금까지는 홈검색을 위해 설정해야 하는 설정값들에 대하여 알아보았는데, 원점검색을 실제적으로 수행하는 함수는 AxmHomeSetStart 이다. 이 함수가 호출되면 앞에서 설정한 설정값대로 지정축에서 홈검색을 시작한다.

```
//0축의 원점검색을 시작한다.
long lAxisNo = 0;
AxmHomeSetStart(lAxisNo);
```

● *원점검색 진행률 확인*

원점검색의 실행 시 현재 원점검색 진행률을 확인할 수 있다. 진행률은 AxmHomeGetRate 함수를 이용하여 확인하며, 특정 단계를 넘어가면 값이 증가하고 0에서 시작하여 100이 되면 원점검색이 종료되었음을 의미한다.

```
//0축의 홈검색 진행률을 확인한다.
long lAxisNo = 0;
DWORD uHomeMainStepNumber, uHomeStepNumber;
AxmHomeGetRate(lAxisNo, &uHomeMainStepNumber , &uHomeStepNumber);
printf("현재 진행률 : %d ", uHomeStepNumber);
// uHomeMainStepNumber은 겐트리일 경우 마스터,슬레이브가 동작하는지 확인할 때 이용한다.
// 마스터축 동작일 때는 0을 리턴, 슬레이브동작일 때는 10을 리턴한다.
```

● *원점검색 완료 및 정지*

원점검색함수를 실행하면 새로운 쓰레드를 만들어 원점검색을 실행하게 되고, 원점검색이 끝나면 원점검색의 결과를 리턴할 필요가 있다. 원점검색의 결과를 확인하기 위해서는 AxmHomeGetResult 함수를 사용하며 아래와 같은 결과를 반환하게 된다.

```
//원점검색의 결과를 반환한다.
long lAxisNo = 0;
DWORD uHomeResult;
AxmHomeGetResult(lAxisNo, &uHomeResult);
```

Define	Description
HOME_ERR_UNKNOWN	알수없는 축 번호로 원점검색을 시작하려 할 때
HOME_ERR_GNT_RANGE	겐트리 구동축의 Master축과 Slave축의 원점검색 결과가 설정한

	OffsetRange를 벗어났을 경우
HOME_ERR_USER_BREAK	원점 검색중 사용자가 정지 명령을 실행했을 경우
HOME_ERR_VELOCITY	원점 검색속도 설정이 되어있지 않을 경우
HOME_ERR_AMP_FAULT	원점 검색중 서보팩 알람이 발생 할 경우
HOME_ERR_NEG_LIMIT	(+)방향으로 원점센서 검출 중 (-)리미트 센서가 감지될 경우
HOME_ERR_POS_LIMIT	(-)방향으로 원점센서 검출 중 (+)리미트 센서가 감지될 경우
HOME_ERR_NOT_DETECT	원점센서가 감지되지 않을 경우
HOME_SEARCHING	현재 원점검색이 진행중일 경우
HOME_SUCCESS	원점 검색이 성공적으로 종료 됐을 경우

5) 모션 구동의 정지

모션구동의 마무리 단계인 모션의 정지에는 급정지, 감속정지, 특정 감속도를 통한 정지 방식이 있다. 급정지를 구현할 때에는 AxmMoveEStop, 감속정지를 구현할 때에는 AxmMoveSStop, 특정 감속도를 통한 정지를 할 때는 AxmMoveStop함수를 각각 사용하여 모션을 정지시킨다. 특히 AxmMoveStop은 감속도를 설정해주도록 되어 있는데, 이 함수가 호출되면 기존의 입력된 감속도는 무효가 되고 새로 입력되는 감속도로 감속정지하게 된다. 이 감속도를 0으로 하면 AxmMoveEStop, 원래의 감속도로 설정하면 AxmMoveSStop과 같은 기능이 된다.

```
// 0축에서 모션구동을 급정지한다.
long lAxisNo = 0;
AxmMoveEStop(lAxisNo);
```

```
// 0축에서 모션구동을 감속 정지한다.
long lAxisNo = 0;
AxmMoveSStop(lAxisNo);
```

```
// 0축에서 모션구동을 100의 감속도로 정지한다.
long lAxisNo = 0;
double dMaxDecel = 100;
AxmMoveStop(lAxisNo, dMaxDecel);
```

```
// Ex8_AXM_HomeSearchDrive.cpp : Defines the entry point for the console application.
// 원점검색 함수를 이용하여 원점검색을 한다.

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

#define AXIS_0 0

void main(void)
{
   // 라이브러리를 초기화 한다.
   DWORD Code = AxlOpen();
   if (Code == AXT_RT_SUCCESS)
   {
         printf("라이브러리가 초기화 되었다.₩n");

         //모션모듈이 있는지 검사
         DWORD dwStatus;
         Code = AxmInfoIsMotionModule(&dwStatus);
         if(Code == AXT_RT_SUCCESS)
         {
```

```
if(dwStatus == STATUS_EXIST)
{
    printf("모션모듈이 존재 한다.₩n");

    // 0축의 +End limit 과 -End limit 의 Active level 을 HIGH로 설정한다.
    AxmSignalSetLimit(AXIS_0, 0, HIGH, HIGH);
    // 0축의 Inpositon 신호의 입력 레벨을 HIGH로 설정한다.
    AxmSignalSetInpos(AXIS_0, HIGH);
    // 0축의 알람 신호의 입력 레벨을 LOW로설정한다.
    AxmSignalSetServoAlarm(AXIS_0, LOW);
    // 0축의 비상정지신호(ESTOP)의 Active 입력레벨을 HIGH로 설정한다.
    AxmSignalSetStop(AXIS_0, 0, HIGH);

    //0축 지령값을 mm단위가 되도록 한다.
    AxmMotSetMoveUnitPerPulse(AXIS_0, 10, 10000);
    //0축에 초기 속도를 1로 설정한다. Default : 1
    AxmMotSetMinVel(AXIS_0, 1);
    //0축의 펄스 출력 방식을 TwoCwCcwHigh로 한다.
    AxmMotSetPulseOutMethod(AXIS_0, TwoCcwCwHigh);
    //지정 축에 Encoder 입력 방식을 4체배로 설정한다.
    AxmMotSetEncInputMethod(AXIS_0, ObverseSqr4Mode);

    //Servo On
    AxmSignalServoOn(AXIS_0, ENABLE);

    // 원점검색 방법을 설정
    long lAxisNo = 0;                   //원점 검색이 이루어질 축 번호
    long nHmDir = -1;                   //원점검색 1단계의 이동방향, (-)방향을 입력
    DWORD uHomeSignal = HomeSensor;  //신호검색구동에서의 신호들을 사용
```

```
DWORD uZphas = 1;            //Z상 검출 유무, 1이면 검출 하고, 0이면 안함
double dHomeClrTime = 2000.0;   //원점 검색 Enc 값 Set하기 위한 대기시간
double dHomeOffset = 0.0;          //원점검색후 Offset값만큼 추가적으로 이동
AxmHomeSetMethod(AXIS_0, nHmDir, uHomeSignal, uZphas, dHomeClrTime,
 dHomeOffset );

// 원점검색시 사용될 속도를 설정
double dVelFirst = 100;           //1단계에서의 속도
double dVelSecond = 20;           //2단계에서의 속도
double dVelThird = 10;            //3단계에서의 속도
double dvelLast = 10;                 // index검색및 정밀하게 검색하기위한 속도
                                      // (Offset값 이동시 적용)
double dAccFirst = 200;           //1단계에서의 가속도
double dAccSecond = 40;                   //2단계에서의 가속도
AxmHomeSetVel(AXIS_0, dVelFirst, dVelSecond, dVelThird, dvelLast, dAccFirst,
                    dAccSecond);

// help Message
printf("[INFORMATION]*************** ₩n");

printf("[ESC] : Exit ₩n");
printf("[1] : 원점검색 시작 ₩n");
printf("[2] : 원점검색 종료 ₩n");
printf("**************************** ₩n");


DWORD uHomeResult;
DWORD uHomeMainStepNumber , uHomeStepNumber;
BOOL fExit = FALSE;
```

```
while (!fExit)                    // 무한 루프
{
    if (kbhit())            // 아무키나 누르면
    {
        int ch = getch();
        switch (ch)
        {
            case 27:        // Esc key
                fExit = TRUE;
                AxmSignalServoOn(AXIS_0, DISABLE);      //Servo Off
            break;

            case '1':// 0축에 원점검색 중이 아니라면,원점검색을 시작한다.
                AxmHomeGetResult(AXIS_0, &uHomeResult);
                if(uHomeResult)//원점검색 진행중 : 0, 원점검색이 완료 : 1
                {
                    AxmHomeSetStart(AXIS_0);
                    printf("₩n원점검색을 시작한다.₩n");
                }
            break;

            case '2'://0축에 원점검색 진행중이라면,원점검색을 종료시킨다.
                AxmHomeGetResult(AXIS_0, &uHomeResult);
                if(uHomeResult == HOME_SEARCHING)
                    //원점검색 진행중 : 0
                {
                    AxmMoveSStop(AXIS_0);
                    printf("₩n원점검색을 종료한다.₩n");
                }
```

```
                                break;
                            }
                        }

                        //현재 원점검색 진행률을 확인
                        AxmHomeGetRate(AXIS_0, &uHomeMainStepNumber,
                          &uHomeStepNumber);
                        printf("₩r 원점검색 진행률 : %d      ", uHomeStepNumber);

                    } //End of While()
                }
                else
                    printf ("AxmInfoIsMotionModule() : ERROR ( NOT STATUS_EXIST )
                            code 0x%x₩n",Code);
            }
            else
                printf ("AxmInfoIsMotionModule() : ERROR ( Return FALSE )
                        code 0x%x₩n",Code);
        }
        else
            printf ("AxlOpen() : ERROR    code 0x%x₩n",Code);



        // 라이브러리를 종료 한다.
        if (AxlClose())
            printf("₩n라이브러리가 종료 되었다.₩n");
        else
            printf("₩n라이브러리가 정상적으로 종료되지 않았다.₩n")
    }
```

6) 모션 구동상태 확인

모션구동을 하면 모션제어기로부터 펄스가 발생되어 기구부가 펄스값 만큼 움직인다. 이러한 모션구동에서 현재의 위치나 속도, 구동상태, 정지원인 등의 정보를 확인하는 것이 구동 및 정지 상태 모니터링이다.

● *현재 위치 및 지령위치 확인*

모션제어기에서 현재 얼마만큼의 이동을 명령하고 있는지, 모터의 엔코더에서 피드백 된 실제 기구부의 이동거리는 얼마인지 등을 확인할 수 있다. 특정 축에 Actual 위치를 확인할 때에는 AxmStatusGetActPos 함수를 사용하여 아래와 같이 한다.

```
//0축의 현재 위치를 확인한다.
long lAxisNo = 0;
double dPosition;
AxmStatusGetActPos(lAxisNo, &dPosition);
printf("0축의 실제 위치 : %f", dPosition);
```

Command 위치를 확인할 때에는 AxmStatusGetCmdPos 함수를 사용하여 아래와 같이 한다.

```
//0축의 지령 위치를 확인한다.
long lAxisNo = 0;
double dPosition;
AxmStatusGetCmdPos(lAxisNo, &dPosition);
printf("0축의 지령 위치 : %f", dPosition);
```

Command 위치와 Actual 위치의 차이인 Error값을 확인할 때에는 AxmStatusReadPosError 함수를 사용하여 아래와 같이 한다.

```
//0축의 지령 위치와 실제위치의 차이를 확인한다.
long lAxisNo = 0;
double dError;
AxmStatusReadPosError(lAxisNo, &dError);
printf("0축의 지령 위치와 실제위치의 차 : %f", dError);
```

● *현재속도 확인*

속도확인은 기구부의 현재 구동하는 속도를 얻어오는 것을 뜻한다. 현재구동속도를 확인할 때에는 AxmStatusReadVel 함수를 사용하여 아래와 같이 한다.

```
//0축의 현재 속도를 확인한다.
long lAxisNo = 0;
double dVelocity;
AxmStatusReadVel(lAxisNo, &dVelocity);
printf("0축의 현재 속도 : %f", dVelocity);
```

● *모션구동 여부 확인*

지정축의 펄스 출력상태를 확인하는 것으로, 간단히 말하자면 기구부가 움직이고 있는지 구동이 끝났는지를 확인하는 것을 뜻한다. 특정 축이 움직이고 있는지를 확인할 때에는 AxmStatusReadInMotion 함수를 사용한다.

```
//0축이 구동을 하고 있는지를 확인한다.
long lAxisNo = 0;
DWORD uStatus;
AxmStatusReadInMotion(lAxisNo, &uStatus);
```

● *구동상태 정보 확인*

지정축의 DriveStatus 레지스터를 확인함으로써 특정 축의 구동상태를 읽어올 수 있다. 특정 축의 구동상태를 확인하기 위해서는 AxmStatusReadMotion 함수를 사용한다.

```
//0축의 구동상태를 읽어온다.
long lAxisNo = 0;
DWORD uStatus;
AxmStatusReadMotion(lAxisNo, &uStatus);
```

AxmStatusReadMotion 함수는 여기서 언급하는 정보 외에도 EIP-M2/4Q 각각 보다 세부적인 정보를 추가적으로 확인할 수 있으므로 해당 칩의 매뉴얼을 참조하기 바란다.특히, 많이 사용하는 값은 다음과 같다.

※ 주요 사용되는 EIP-M2/4Q Drive Status

Define	설 명
QIDRIVE_STATUS_0	BUSY (DRIVE 중)
QIDRIVE_STATUS_1	DOWN (감속 DRIVE 중)
QIDRIVE_STATUS_2	CONST (정속 DRIVE 중)
QIDRIVE_STATUS_3	Up (가속 DRIVE 중)
QIDRIVE_STATUS_4	연속 드라이브 구동 중
QIDRIVE_STATUS_5	지정 거리 드라이브 구동 중
QIDRIVE_STATUS_6	MPG 드라이브 구동 중

● *정지상태 정보 확인*

지정축의 EndStatus 레지스터를 확인함으로써 특정축의 종료상태를 확인할 수 있다. 특정 축의 종료상태를 확인하기 위해서는 AxmStatusReadStop 함수를 사용한다.

```
//0축의 종료 상태를 확인한다.
long lAxisNo = 0;
```

```
DWORD uStatus;
AxmStatusReadStop(lAxisNo, &uStatus);
```

AxmStatusReadStop 함수는 라이브러리 매뉴얼에서 살펴볼 수 있듯이 여러 가지의 정지상태를 반환하므로, 현재 어떤 이유로 정지되었는지를 확인할 수 있는데, 0을 반환했다면, 정상적으로 구동을 종료했다는 의미이고, 0이 아닌 값을 반환한다면 해당하는 종료원인에 의해 종료되었다고 볼 수 있다. 주로 사용하는 값은 다음과 같다.

※ 주요 사용되는 EIP-M2/4Q

Define	설 명
QIEND_STATUS_0	정방향 리미트 신호(PELM)에 의한 종료
QIEND_STATUS_1	역방향 리미트 신호(NELM)에 의한 종료
QIEND_STATUS_4	정방향 소프트 리미트 급정지 기능에 의한 구동 종료
QIEND_STATUS_5	역방향 소프트 리미트 급정지 기능에 의한 구동 종료
QIEND_STATUS_6	정방향 소프트 리미트 감속정지 기능에 의한 구동 종료
QIEND_STATUS_7	역방향 소프트 리미트 감속정지 기능에 의한 구동 종료
QIEND_STATUS_8	서보 알람 기능에 의한 구동 종료
QIEND_STATUS_9	비상 정지 신호 입력에 의한 구동 종료
QIEND_STATUS_10	급 정지 명령에 의한 구동 종료
QIEND_STATUS_11	감속 정지 명령에 의한 구동 종료

```
// Ex9_AXM_MotionStatus1.cpp : Defines the entry point for the console application.
// 상대위치구동 함수를 이용하여 200만큼 (+)방향으로 이동하면서 이동거리 및 속도를 확인

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

```
#define AXIS_0 0

void main(void)
{
   // 라이브러리를 초기화 한다.
   DWORD Code = AxlOpen();
   if (Code == AXT_RT_SUCCESS)
   {
         printf("라이브러리가 초기화 되었습니다.₩n");

         //모션모듈이 있는지 검사
         DWORD dwStatus;
         Code = AxmInfoIsMotionModule(&dwStatus);
         if(Code == AXT_RT_SUCCESS)
         {
               if(dwStatus == STATUS_EXIST)
               {
                     printf("모션모듈이 존재 합니다.₩n");

                     // 0축의 +End limit 과 -End limit 의 Active level 을 HIGH로 설정한다.
                     AxmSignalSetLimit(AXIS_0, 0, HIGH, HIGH);
                     // 0축의 Inpositon 신호의 입력 레벨을 HIGH로 설정한다.
                     AxmSignalSetInpos(AXIS_0, HIGH);
                     // 0축의 알람 신호의 입력 레벨을 LOW로 설정한다.
                     AxmSignalSetServoAlarm(AXIS_0, LOW);
                     // 0축의 비상정지신호(ESTOP)의 Active 입력레벨을 HIGH로 설정한다.
                     AxmSignalSetStop(AXIS_0, 0, HIGH);

                     //0축 지령값을 mm단위가 되도록 한다.
```

```
AxmMotSetMoveUnitPerPulse(AXIS_0, 10, 10000);
//0축에 초기 속도를 1로 설정한다. Default : 1
AxmMotSetMinVel(AXIS_0, 1);
//0축의 펄스 출력 방식을 TwoCwCcwHigh로 한다.
AxmMotSetPulseOutMethod(AXIS_0, TwoCcwCwHigh);
//지정 축에 Encoder 입력 방식을 4체배로 설정한다.
AxmMotSetEncInputMethod(AXIS_0, ObverseSqr4Mode);

AxmMotSetAbsRelMode(AXIS_0, 0);          //절대위치구동
AxmMotSetProfileMode(AXIS_0, 3);         //대칭 S자 구동

//Servo On
AxmSignalServoOn(AXIS_0, ENABLE);

printf("₩n[INFORMATION]***************************** ₩n");

printf("AP : Actual Position ₩n");
printf("CP : Command Position ₩n");
printf("Error :  Actual Position - Command Position ₩n");
printf("CV : Command Velocity ₩n");
printf("****************************************** ₩n");

printf("0축을 S자 속도프로파일로 (+)방향 200만큼 이동(속도:100, 가속도:200,
    시점탈출)₩n");
printf("아무 키나 누르면 구동을 시작합니다.");
getch();
AxmMoveStartPos(AXIS_0, 200, 100, 200, 200);

// 구동되고 있는 동안 구동상태를 확인한다.
```

```
                DWORD uStatus;
                double dActualPosition, dCommandPosition, dError, dVelocity;
                AxmStatusReadInMotion( AXIS_0, &uStatus);
                while(uStatus)
                {
                    AxmStatusReadInMotion( AXIS_0, &uStatus);
                    AxmStatusGetActPos(AXIS_0, & dActualPosition); //0축의 현재 위치를 확인
                    AxmStatusGetCmdPos(AXIS_0, & dCommandPosition);
                    //0축의 지령 위치를 확인.
                    AxmStatusReadPosError(AXIS_0, & dError);
                    //0축의 지령 위치와 실제위치의 차이를 확인.
                    AxmStatusReadVel(AXIS_0, & dVelocity);      //0축의 현재 속도를 확인.

                    printf("₩rAP:%.1f, CP:%.1f, Error:%.1f, CV:%.1f ",
                            dActualPosition, dCommandPosition, dError, dVelocity );

                } //End of While()

                printf("₩n구동을 완료하였습니다.₩n");
                AxmSignalServoOn(AXIS_0, DISABLE);    //Servo Off
            }
            else
                printf ("AxmInfoIsMotionModule() : ERROR ( NOT STATUS_EXIST )
                        code 0x%x₩n",Code);
        }
        else
            printf ("AxmInfoIsMotionModule() : ERROR ( Return FALSE )    code 0x%x₩n",Code);

    }
```

```
        else
            printf ("AxlOpen() : ERROR    code 0x%x₩n",Code);

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

다축 구동

일반적으로 장비에서의 구동시스템은 여러 축을 한꺼번에 일정한 위치로 이동시켜야 하거나, 일정한 패턴으로 각 축들이 서로 연관된 운동을 해야 하는 경우도 많이 있다. 이러한 구동을 위해서 다축구동 함수를 제공한다.
다축구동을 위한 설정은 앞에서 해왔던 단축 구동설정을 모든 축에 대해서 각각 설정 해주면 된다. 예를 들어 축이 2개가 있다면, 단축구동함수에서의 설정을 0축과 1축에 대해서 각각 설정해주면 되는 것이다.
다축구동 함수는 여러 축을 설정된 위치와 속도로 한꺼번에 구동이다.

1) 다축위치구동
2) 보간 구동

1) 다축위치구동

다축위치구동은 이미 다루었던 단축구동에서의 위치구동이 두 축에서 동시에 이루어지는 것이라고 볼 수 있다. 현재 위치에서 입력해준 각 축의 종료위치로 축들을 동시에 출발시키는데, 두 축의 이동 속도나 가속도 등은 개별 설정이 가능하다. 이러한 다축 위치구동은 종점탈출 함수인 AxmMoveMultiPos 함수 시점탈출 함수인 AxmMoveStartMultiPos 함수가 있다.
두 함수는 시점탈출인지 종점탈출인지의 차이만 있을 뿐 입력되는 인자는 동일하다. 여기서 축번호는 반드시 오름차순으로 순서대로 입력해야 한다. 아래의 예제는 0축과 1축이 각각 100과 50의 위치로 동일한 속도와 가속도로 이동하도록 하는 예제이다. 위치구동함수는 보간구동이 아니며, 각각의 축에 설정된 위치, 속도, 가속도 등의 값에 따라 개별적으로 움직인다. 단지 출발만 동시에 한다는 것 외에 차이점이 없다. 당연히 1축이 0축보다 목표지점에 먼저 도달하게 되며 1축은 정지된 상태에서 0축은 계속 움직이게 된다. 종점탈출함수인 AxmMoveMultiPos 함수는 명령을 내린 모든 축이 정지해야 함수에서 빠져 나와 프로그램상의 다음 명령을 실행한다.

```
//0축을 상대좌표로 (100,50)위치까지 대칭 S자 구동모드로 이동 (속도 100, 가속도 200)
DWORD uAbsRelMode = 0;
DWORD uProfileMode = 3;
AxmMotSetAbsRelMode(lAxisNo, uAbsRelMode);
AxmMotSetProfileMode(lAxisNo, uProfileMode);
```

```
long lArraySize = 2;
long lAxisNo[2] = {0,1};
double dPosition[2] = {100,50};
double dMaxVelocity[2] = {100,100};
double dMaxAccel[2] = {200,200};
double dMaxDecel[2] = {200,200};
AxmMoveMultiPos(lArraySize,lAxisNo,dPosition,dMaxVelocity,dMaxAccel,dMaxDecel);
```

시점탈출 함수인 AxmMoveStartMultiPos 함수를 사용하고자 한다면 다음과 같이 한다. 시점탈출이므로 구동이 시작되고 바로 다음 함수가 실행되며, 구동의 종료여부를 확인하고자 한다면 각 축에 대해서 AxmStatusReadInMotion함수를 사용한다.

```
//0축을 상대좌표로 (100,50)위치까지 대칭 S자 구동모드로 이동 (속도 100, 가속도 200)
DWORD uAbsRelMode = 0;
DWORD uProfileMode = 3;
AxmMotSetAbsRelMode(lAxisNo, uAbsRelMode);
AxmMotSetProfileMode(lAxisNo, uProfileMode);

long lArraySize = 2;
long lAxesNo[2] = {0,1};
double dPosition[2] = {100,50};
double dMaxVelocity[2] = {100,100};
double dMaxAccel[2] = {200,200};
double dMaxDecel[2] = {200,200};
AxmMoveStartMultiPos(lArraySize, lAxesNo, dPosition, dMaxVelocity, dMaxAccel, dMaxDecel);

//모션이 모두 종료될 때까지 기다린다.
BOOL WaitForDone = TRUE;
```

```
DWORD uStatus1, uStatus2;
while(WaitForDone)
{
    AxmStatusReadInMotion(lAxesNo[0], &uStatus1);
AxmStatusReadInMotion(lAxesNo[1], &uStatus2);
    If(uStatus1 == 0 && uStatus2 == 0)
    WaitForDone = FALSE;
}
```

```
// Ex11_AXM_MultiPositionDrive.cpp: Defines the entry point for the console application.
// 2축 기구부의 원점을 잡은 후 (100,50)위치까지 이동하다가 다시 원점으로 복귀한다.
#include "stdafx.h"
#include "AXL.h"
#include <conio.h>
#include "stdio.h"

#define AXIS_0 0
#define AXIS_1 1

void main(void)
{
    // 라이브러리를 초기화 한다.
    DWORD Code = AxlOpen();
    if (Code == AXT_RT_SUCCESS)
    {
        printf("라이브러리가 초기화 되었습니다.₩n");

        //모션모듈이 있는지 검사
```

```
DWORD dwStatus;
Code = AxmInfoIsMotionModule(&dwStatus);
if(Code == AXT_RT_SUCCESS)
{
    if(dwStatus == STATUS_EXIST)
    {
        printf("모션모듈이 존재 합니다.₩n");

        for(int nAxisNo = 0; nAxisNo<2; nAxisNo++)
        {
        // 0축의 +End limit 과 -End limit 의 Active level 을 HIGH와 급정지로 설정한다.
            AxmSignalSetLimit(nAxisNo, 0, HIGH, HIGH);
            // 0축의 Inpositon 신호의 입력 레벨을 HIGH로 설정한다.
            AxmSignalSetInpos(nAxisNo, HIGH);
            // 0축의 알람 신호의 입력 레벨을 LOW로 설정한다.
            AxmSignalSetServoAlarm(nAxisNo, LOW);
            // 0축의 비상정지신호(ESTOP)의 Active 입력레벨을 HIGH로 설정한다.
            AxmSignalSetStop(nAxisNo, 0, HIGH);

            //0축 지령값을 mm단위가 되도록 한다.
            AxmMotSetMoveUnitPerPulse(nAxisNo, 10, 10000);
            //0축에 초기 속도를 1로 설정한다. Default : 1
            AxmMotSetMinVel(nAxisNo, 1);
            //0축의 펄스 출력 방식을 TwoCwCcwHigh로 한다.
            AxmMotSetPulseOutMethod(nAxisNo, TwoCcwCwHigh);
            //지정 축에 Encoder 입력 방식을 4체배로 설정한다.
            AxmMotSetEncInputMethod(nAxisNo, ObverseSqr4Mode);

            // 원점검색 방법을 설정
```

```
        AxmHomeSetMethod(nAxisNo,-1,HomeSensor,1,2000.0,0.0);
        // 원점검색시 사용될 속도를 설정
        AxmHomeSetVel(nAxisNo, 100, 20, 10, 10, 200, 40);

        //Servo On
        AxmSignalServoOn(nAxisNo, ENABLE);
}

long lArraySize = 2;
long lAxesNo[2] = {0,1};
double dPosition[2], dMaxVelocity[2], dMaxAccel[2], dMaxDecel[2];

printf("₩n사다리꼴 속도 프로파일로 0축은 (+)100만큼, 1축은 (+)50만큼 이동
    (종점탈출)₩n");
printf("아무 키나 누르면 구동을 시작합니다.₩n");
getch();
for(nAxisNo = 0; nAxisNo<1; nAxisNo++)
{
        AxmMotSetAbsRelMode(nAxisNo, 1);   //상대좌표로 설정
        AxmMotSetProfileMode(nAxisNo, 0);    //대칭 사다리꼴 속도프로파일로 설정
}
dPosition[0] = 100;          dPosition[1] =50;
dMaxVelocity[0] = 100;       dMaxVelocity[1] =100;
dMaxAccel[0] = 200;          dMaxAccel[1] = 200;
dMaxDecel[0] = 200;          dMaxDecel[1] =200;
AxmMoveMultiPos(lArraySize, lAxesNo, dPosition, dMaxVelocity, dMaxAccel,
    dMaxDecel);
printf("구동을 완료하였습니다.₩n");
```

```
printf("S자 속도 프로파일로 원점으로 이동 (시점탈출)₩n");
printf("아무 키나 누르면 구동을 시작합니다.");
getch();
for(nAxisNo = 0; nAxisNo<1; nAxisNo++)
{
    AxmMotSetAbsRelMode(nAxisNo, 0);    //절대좌표로 설정
    AxmMotSetProfileMode(nAxisNo, 3);    //대칭 S자 속도프로파일로 설정
}
dPosition[0] = 0;            dPosition[1] =0;
dMaxVelocity[0] = 100;       dMaxVelocity[1] =100;
dMaxAccel[0] = 200;          dMaxAccel[1] = 200;
dMaxDecel[0] = 200;          dMaxDecel[1] =200;
AxmMoveStartMultiPos(lArraySize, lAxesNo, dPosition, dMaxVelocity, dMaxAccel,
    dMaxDecel);

//모션이 모두 종료될 때까지 기다린다.
BOOL WaitForDone = TRUE;
DWORD uStatus0, uStatus1;
while(WaitForDone) {
    AxmStatusReadInMotion(AXIS_0, &uStatus0);
    AxmStatusReadInMotion(AXIS_1, &uStatus1);
    if(uStatus0 == 0 && uStatus1 == 0)
        WaitForDone = FALSE;
}
printf("구동을 완료하였습니다.₩n");

//Servo Off
AxmSignalServoOn(0, DISABLE);
AxmSignalServoOn(1, DISABLE);
```

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

    }
        else
            printf ("AxlOpen() : ERROR    code 0x%x₩n",Code);

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

2) 보간 구동

앞에서 언급한 다축 위치구동은 두 축이 동시에 출발해서 각각의 동작을 수행하는 다축구동이었다. 하지만, 두 축이 움직이는데 있어서 상호 연관된 움직임을 통해 직선 또는 원호의 지정된 경로를 따라 이동해야 하는 경우가 발생하는데, 이러한 구동을 위해 직선보간, 원호보간 등을 제공한다. 이러한 보간구동의 속도프로파일 및 절대/상대좌표 구동에 대한 설정은 보간구동되는 축들 중에 가장 축번호가 빠른 축을 따라 가도록 되어 있으므로 설정에 주의하기 바란다.

● *직선 보간 구동*

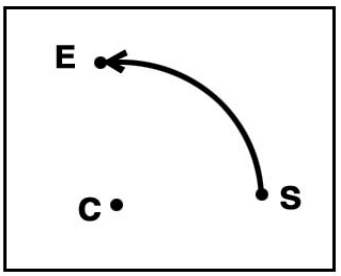
직선 보간구동 함수는 AxmLineMove 함수를 사용하여 하나의 모듈에 포함된 두 개의 축을 직선 보간구동 시킨다. EIP-M2Q는 2개의 축, EIP-M4Q는 4개의 축이 지원 되므로, 최대 지원 할 수 있는 모든 축의 직선 보간이 가능하다. 이 함수는 현재 위치로부터 입력된 종료점까지 특정 속도 및 가속도로 구동하도록 두 축이 적절히 구동한다.

```
//상대위치 (100,50)까지 S자 구동으로 보간구동
long lSize = 2;
long lAxesNo = {0,1};
double dPosition = {100,50};
double dMaxVelocity = 100;
double dMaxAccel = 200;
double dMaxDecel = 200;
long lCoordinate = 0;

DWORD uAbsRelMode = 1;
DWORD uProfileMode = 3;

AxmContiSetAxisMap(lCoordinate, lSize, lAxesNo);
AxmContiSetAbsRelMode (lCoordinate, uAbsRelMode);     // 상대위치구동으로 설정

AxmLineMove(lCoordinate, dPosition, dMaxVelocity, dMaxAccel, dMaxDecel);
```

● *원호 보간 구동*

원호 보간구동은 현재 위치에서 지정된 위치까지 해당 설정에 따라 원호 경로로 보간구동하도록 되어 있는데, 이러한 원호 보간구동 함수들은 AxmCircleCenterMove 함수, AxmCircleRadiusMove 함수, AxmCircleAngleMove 함수 그리고 AxmCirclePointMove함수를 제공한다.

● *AxmCircleCenterMove*

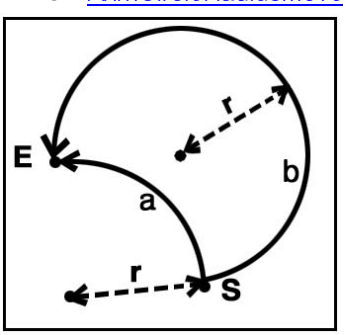


AxmCircleCenterMove 함수는 종료점(E)과 중심점(C)을 입력받아서 현재위치(S)에서부터 원호 보간구동을 한다. 세 점의 위치를 이용하여 원호를 계산하는 방법이므로 정확한 중심점과 종료점을 입력하여야 원을 구성할 수 있고, 함수가 작동된다. 만약 잘못된 좌표를 입력하였다면 함수가 작동되지 않고 FALSE를 반환한다. 현재 위치(0,0)에서 (100,100)위치까지 (0,100)을 중심으로 반시계방향으로 원호 보간 구동을 하고자 한다면, 아래와 같이 한다.

```
// 현재위치(0,0)에서 (100,100)위치까지 (0,100)을 중심으로 원호보간을 수행한다.
long lSize = 2;
long lAxesNo = {0,1};
double   dCenPos[2];
double   dEndPos[2];
long     lCoordinate = 0;
DWORD uAbsRelMode = 1;
AxmContiSetAxisMap(lCoordinate, lSize, lAxesNo);
AxmContiSetAbsRelMode (lCoordinate, uAbsRelMode);      // 상대위치구동으로 설정

double dCenterXPosition = 0;
double dCenterYPosition = 100;
double dEndXPosition = 100;
double dEndYPosition = 100;

dCenPos[0] = dCenterXPosition;
dCenPos[1] = dCenterYPosition;
```

```
dEndPos[0] = dEndXPosition;
dEndPos[1] = dEndYPosition;

double dMaxVelocity = 100;
double dMaxAccel = 200;
double dMaxDecel = 200;
DWORD uCWDir = 1;            //[0] Ccw방향, [1] Cw방향
AxmCircleCenterMove(lCoordinate, lAxesNo, dCenPos, dEndPos, dMaxVelocity,
                    dMaxAccel, dMaxDecel, uCWDir);
```

- *AxmCircleRadiusMove*

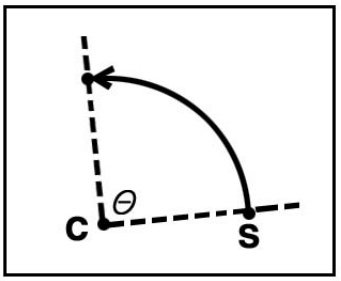


AxmCircleRadiusMove 함수는 종료점(E)과 반지름(r)을 입력받아서 현재위치(S)에서부터 원호보간구동을 한다. 두 점을 지나고 일정한 반지름을 가지는 호를 계산하는 방법이므로 두 개의 원을 구성할 수 있게 된다. 그래서 입력받는 반지름에 대해서 uShortDistance의 값이 0이면, 작게 도는 원(a)이 선택되고, 1이면 크게 도는 원(b)이 선택된다.
현재 위치에서 (300,200)위치까지 반지름이 120인 시계방향으로 크게 도는 원호보간을 수행하고자 한다면 다음과 같이 한다.

```
// 현재 위치에서 (300,200)위치까지 반지름 120인 크게 도는 원호보간을 수행
long lSize = 2;
long lAxesNo = {0,1};
```

```
double   dEndPos[2];
long     lCoordinate = 0;

DWORD uAbsRelMode = 1;
DWORD uProfileMode = 3;
AxmContiSetAxisMap(lCoordinate, lSize, lAxesNo);
AxmContiSetAbsRelMode (lCoordinate, uAbsRelMode);      // 상대위치구동으로 설정

double dRadius = 120;
double dEndXPosition = 300
double dEndYPosition = 200;
dEndPos[0] = dEndXPosition;
dEndPos[1] = dEndYPosition;
double dMaxVelocity = 100;
double dMaxAccel = 200;
double dMaxDecel = 200;
DWORD uCWDir = 0;                    //[0] Cw방향, [1] Ccw방향
DWORD uShortDistance = 1;            //[0]: 작은(원)거리 ,[1] 큰(원)거리
AxmCircleRadiusMove(lCoordinate, lAxesNo, dRadius, dEndPos,
                    dMaxVelocity, dMaxAccel, dMaxDecel, uCWDir, uShortDistance);
```

● *AxmCircleAngleMove*

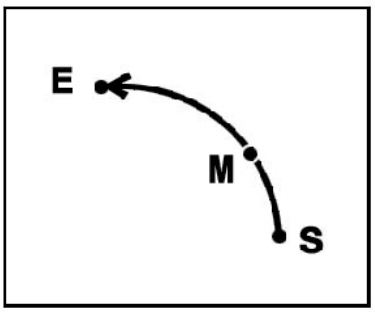


AxmCircleAngleMove 함수는 중심점(C)과 회전각(θ)을 입력받아서 현재위치(S)에서부터 원호 보간구동을 한다. 호의 반지름과 각도를 이용하여 원호를 구하는 방법으로 하나의 원이 구성된다.
현재 위치에서 (200,200)을 중점으로 200도 만큼의 원호보간을 수행하는데, 시계방향으로 보간구동 하고자 한다면 아래와 같이 한다.

```
// 현재 위치에서 (200,200)을 중심으로 200도만큼 원호보간을 수행한다.
long lSize = 2;
long lAxesNo = {0,1};
double   dCenPos[2];
long     lCoordinate = 0;

DWORD uAbsRelMode = 1;
DWORD uProfileMode = 3;
AxmContiSetAxisMap(lCoordinate, lSize, lAxesNo);
AxmContiSetAbsRelMode (lCoordinate, uAbsRelMode);      // 상대위치구동으로 설정

double dCenterXPosition = 200;
double dCenterYPosition = 200;

dCenPos[0] = dCenterXPosition;
dCenPos[1] = dCenterYPosition;
```

```
double dAngle = 200;
double dMaxVelocity = 100;
double dMaxAccel = 200;
double dMaxDecel = 200;
DWORD uCWDir = 0;                    //[0] Cw방향, [1] Ccw방향
AxmCircleAngleMove(lCoordinate, lAxesNo, dCenPos, dAngle,
                   dMaxVelocity, dMaxAccel, dMaxDecel, uCWDir);
```

● *AxmCirclePointMove*



AxmCirclePointMove 함수는 중간점(M)과 종료점(E)을 입력받아서 현재위치(S)에서부터 원호 보간구동을 한다. 세 점을 지나는 원호를 구하는 방법으로 하나의 원이 구성된다.
현재 위치에서 (100,50)을 지나면서 (150,150)까지 보간구동 하고자 한다면 아래와 같이 한다.

```
// 현재 위치에서 (100,50)을 지나  (150,150)까지 원호보간을 수행한다.
long lSize = 2;
long lAxesNo = {0,1};
double   dMidPos[2];
double   dEndPos[2];
long     lCoordinate = 0;
DWORD uAbsRelMode = 1;
```

```
DWORD uProfileMode = 3;
AxmContiSetAxisMap(lCoordinate, lSize, lAxesNo);
AxmContiSetAbsRelMode (lCoordinate, uAbsRelMode);      // 상대위치구동으로 설정

double dMidXPosition = 100;
double dMidYPosition = 50;
double dEndXPosition = 150;
double dEndYPosition = 150;

dMidPos[0] = dMidXPosition;
dMidPos[1] = dMidYPosition;

dEndPos[0] = dEndXPosition;
dEndPos[1] = dEndYPosition;

double dMaxVelocity = 100;
double dMaxAccel = 200;
double dMaxDecel = 200;
AxmCirclePointMove(lCoordinate, lAxesNo, dMidPos, dEndPos,
                   dMaxVelocity, dMaxAccel, dMaxDecel);
```

```
// Ex12_AXM_Interpolation.cpp : Defines the entry point for the console application.
// 2축 기구부의 원점을 잡은 후 보간구동을 시작한다.
#include "stdafx.h"
#include "AXL.h"
#include <conio.h>
#include "stdio.h"
```

```
#define AXIS_0 0
#define AXIS_1 1

void main(void)
{
   // 라이브러리를 초기화 한다.
   DWORD Code = AxlOpen();
   if (Code == AXT_RT_SUCCESS)
   {
         printf("라이브러리가 초기화 되었습니다.₩n");

         //모션모듈이 있는지 검사
         DWORD dwStatus;
         Code = AxmInfoIsMotionModule(&dwStatus);
         if(Code == AXT_RT_SUCCESS)
         {
               if(dwStatus == STATUS_EXIST)
               {
                     printf("모션모듈이 존재 합니다.₩n");
                     for(int nAxisNo = 0; nAxisNo<2; nAxisNo++)
                     {
                     // 0축의 +End limit 과 -End limit 의 Active level 을 HIGH와 급정지로 설정한다.
                           AxmSignalSetLimit(nAxisNo, 0, HIGH, HIGH);
                           // 0축의 Inpositon 신호의 입력 레벨을 HIGH로 설정한다.
                           AxmSignalSetInpos(nAxisNo, HIGH);
                           // 0축의 알람 신호의 입력 레벨을 LOW로 설정한다.
                           AxmSignalSetServoAlarm(nAxisNo, LOW);
                           // 0축의 비상정지신호(ESTOP)의 Active 입력레벨을 HIGH로 설정한다.
                           AxmSignalSetStop(nAxisNo, 0, HIGH);
```

```
        //0축 지령값을 mm단위가 되도록 한다.
        AxmMotSetMoveUnitPerPulse(nAxisNo, 10, 10000);
        //0축에 초기 속도를 1로 설정한다. Default : 1
        AxmMotSetMinVel(nAxisNo, 1);
        //0축의 펄스 출력 방식을 TwoCwCcwHigh로 한다.
        AxmMotSetPulseOutMethod(nAxisNo, TwoCcwCwHigh);
        //지정 축에 Encoder 입력 방식을 4체배로 설정한다.
        AxmMotSetEncInputMethod(nAxisNo, ObverseSqr4Mode);

        // 원점검색 방법을 설정
        AxmHomeSetMethod(nAxisNo,-1,HomeSensor,1,2000.0,0.0);
        // 원점검색시 사용될 속도를 설정
        AxmHomeSetVel(nAxisNo, 100, 20, 10, 10, 200, 40);

        //Servo On
        AxmSignalServoOn(nAxisNo, ENABLE);
    }

    //원점을 검색한다.
    printf("아무 키나 누르면 원점 검색을 시작합니다.₩n");
    getch();
    AxmHomeSetStart(AXIS_0);
    AxmHomeSetStart(AXIS_1);

    //현재 원점검색 진행률을 확인
    DWORD uHomeResult_0, uHomeResult_1;
    DWORD uHomeMainStepNumber , uHomeStepNumber_0,uHomeStepNumber_1;
    AxmHomeGetResult(AXIS_0, &uHomeResult_0);
```

```
AxmHomeGetResult(AXIS_1, &uHomeResult_1);
while(uHomeResult_0 == HOME_SEARCHING || uHomeResult_1
        == HOME_SEARCHING)
{
 AxmHomeGetResult(AXIS_0, &uHomeResult_0);
 AxmHomeGetResult(AXIS_1, &uHomeResult_1);

 AxmHomeGetRate(AXIS_0, &uHomeMainStepNumber , &uHomeStepNumber_0);
 AxmHomeGetRate(AXIS_1, &uHomeMainStepNumber , &uHomeStepNumber_1);
      printf("₩r원점검색중... 0축 : %d%, 1축 : %d",
            uHomeStepNumber_0, uHomeStepNumber_1);
}
printf("₩n상대위치(150,0)까지 직선보간 구동한다.₩n");
printf("아무 키나 누르면 구동을 시작합니다.₩n");
getch();
long    lSize = 2;
long    lpAxesNo[2] = {AXIS_0, AXIS_1};
double dpPosition[2] = {150, 0};
double dMaxVelocity = 50;
double dMaxAccel = 100;
double dMaxDecel = 100;
long    lCoordinate = 0;
double dCenPos[2];
double dMidPos[2];
double dEndPos[2];

AxmContiSetAxisMap(lCoordinate, lSize, lpAxesNo);
AxmContiSetAbsRelMode (lCoordinate, 1);     // 상대위치구동으로 설정
AxmLineMove(lCoordinate, dpPosition, dMaxVelocity, dMaxAccel, dMaxDecel);
```

```
BOOL WaitForDone = TRUE;
DWORD uStatus0, uStatus1;
while(WaitForDone) {
    AxmStatusReadInMotion(AXIS_0, &uStatus0);
    AxmStatusReadInMotion(AXIS_1, &uStatus1);
    if(uStatus0 == 0 && uStatus1 == 0)
        WaitForDone = FALSE;
}
printf("₩n상대위치(50,50)까지 (0,50)을 중심으로 Ccw방향으로
    원호보간 구동한다.₩n");
printf("아무 키나 누르면 구동을 시작합니다.₩n");
getch();

dCenPos[0] = 0;     dCenPos[1] = 50;
dEndPos[0] = 50;    dEndPos[1] = 50;
AxmCircleCenterMove(lCoordinate, lpAxesNo, dCenPos,
                dEndPos, dMaxVelocity, dMaxAccel, dMaxDecel, 1);

//모션이 모두 종료될 때까지 기다린다.
WaitForDone = TRUE;
while(WaitForDone) {
    AxmStatusReadInMotion(AXIS_0, &uStatus0);
    AxmStatusReadInMotion(AXIS_1, &uStatus1);
    if(uStatus0 == 0 && uStatus1 == 0)
        WaitForDone = FALSE;
}

printf("₩n상대위치(0,100)까지 직선보간 구동한다.₩n");
```

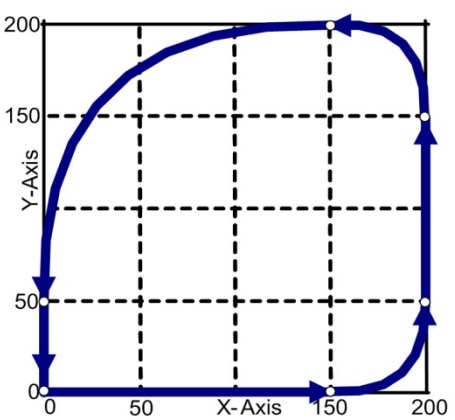
```
printf("아무 키나 누르면 구동을 시작합니다.₩n");
getch();
dpPosition[0] = 0; dpPosition[1] = 100;
AxmLineMove(lCoordinate, dpPosition, dMaxVelocity, dMaxAccel, dMaxDecel);
//모션이 모두 종료될 때까지 기다린다.
WaitForDone = TRUE;
while(WaitForDone) {
    AxmStatusReadInMotion(AXIS_0, &uStatus0);
    AxmStatusReadInMotion(AXIS_1, &uStatus1);
    if(uStatus0 == 0 && uStatus1 == 0)
        WaitForDone = FALSE;
}
printf("₩n상대위치(-50,50)까지 Ccw방향으로 반지름이 50인 작게
        또는 원호보간 구동한다.₩n");
printf("아무 키나 누르면 구동을 시작합니다.₩n");
getch();

dEndPos[0] = 50; dEndPos[1] = -50;
 AxmCircleRadiusMove(lCoordinate, lpAxesNo, 50, dEndPos,
dMaxVelocity, dMaxAccel, dMaxDecel, 0, 1);

//모션이 모두 종료될 때까지 기다린다.
WaitForDone = TRUE;
while(WaitForDone) {
    AxmStatusReadInMotion(AXIS_0, &uStatus0);
    AxmStatusReadInMotion(AXIS_1, &uStatus1);
    if(uStatus0 == 0 && uStatus1 == 0)
        WaitForDone = FALSE;
}
```

```
printf("₩n상대위치(0,-150)을 중심으로 Ccw방향으로 90'만큼
        원호보간 구동한다.₩n");
printf("아무 키나 누르면 구동을 시작합니다.₩n");
getch();

//모션이 모두 종료될 때까지 기다린다.
dCenPos[0] = 0; dCenPos[1] = -150;
 AxmCircleAngleMove(lCoordinate, lpAxesNo, dCenPos, 90,
                    dMaxVelocity, dMaxAccel, dMaxDecel, 1);

  //모션이 모두 종료될 때까지 기다린다.
WaitForDone = TRUE;
while(WaitForDone) {
      AxmStatusReadInMotion(AXIS_0, &uStatus0);
      AxmStatusReadInMotion(AXIS_1, &uStatus1);
      if(uStatus0 == 0 && uStatus1 == 0)
            WaitForDone = FALSE;
}

printf("₩n상대위치(50,50)을 중간점으로 Ccw방향으로 (100,0)만큼
        원호보간 구동한다.₩n");
printf("아무 키나 누르면 구동을 시작합니다.₩n");
getch();

//모션이 모두 종료될 때까지 기다린다.
dMidPos[0] = 50;      dMidPos[1] = 50;
dEndPos[0] = 100;     dEndPos[1] = 0;
    AxmCirclePointMove(lCoordinate, lpAxesNo, dMidPos, dEndPos,
```

```
                dMaxVelocity, dMaxAccel, dMaxDecel,0);

        WaitForDone = TRUE;
        while(WaitForDone) {
            AxmStatusReadInMotion(AXIS_0, &uStatus0);
            AxmStatusReadInMotion(AXIS_1, &uStatus1);
            if(uStatus0 == 0 && uStatus1 == 0)
                WaitForDone = FALSE;
        }

        printf("₩n원점까지 직선보간 구동한다.₩n");
        printf("아무 키나 누르면 구동을 시작합니다.₩n");
        getch();
        dpPosition[0] = 0; dpPosition[1] = -50;
        AxmLineMove(lCoordinate, dpPosition, dMaxVelocity, dMaxAccel, dMaxDecel);
        //모션이 모두 종료될 때까지 기다린다.
        WaitForDone = TRUE;
        while(WaitForDone) {
            AxmStatusReadInMotion(AXIS_0, &uStatus0);
            AxmStatusReadInMotion(AXIS_1, &uStatus1);
            if(uStatus0 == 0 && uStatus1 == 0)
                WaitForDone = FALSE;
        }

        //Servo Off
        AxmSignalServoOn(AXIS_0, ENABLE);
        AxmSignalServoOn(AXIS_1, ENABLE);
    }
    else
```

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

    }
    else
        printf ("AxlOpen() : ERROR    code 0x%x₩n",Code);

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

※ 위의 직선 보간구동 예제를 실행하면 다음과 같은 결과를 확인할 수 있다.



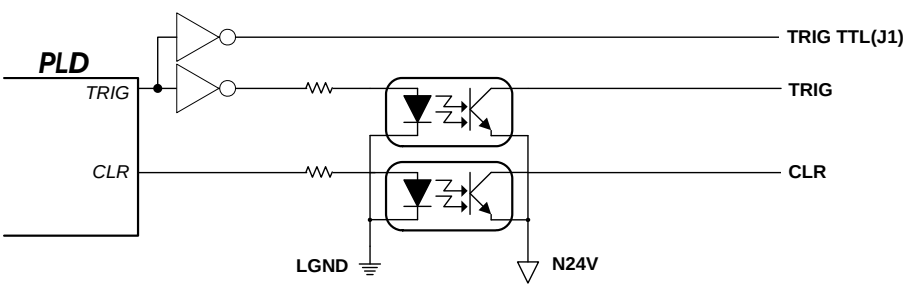
확장 모션구동

지금까지 일반적인 모션제어를 위한 여러 함수들에 대해서 살펴보았다. 본 장에서는 고급 모션구동을 위한 함수들과 흔히 사용하지는 않지만, 꼭 필요한 몇 가지 기능에 대해서 살펴보기로 한다.

CRC 신호 설정
오버라이드 구동
MPG 구동 (Manual Pulse Generation)
동기구동
갠트리 구동
1) 겐트리 구동 설정 및 해제
2) 갠트리 구동에서의 홈 검색
가상축 매핑
트리거 신호의 사용

CRC 신호 설정

특정 서보팩의 경우 구동이 완료된 시점에서 또는 리미트센서를 감지한 경우에 외부에서 CRC(Current Remaining Clear)신호를 받아 서보팩이 가지고 있는 잔여 펄스를 지워주어야 하는 경우가 발생한다. 이를 위해 신호핀을 할당하여 서보팩에 직접 연결할 수 있도록 제공하고 있으며, CRC신호의 사용여부, 출력레벨, 리미트 신호 사용여부 등을 설정할 수 있도록 함수로 제공하고 있다. 또한 임의로 사용자가 CRC신호를 발생시킬 수도 있도록 함수를 제공한다.
출력 레벨은 칩 출력의 반전된 레벨이 나오게 된다. 즉, 칩 출력 레벨이 논리 '1'일 때, CLR출력은 N24V로, 논리 '0'일 때의 출력은 Open Collector이다. 이 신호 모두 범용출력과 마찬가지로 Photo-Coupler로 외부 신호와 전기적으로 절연되어 있다. 이 신호들은 외부의 특정 위치 또는 외부의 특정 조건에 의해 출력을 내게 되는데, 시간 지연은 Photo-Coupler를 거치는 시간과, PLD내부 회로의 시간 지연에 의한 것이다. PELM, NELM을 입력으로 받는 CLR신호의 경우 약 6㎲~8시간지연이 발생된다. Photo Coupler 출력은 Normally Open 형태의 출력이므로, 24V레벨의 전압을 얻기 위해서는 사용자는 Pull-up저항을 붙여야 한다. Pull-up저항은 약 10㎏의 저항을 사용한다.



● *EndLimit 설정함수*

기구부가 End Limit에 걸렸을 때 자동으로 CRC신호를 발생시키도록 설정할 수 있다. 이를 위해 CRC신호의 레벨을 설정하는 AxmCrcSetMaskLevel 함수와 각 EndLimit 센서에 해당하는 CRC 신호레벨을 설정하는 AxmCrcSetEndLimit함수가 사용된다.

```
//0축에 CRC을 사용하기 위해서 초기 셋팅해준다. 레벨 설정이 잘못되면 모터가 움직이지 않을수 있다.

//0축에 CRC 신호 사용 여부를 설정한다
```

```
long lAxisNo = 0;
DWORD uLevel = HIGH;
AxmCrcSetMaskLevel(lAxisNo, uLevel, 6); //CRC 신호폭 100msec설정
//0축에 리미트/알람/비상정지/동기정지 신호에 의한 정지시 서버 잔여 펄스 제거 신호 출력 사용유무
// PositiveLevel , NegativeLevel 사용안함
AxmCrcSetEndLimit(lAxisNo, PositiveLevel, NegativeLevel);
```

● *Program으로 강제출력 설정 함수*

리미트 센서에 걸렸을 때만 CRC신호가 필요한 것은 아니다. 상황에 따라서 CRC신호가 발생해 주어야 하는데, 이때 사용하는 함수가 AxmCrcSetOutput 함수이다.

```
//0축에 CRC을 사용하기 위해서 초기 셋팅해준다. 레벨 설정이 잘못되면 모터가 움직이지 않을 수 있다.

//0축에 CRC 신호 사용 여부를 설정한다
long lAxisNo = 0;
DWORD uLevel = HIGH;
AxmCrcSetMaskLevel(lAxisNo, uLevel);

//0축에 CRC 신호를 강제적으로 내보낸다.
DWORD uUse = ENABLE;          //  0: DISABLE, 1:ENABLE
AxmCrcSetOutput(lAxisNo, uUse);
```

```
// Ex14_AXM_AdvancedCRC.cpp : Defines the entry point for the console application.
// CRC 출력을 설정 및 확인.
```

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

#define AXIS_0 0

void main(void)
{
   // 라이브러리를 초기화 한다.
   DWORD Code = AxlOpen();
   if (Code == AXT_RT_SUCCESS)
   {
         printf("라이브러리가 초기화 되었다.₩n");

         //모션모듈이 있는지 검사
         DWORD dwStatus;
         Code = AxmInfoIsMotionModule(&dwStatus);
         if(Code == AXT_RT_SUCCESS)
         {
               if(dwStatus == STATUS_EXIST)
               {
                     printf("모션모듈이 존재 한다.₩n");

                     // 0축의 +End limit 과 -End limit 의 Active level 을 HIGH로 설정한다.
                     AxmSignalSetLimit(AXIS_0, 0, HIGH, HIGH);
                     // 0축의 Inpositon 신호의 입력 레벨을 HIGH로 설정한다.
                     AxmSignalSetInpos(AXIS_0, HIGH);
```

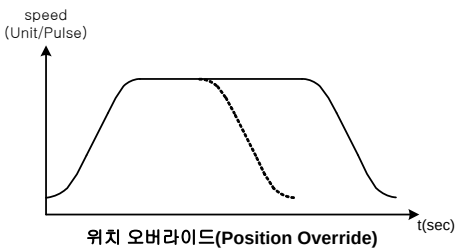
```
// 0축의 알람 신호의 입력 레벨을 LOW로설정한다.
AxmSignalSetServoAlarm(AXIS_0, LOW);
// 0축의 비상정지신호(ESTOP)의 Active 입력레벨을 HIGH로 설정한다.
AxmSignalSetStop(AXIS_0, 0, HIGH);

//0축 지령값을 mm단위가 되도록 한다.
AxmMotSetMoveUnitPerPulse(AXIS_0, 10, 10000);
//0축에 초기 속도를 1로 설정한다. Default : 1
AxmMotSetMinVel(AXIS_0, 1);
//0축의 펄스 출력 방식을 TwoCwCcwHigh로 한다.
AxmMotSetPulseOutMethod(AXIS_0, TwoCcwCwHigh);
//지정 축에 Encoder 입력 방식을 4체배로 설정한다.
AxmMotSetEncInputMethod(AXIS_0, ObverseSqr4Mode);

//Servo On
AxmSignalServoOn(AXIS_0, ENABLE);

//0축에 CRC을 사용하기 위해서 초기 셋팅해준다. 레벨 설정이 잘못되면
  모터가 움직이지 않을수있다.
//0축에 CRC 신호 사용 여부를 설정한다
long lAxisNo = 0;
DWORD uLevel = HIGH;
AxmCrcSetMaskLevel(lAxisNo, uLevel, 6);
//0축에 리미트 신호에 대한 CRC 신호의 사용 여부를 설정한다
DWORD uPositiveLevel = HIGH;
DWORD uNegativeLevel = HIGH;

// help Message
printf("[INFORMATION]*************** ₩n");
```

```
printf("[ESC] : Exit ₩n");
printf("[1] : CRC HIGH로 출력 ₩n");
printf("[2] : CRC LOW로 출력 ₩n");
printf("****************************** ₩n");


DWORD uCRCOnOff = 2;
BOOL fExit = FALSE;
while (!fExit)    // 무한 루프
{
    if (kbhit())   // 아무키나 누르면
    {
        int ch = getch();
        switch (ch)
        {
        case 27:        // Esc key
            fExit = TRUE;
            AxmSignalServoOn(AXIS_0, DISABLE);   //Servo Off
        break;

            case '1': // CRC 신호를 프로그램적으로 강제적으로 HIGH 출력
                AxmCrcSetOutput(AXIS_0, ENABLE);
            break;

            case '2': // CRC 신호를 프로그램적으로 강제적으로 LOW 출력
                AxmCrcSetOutput(AXIS_0, DISABLE);
            break;
        }
    }
```

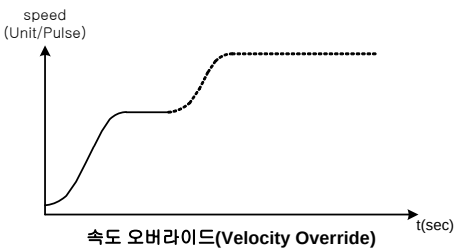
```
                        //현재의 CRC출력을 확인한다.

                        AxmCrcGetOutput(AXIS_0, &uCRCOnOff);
                        printf("₩rCRC 출력 : %x",uCRCOnOff);

                    } //End of While()
                }
                else
                    printf ("AxmInfoIsMotionModule() : ERROR ( NOT STATUS_EXIST )
                        code 0x%x₩n",Code);
            }
            else
                printf ("AxmInfoIsMotionModule() : ERROR ( Return FALSE )
                    code 0x%x₩n",Code);
        }
        else
            printf ("AxlOpen() : ERROR    code 0x%x₩n",Code);


        // 라이브러리를 종료 한다.
        if (AxlClose())
            printf("라이브러리가 종료 되었다.₩n");
        else
            printf("라이브러리가 정상적으로 종료되지 않았다.₩n");

    }
```

오버라이드 구동

이장에서는 속도 및 위치 오버라이드 구동에 대해서 소개한다. 속도 오버라이드 구동은 모션이 진행되는 도중에 작업 속도를 변경하는 것을 의미하고 위치 오버라이드 드라이브는 모션을 수행하고 있는 중에 목표거리를 변경하는 것을 의미한다.



위치 오버라이드(Position Override)



속도 오버라이드(Velocity Override)

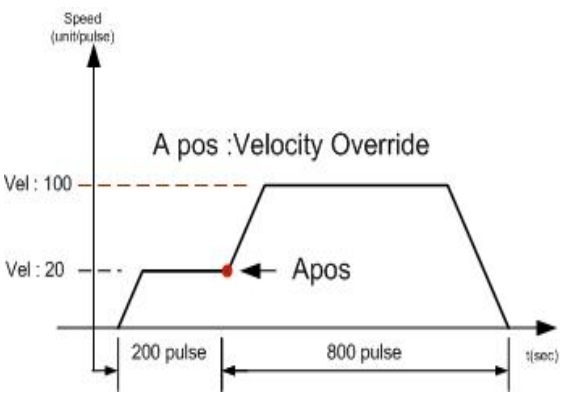
위치 오버라이드는 AxmOverridePos 함수를 이용하고, 속도 오버라이드는 AxmOverrideVel 함수를 이용한다. 오버라이드 함수는 반드시 구동중에 호출되어야 하며, 구동이 끝난 후에 호출되면 무시된다.

```
//0축을 절대좌표로 100위치까지 대칭 S자 구동모드로 이동 (속도 100, 가속도 200, 시점탈출)
long lAxisNo = 0;
AxmMotSetAbsRelMode(lAxisNo, 0);
AxmMotSetProfileMode(lAxisNo, 3);
// 속도를 처음 지정한 속도보다 크게 하기 위해서는 반드시 이 함수를 사용한다.
AxmMoveStartPos(lAxisNo, 100, 100, 200, 200);

AxmOverrideVel(lAxisNo, 500); // 속도를 500으로 조정
// 0축의 구동이 종료되기 전 절대 좌표2000으로 조정
double dOverridePosition = 2000;
AxmOverridePos(lAxisNo, dOverridePosition);

// 0축의 구동이 종료되기 전 속도를 50으로 조정
```

```
double dOverrideVelocity = 50;
AxmOverrideVel(lAxisNo, dOverrideVelocity);
```

AxmOverrideVelAtPos 함수는 어느 위치 지점과 오버라이드 할 속도를 입력시켜 그 위치에서 속도 오버라이드 되는 함수로 특정 위치를 지나면 오버라이드 되도록 사용할 수 있다. 이 함수는 구동 중에 변화시키는 함수가 아니라 정지상태에서 해당정보를 가지고 구동을 시작해야 한다는 것이 차이점이 있다.

```
// 0축을 1000까지 보내는데, 200위치를 지날 때 속도를 100으로 오버라이드 시킨다.
long lAxisNo = 0;
double dMaxVelocity = 20;
double dMaxAccel = 100;
double dMaxDecel = 100;
double dOverrideVelocity = 100;
double dOverridePosition = 200;
double dPosition = 1000;
long lTarget = 1;    // [0] Command Position , [1] Actual Position
AxmOverrideVelAtPos(lAxisNo, dPosition, dMaxVelocity, dMaxAccel, dMaxDecel,
dOverridePosition, dOverrideVelocity, lTarget);
```



```
// Ex16_AXM_OverrideDrive.cpp : Defines the entry point for the console application.
// Override Drive   Example

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

#define AXIS_0 0

void main(void)
{
   // 라이브러리를 초기화 한다.
   DWORD Code = AxlOpen();
   if (Code == AXT_RT_SUCCESS)
   {
         printf("라이브러리가 초기화 되었습니다.₩n");

         //모션모듈이 있는지 검사
         DWORD dwStatus;
         Code = AxmInfoIsMotionModule(&dwStatus);
         if(Code == AXT_RT_SUCCESS)
         {
               if(dwStatus == STATUS_EXIST)
               {
                     printf("모션모듈이 존재 합니다.₩n");
```

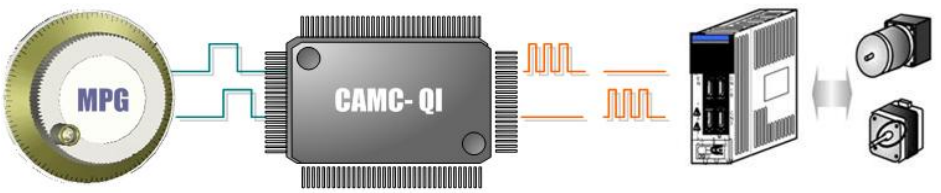
```
// 0축의 +End limit 과 -End limit 의 Active level 을 HIGH로 설정한다.
AxmSignalSetLimit(AXIS_0, 0, HIGH, HIGH);
// 0축의 Inpositon 신호의 입력 레벨을 HIGH로 설정한다.
AxmSignalSetInpos(AXIS_0, HIGH);
// 0축의 알람 신호의 입력 레벨을 LOW로 설정한다.
AxmSignalSetServoAlarm(AXIS_0, LOW);
// 0축의 비상정지신호(ESTOP)의 Active 입력레벨을 HIGH로 설정한다.
AxmSignalSetStop(AXIS_0, 0, HIGH);

//0축 지령값을 mm단위가 되도록 한다.
AxmMotSetMoveUnitPerPulse(AXIS_0, 10, 10000);
//0축에 초기 속도를 1로 설정한다. Default : 1
AxmMotSetMinVel(AXIS_0, 1);
//0축의 펄스 출력 방식을 TwoCwCcwHigh로 한다.
AxmMotSetPulseOutMethod(AXIS_0, TwoCcwCwHigh);
//지정 축에 Encoder 입력 방식을 4체배로 설정한다.
AxmMotSetEncInputMethod(AXIS_0, ObverseSqr4Mode);

//상대위치 S자 속도 프로파일을 사용
AxmMotSetAbsRelMode(AXIS_0, 1);
AxmMotSetProfileMode(AXIS_0, 3);

//Servo On
AxmSignalServoOn(AXIS_0, ENABLE);

// help Message
printf("[INFORMATION]********************************** ₩n");
printf("[ESC] : Exit ₩n");
printf("[1] :  상대좌표 100위치를 향해 속도 100으로 구동 시작 ₩n");
```

```
printf("[2] :  위치을 200으로 오버라이드 ₩n");
printf("[3] :  속도를 50으로 오버라이드  ₩n");
printf("[4] :  -200까지 구동을 시작하고 -100에서 속도를 50으로 오버라이드
        ₩n");
printf("************************************************ ₩n");


BOOL fExit = FALSE;
while (!fExit)    // 무한 루프
{
    if (kbhit())   // 아무 키나 누르면
    {
        int ch = getch();
        switch (ch)
        {
        case 27:        // Esc key
                fExit = TRUE;
                AxmSignalServoOn(AXIS_0, DISABLE);    //Servo Off
            break;
            case '1': //0축을 상대좌표로 100위치까지 대칭 S자 구동모드로
                 이동
                AxmMoveStartPos(AXIS_0, 100, 100, 200, 200);
            break;
            case '2': // 0축의 구동이 종료되기 전에 좌표200으로 조정
              AxmOverridePos(AXIS_0, 200);
            break;
            case '3': // 0축의 구동이 종료되기 전에 속도를 50으로 조정
                AxmOverrideVel(AXIS_0, 50);
            break;
```

```
                                case '4': // 0축이 200까지 구동을 시작하고 100위치가 되면 속
// 도를 50으로 오버라이드 한다.
                                AxmOverrideVelAtPos(AXIS_0, -200, 100, 200, 200, -100,50,
                                COMMAND);
                            }
                        }
                    } //End of While()
                }
                else
                    printf ("AxmInfoIsMotionModule() : ERROR ( NOT STATUS_EXIST )
                      code 0x%x₩n",Code);
            }
            else
                printf ("AxmInfoIsMotionModule() : ERROR ( Return FALSE )
                  code 0x%x₩n",Code);
        }
        else
            printf ("AxlOpen() : ERROR    code 0x%x₩n",Code);

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

}
```

MPG 구동 (Manual Pulse Generation)

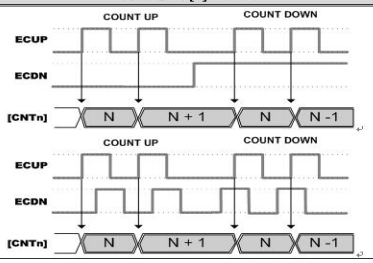
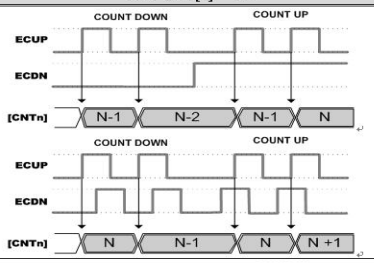
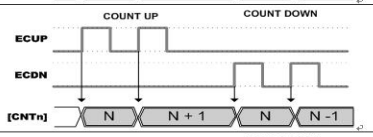
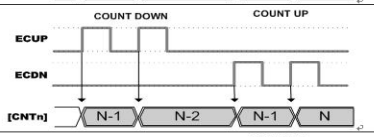
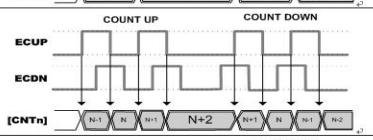
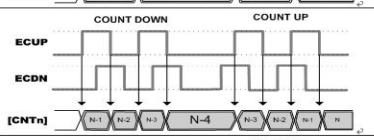
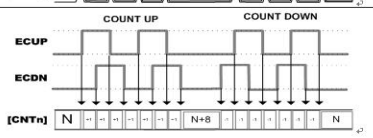
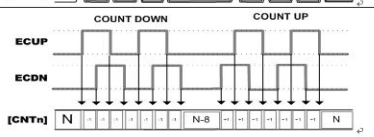
EXPP, EXMP 신호의 입력에 따라 방향 신호가 결정되고, 입력 사양 설정 레지스터에 따라 펄스 수가 결정된다. 출력되는 펄스의 속도는 설정된 Object Speed에 의해 출력된다.



● *EXPP/EXMP에 의한 속도구동*

Cont Mode는 EXPP/EXMP 신호가 Active Level이 되는 동안에 펄스를 출력한다. 구동 형태는 Velocity Drive와 같다. EXPP 신호가 Low('0')이고 EXMP가 High('1')인 동안은 (+)방향, EXMP 신호가 Low('0')이고 EXPP가 High('1')인 동안은 (–)방향으로 구동 한다.

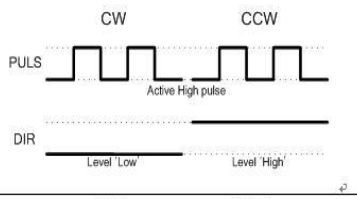
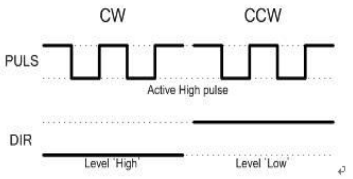
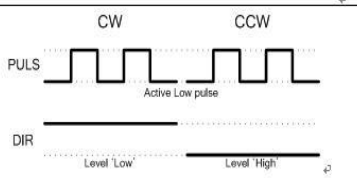
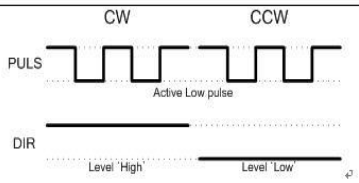
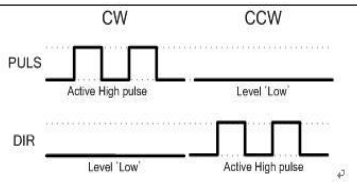
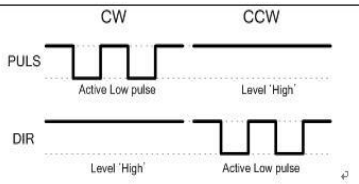
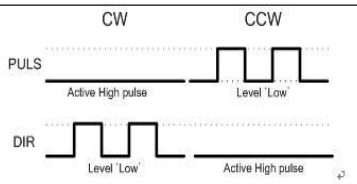
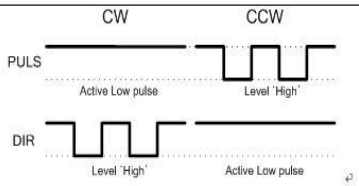
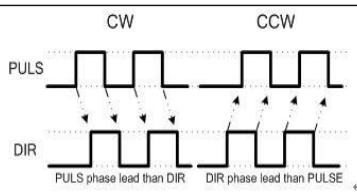
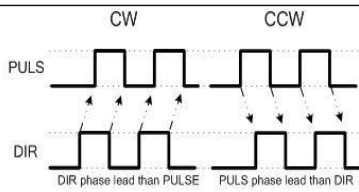
입력펄스형태

입력 방식[2:0]	입력방식[3] = '0'	입력방식[3] = '1'
"000" (One pulse input) (Two phase 1 time)		
"100" (Two pulse)		
"010" (Two phase 2 time)		
"011" (Two phase 4 times)		

● *MPG 외부와의 인터페이스*

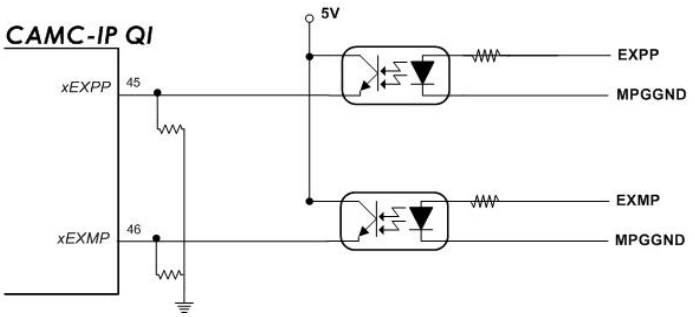
EXPP, EXMP 두 신호는 EIP-M2/4Q의 외부 로타리 인코더의 입력 신호로 이용되며, 최대 1.2MHz 고속 포토커플러를 통하여 EIP-M2/4Q칩과 연결되어 있다.

출력펄스형태

[UCFG3][2~0]	[UCFG3][3] = '0'	[UCFG3][3] = '1'
"000" (One pulse mode)		
"001" (One pulse mode)		
"010" (Two pulse mode)		
"011" (Two pulse mode)		
"100" (Two phase mode)		

● *MPG 외부와의 인터페이스*

EXPP, EXMP 두 신호는 EIP-M2/4Q의 외부 로타리 인코더의 입력 신호로 이용되며, 최대 1.2MHz 고속 포토커플러를 통하여 EIP-M2/4Q칩과 연결되어 있다.



MPG 입력 신호는 각 축에 독립적으로 입력되는 것이 아니고 두 축이 공동으로 사용하는데 0축에 입력되는 MPG신호는 2축, 1축에 입력되는 MPG신호는 3축에서 공통으로 사용한다. 즉 2/3축과 연결되는 외부 컨넥트 J2/J4의 MPG신호 입력핀은 사용되지 않는다.

● *EXPP/EXMP 신호의 확인*

EXPP / EXMP 신호의 현재 상태를 확인하기 위해서 따로 준비되어 있는 함수는 없으나, Mechanical Signal의 8번과 9번 비트가 각각 EXMP와 EXPP로 연결되어 있으므로 이를 통해 현재 상태를 확인할 수 있다. Mechanical Signal을 확인하기 위해서는 AxmStatusReadMechanical 함수를 사용하며, 이를 비트연산하여 다음과 같이 EXPP와 EXMP를 확인한다.

EIP-M2/4Q	QIMECHANICAL_EXPP_LEVEL	EIP-M2/4Q에서 EXMP 신호를 확인
	QIMECHANICAL_EXMP_LEVEL	EIP-M2/4Q에서 EXPP 신호를 확인

```
// EIP-M2.4에서 EXPP / EXMP 신호를 확인한다.
AxmStatusReadMechanical(AXIS_0, &MechanicalStatus);
if(MechanicalStatus & QIMECHANICAL_EXMP_LEVEL) { bExmp = 1; } else { bExmp = 0; }
```

```
if(MechanicalStatus & QIMECHANICAL_EXPP_LEVEL) { bExpp = 1; } else { bExpp = 0; }
printf("₩rEXPP : %x , EXMP : %x   ",bExpp,bExmp);
```

QIMECHANICAL_EXMP_LEVEL , QIMECHANICAL_EXPP_LEVEL 은
EIP-M2/4Q의 MechanicalStatus 의 비트 11, 비트 12 를 의미한다.

● *MPG 구동 함수의 사용*

MPG구동을 Set하면 자동으로 그때부터 EXPP핀과 EXMP핀에 신호가 들어오는 것에 따라 구동된다. 위치구동 모드로 사용하기 위해서는 아래와 같이 설정한다.

```
//0축을 위치구동 모드로 입력된 펄스당 10만큼씩 이동되도록 설정
AxmMPGSetEnable(0, MPG_DIFF_ONE_PHASE, 1, 10, 200);
AxmMPGSetRatio(0, 10, 4096   ); //IP에서는 지원안됨.

// 0축의 Mpg 신호를 설정을 반환한다.
long     lpInputMethod, lpDriveMode;
double dpVel, dpAccel, dpDecel, dpMPGPos;
double dMPGdenominator, dMPGnumerator;

AxmMPGGetEnable(0, &lpInputMethod, & lpDriveMode, &dpMPGPos, & dpVel);
AxmMPGGetRatio(0, &dMPGnumerator, &dMPGdenominator);
```

여기서 dMPGdenominator = 4096, MPGnumerator=1 가 의미하는 것은 MPG 한바퀴에 200펄스면 그대로 1:1로 1펄스씩 출력을 의미한다. 만약 dMPGdenominator = 4096, MPGnumerator=2 로 했을경우는 1:2로 2펄스씩 출력을 내보낸다는의미이다.
여기에 MPG PULSE = ((Numerator) * (Denominator)/ 4096) 칩내부에 출력나가는 계산식이다.

속도모드로 사용하고자 한다면 다음과 같이 한다.

```
//0축을 속도구동 모드로 MPG입력에 따라 구동하도록 설정
AxmMPGSetEnable(0, MPG_DIFF_ONE_PHASE, 0, 10, 200); //0으로 설정한다.
AxmMPGSetRatio(0, 10, 4096   );

// 0축의 MPG 신호를 설정을 반환한다.
long     lpInputMethod, lpDriveMode;
```

```
double dpVel, dpAccel, dpDecel, dpMPGPos;
double dMPGdenominator, dMPGnumerator;

AxmMPGGetEnable(0, &lpInputMethod, & lpDriveMode, &dpMPGPos, &dpVel);
AxmMPGGetRatio(0, &dMPGnumerator, &dMPGdenominator);
```

이렇게 설정하고 나면 EXPP / EXMP 신호의 입력에 따라 MPG구동을 하게 되는데, MPG구동을 해제하고자 한다면 다음과 같이 한다.

```
//0축의 MPG구동을 해제한다.
long lAxisNo = 0;
AxmMPGReset(lAxisNo);
```

```
// Ex17_AXM_EIP-M2/4QMPG.cpp : Defines the entry point for the console application.
// MPG 구동을 한다.

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

#define AXIS_0 0

void main(void)
{
   // 라이브러리를 초기화 한다.
   DWORD Code = AxlOpen();
   if (Code == AXT_RT_SUCCESS)
```

```
{
    printf("라이브러리가 초기화 되었습니다.₩n");

    //모션모듈이 있는지 검사
    DWORD dwStatus;
    Code = AxmInfoIsMotionModule(&dwStatus);
    if(Code == AXT_RT_SUCCESS)
    {
        if(dwStatus == STATUS_EXIST)
        {
            printf("모션모듈이 존재 합니다.₩n");

            // 0축의 +End limit 과 -End limit 의 Active level 을 HIGH로 설정한다.
            AxmSignalSetLimit(AXIS_0, 0, HIGH, HIGH);
            // 0축의 Inpositon 신호의 입력 레벨을 HIGH로 설정한다.
            AxmSignalSetInpos(AXIS_0, HIGH);
            // 0축의 알람 신호의 입력 레벨을 LOW로설정한다.
            AxmSignalSetServoAlarm(AXIS_0, LOW);
            // 0축의 비상정지신호(ESTOP)의 Active 입력레벨을 HIGH로 설정한다.
            AxmSignalSetStop(AXIS_0, 0, HIGH);

            //0축 지령값을 mm단위가 되도록 한다.
            AxmMotSetMoveUnitPerPulse(AXIS_0, 10, 10000);
            //0축에 초기 속도를 1로 설정한다. Default : 1
            AxmMotSetMinVel(AXIS_0, 1);
            //0축의 펄스 출력 방식을 TwoCwCcwHigh로 한다.
            AxmMotSetPulseOutMethod(AXIS_0, TwoCcwCwHigh);
            //지정 축에 Encoder 입력 방식을 4체배로 설정한다.
            AxmMotSetEncInputMethod(AXIS_0, ObverseSqr4Mode);
```

```
//상대위치 S자 속도 프로파일을 사용
AxmMotSetAbsRelMode(AXIS_0, 1);
AxmMotSetProfileMode(AXIS_0, 3);

//Servo On
AxmSignalServoOn(AXIS_0, ENABLE);

// help Message
printf("[INFORMATION]********************************** ₩n");
printf("[ESC] : Exit. ₩n");
printf("[1] : MPG_POS_DRV. ₩n");
printf("[2] : MPG_CONTI_DRV. ₩n");
printf("************************************************ ₩n");

BOOL fExit = FALSE;
DWORD MechanicalStatus;
BOOL bExmp, bExpp;
double dVelocity     = 2000;          //속도 사용 설정값
double dAccel        = 4000;          //가속도 사용 설정값
double dDecel        = 4000;          //감속도 사용 설정값
long    lInputMethod, lDriveMode;
long    lDirMode       =0;          // EIP-M2/4Q에서 사용안함
long    lUserDir       =0;          //EIP-M2/4Q에서 사용안함
double dMPGPosition = 10;          // MPG_PRESET_MODE 모드일경우 이동거리
double dMPGdenominator = 4095;
    // MPG(수동 펄스 발생 장치 입력)구동 시 나누기 값(입력 값에 +1로 계산됨)
double dMPGnumerator = 0;
   //  MPG(수동 펄스 발생 장치 입력)구동 시 곱하기 값(입력 값에 +1로 계산됨)
```

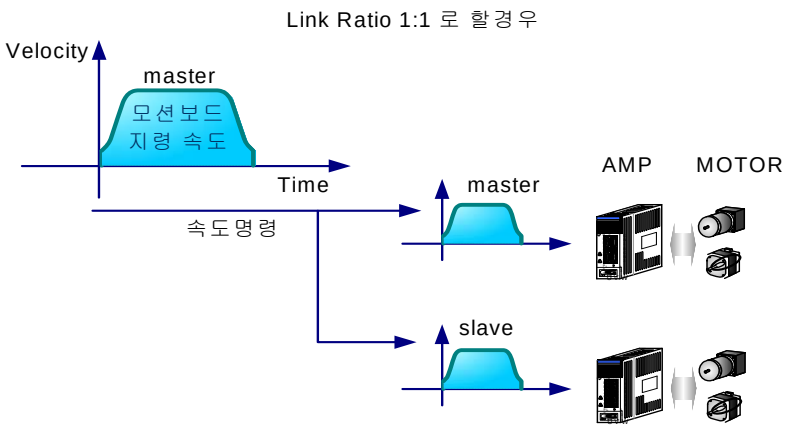
```
while (!fExit)    // 무한 루프
{
    if (kbhit())   // 아무키나 누르면
    {
        int ch = getch();
        switch (ch)
        {
        case 27:        // Esc key
                fExit = TRUE;
                AxmMPGReset(AXIS_0); // MPG구동을 종료한다.
                AxmSignalServoOn(AXIS_0, DISABLE);    //Servo Off
            break;

            case '1':
                //MPG구동을 종료한다.
                AxmMPGReset(AXIS_0);

                printf("₩nMPG_PRESET_MODE로 MPG구동을 시작한다.₩n");
                lInputMethod =0; //입력신호 방식, MPG_DIFF_ONE_PHASE(0)
                lDriveMode    = 1;
              // MPG_CONTI_DRV (0), MPG_POS_DRV (1)
              // MPG_ABS_COMMAND_POS(2), MPG_ABS_ACTUAL_POS(3),
             //MPG_ABS_COMMAND_ZERO(4). MPG_ABS_ACTUAL_ZERO(5)

                AxmMPGSetEnable(AXIS_0, lInputMethod,
                                lDriveMode, dMPGPosition, dVelocity);
                AxmMPGSetRatio(AXIS_0, dMPGnumerator,
                               dMPGdenominator );
```

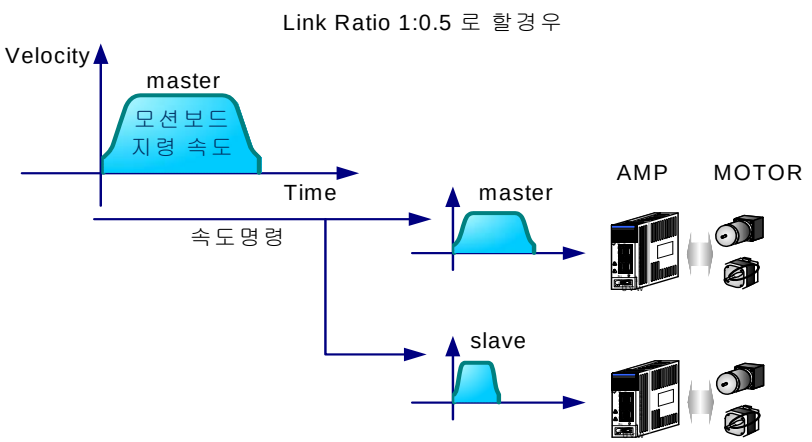
```
                    break;

                    case '2':
                        //MPG구동을 종료한다.
                        AxmMPGReset(AXIS_0);

                        printf("₩nMPG_CONT_MODE로 MPG구동을 시작한다.₩n");
                        lInputMethod = 4;    //입력 신호 방식
                                                MPG_LEVEL_ONE_PHASE(4)
                        lDriveMode = 0;   // MPG_CONTI_DRV (0), MPG_POS_DRV
(1)
                      // MPG_ABS_COMMAND_POS(2), MPG_ABS_ACTUAL_POS(3),
                    // MPG_ABS_COMMAND_ZERO(4), MPG_ABS_ACTUAL_ZERO(5)

                        AxmMPGSetEnable(AXIS_0, lInputMethod, lDriveMode,
                           dMPGPosition, dVelocity);
                        AxmMPGSetRatio(AXIS_0, dMPGnumerator,
                                            dMPGdenominator );

                    break;
                }
            }
            AxmStatusReadMechanical(AXIS_0, &MechanicalStatus);
            if(MechanicalStatus & QIMECHANICAL_EXMP_LEVEL) { bExmp = 1; }
           else { bExmp = 0; }
            if(MechanicalStatus & QIMECHANICAL_EXPP_LEVEL) { bExpp = 1; }
           else { bExpp = 0; }
            printf("₩rEXPP : %x , EXMP : %x   ",bExpp,bExmp);
```

```
                    } //End of While()
                }
                else
                    printf ("AxmInfoIsMotionModule() : ERROR ( NOT STATUS_EXIST )
                      code 0x%x₩n",Code);
            }
            else
                printf ("AxmInfoIsMotionModule() : ERROR ( Return FALSE )    code 0x%x₩n",Code);

    }
            else
                printf ("AxlOpen() : ERROR    code 0x%x₩n",Code);

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

동기구동

동기구동이란 Electric Gear Mode 라고도하는데, 독립된 두개의 축을 하나의 축처럼 제어하는 방식을 말한다. 사용자가 특정 Master축과 Slave축을 설정하고, 두 축간의 구동비를 설정해주면 Master축이 움직일 때 설정된 구동비로 Slave축이 같이 움직이는 구동이다.





이러한 동기구동을 하기 위해서는 AxmLinkSetMode 함수를 통해 설정해 주면 된다. AxmLinkSetMode 함수가 호출된 이후부터는 동기구동이 시작된다.

```
//0축을 Master, 1축을 Slave로 하고 2:1의 비율로 구동되도록 설정
long lMasterAxisNo = 0;
long lSlaveAxisNo = 1;
double dSlaveRatio = 0.5;
AxmLinkSetMode(lMasterAxisNo, lSlaveAxisNo, dSlaveRatio);

//0축을 100만큼 구동시킨다.
AxmMoveStartPos(AXIS_0, 100, 100, 200, 200);
```

이렇게 하면 0축이 100만큼 구동되는 동안 1축이 따라서 2:1의 비율로 50만큼 구동된다. 이러한 동기구동을 종료시키고자 한다면 AxmLinkResetMode함수를 사용한다.

```
//0축과 1축으로 이루어진 동기구동모드를 해제한다.
long lMasterAxisNo = 0;
AxmLinkResetMode(lMasterAxisNo);
```

```
// Ex18_AXM_ElectricGearMode.cpp : Defines the entry point for the console application.
// 동기구동을 한다.

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

```
#define AXIS_0 0
#define AXIS_1 1

void main(void)
{
   // 라이브러리를 초기화 한다.

   if (Code == AXT_RT_SUCCESS)
   {
         printf("라이브러리가 초기화 되었습니다.₩n");

         //모션모듈이 있는지 검사
         DWORD dwStatus;
         Code = AxmInfoIsMotionModule(&dwStatus);
         if(Code == AXT_RT_SUCCESS)
         {
               if(dwStatus == STATUS_EXIST)
               {
                     printf("모션모듈이 존재 합니다.₩n");

                     for(int nAxisNo = 0; nAxisNo<2; nAxisNo++)
                     {
               // 0축의 +End limit 과 -End limit 의 Active level 을 HIGH와 급정지로 설정한다.
                           AxmSignalSetLimit(nAxisNo, 0, HIGH, HIGH);
                           // 0축의 Inpositon 신호의 입력 레벨을 HIGH로 설정한다.
                           AxmSignalSetInpos(nAxisNo, HIGH);
                           // 0축의 알람 신호의 입력 레벨을 LOW로설정한다.
                           AxmSignalSetServoAlarm(nAxisNo, LOW);
```

```
        // 0축의 비상정지신호(ESTOP)의 Active 입력레벨을 HIGH로 설정한다.
        AxmSignalSetStop(nAxisNo, 0, HIGH);

        //0축 지령값을 mm단위가 되도록 한다.
        AxmMotSetMoveUnitPerPulse(nAxisNo, 10, 10000);
        //0축에 초기 속도를 1로 설정한다. Default : 1
        AxmMotSetMinVel(nAxisNo, 1);
        //0축의 펄스 출력 방식을 TwoCwCcwHigh로 한다.
        AxmMotSetPulseOutMethod(nAxisNo, TwoCcwCwHigh);
        //지정 축에 Encoder 입력 방식을 4체배로 설정한다.
        AxmMotSetEncInputMethod(nAxisNo, ObverseSqr4Mode);

        //상대위치 S자 속도 프로파일을 사용
        AxmMotSetAbsRelMode(AXIS_0, 1);
        AxmMotSetProfileMode(AXIS_0, 3);

        //Servo On
        AxmSignalServoOn(nAxisNo, ENABLE);
    }

    //동기구동을 시작한다.
    long lMasterAxisNo = AXIS_0;
    long lSlaveAxisNo = AXIS_1;
    double dSlaveRatio = 0.5;
    AxmLinkSetMode(lMasterAxisNo, lSlaveAxisNo,dSlaveRatio);

    // help Message
    printf("[INFORMATION]********************************** ₩n");
    printf("동기구동이 설정되었습니다.₩n아무키나 누르면 구동을 시작 합니다.₩n");
```

```
            printf("************************************************ ₩n");



            //아무키나 누를때까지 대기
            getch();

            //0축을 100만큼 구동시킨다.
            AxmMoveStartPos(AXIS_0, 100, 100, 200, 200);
            //구동중인지 확인한다.
            DWORD uStatus;
            AxmStatusReadInMotion(AXIS_0, &uStatus);
            while(uStatus) {
                AxmStatusReadInMotion(AXIS_0, &uStatus);
            }

            //동기구동을 종료한다.
            AxmLinkResetMode(AXIS_0);

            //Servo Off
            AxmSignalServoOn(AXIS_0, DISABLE);
            AxmSignalServoOn(AXIS_1, DISABLE);
        }
        else
            printf ("AxmInfoIsMotionModule() : ERROR ( NOT STATUS_EXIST )
              code 0x%x₩n",Code);
    }
    else
        printf ("AxmInfoIsMotionModule() : ERROR ( Return FALSE )
          code 0x%x₩n",Code);
```

```
    }
    else
        printf ("AxlOpen() : ERROR    code 0x%x₩n",Code);

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

}
```

갠트리 구동

두 축이 아래의 그림에서와 같이 기구적으로 Link되어있는 겐트리 구동시스템을 제어하기 위해 갠트리 구동 함수를 지원한다. 이 함수를 이용해 Master축을(짝수인 축 번호) 겐트리 제어로 설정하면 해당 Slave축은 Master축과 동기되어 구동된다. 만약 겐트리 설정이후 Slave축에 구동명령이나 정지 명령등을 내려도 모두 무시된다.

1) 겐트리 구동 설정 및 해제
2) 갠트리 구동에서의 홈 검색

1) 겐트리 구동 설정 및 해제

갠트리 구동을 설정하기 위해서는 AxmGantrySetEnable 함수를 사용하여 갠트리 구동을 Enable해주면 된다. 겐트리 시스템의 경우 각각의 로봇을 평행으로 놓고 기구적으로 Link하는 구조로 조립 후 Master축과 Slave축의 원점센서가 일정 값의 오차를 가지게 된다. 원점검색을 하게 되면 두축의 원점센서가 가지는 오차를 마지막에 보상해 주게 된다. 기구조립시 발생하는 센서간의 오차를 dSlOffset에 설정해 주면 된다.

```
//0축과 1축을 이용하여 갠트리 구동을 시작한다.
long lMasterAxisNo = 0;
long lSlaveAxisNo = 1;
DWORD uGantryOn;
DWORD bSlHomeUse = FALSE;   //[0] : Home 축만 원점검색 선택
//[1] : Home축 과 Slave축에 원점검색 같이 선택
double dSlOffset = 0;                //마스터와 슬레이브 홈센서들간의 기구적인 Offset값
double dSlOffsetRange = 10;          //원점 검색시 마스터 홈센서와 슬레이브 홈센서간의 오차 한계
AxmGantrySetEnable(lMasterAxisNo, lSlaveAxisNo, uSlHomeUse, dSlOffset, dSlOffsetRange);
// 사용자가 직접 설정한 값을 읽어낼수있다.
AxmGantryGetEnable (lMasterAxisNo, &SlHomeUse, &dSlOffset, &dSlORange, &uGatryOn);
```

이렇게 설정한 후에 Master축인 0축에 내리는 명령대로 0축과 1축이 동일하게 구동된다. 그리고 1축에 내리는 명령은 모두 무시된다. 홈검색을 하고자 한다면 이 상태에서 0축에 대한 홈검색 함수를 실행시키면 된다.

```
//갠트리 구동에서 원점을 검색한다.
long MasterAxisNo     =     0;
long nHmDir = 0;              //0 : (-)방향 ,  1 : (+)방향
DWORD nHmSig = 4;             //[0]PELM, [1]NELM, [2]PSLM, [3]NSLM, [4]IN0, [5]IN1, [6]IN2,
[7]IN3
BOOL bZphas = TRUE;           //Z상 검출 유무
double dHClrTime = 100.0;     //엔코더 클리어 하기 위한 대기 시간
double dHOffset  =     0.0;
AxmHomeSetMethod(MasterAxisNo, nHmDir, nHmSig, bZphas, dHClrTime, dHOffset);

double dVelF = 10000; //원점 센서가 감지되어 있지 않을때 원점검색 진행방향으로 구동할 속도
```

```
double dVelA = 1000;  //원점센서가 감지되어 있을때 원점검색 진행 반대 방향으로 구동할 속도
double dVelL = 500;         //1차 원점 검색후 재검색할 때 사용할 속도
double dvell    = 10;           //마지막 원점검출시 사용할 속도
double dAccF    =    20000; //원점센서가 감지되지 않을때 원점센서 방향으로 구동할 가감속도
double dAccA    =    10000;   //원점센서가 감지되었을때 진행 반대방향으로 구동할 가감속도
AxmHomeSetVel(MasterAxisNo, dVelF,  dVelA,  dVelL,  dvell,  dAccF,  dAccA);

AxmHomeSetStart(MasterAxisNo);
```

겐트리 구동을 종료하기 위해서는 AxmGantrySetDisable함수를 사용한다.

```
//겐트리 구동을 종료한다.
long lMasterAxisNo = 0;
long lSlaveAxisNo = 1;
AxmGantrySetDisable(lMasterAxisNo, lSlaveAxisNo);
```

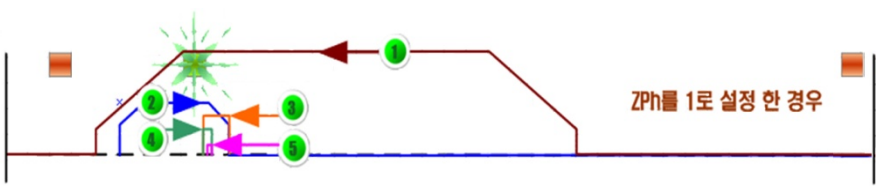
2) 갠트리 구동에서의 홈 검색

겐트리 제어에서 중요한 것은 원점검색을 수행하는데 있어 Master축과 Slave축을 고려해서 원점검색을 해야 된다는 것이다. 일반적으로 Master축의 홈센서를 먼저 찾은 후 Slave축의 홈센서을 찾는데, 제일 처음에 AxmGantrySetEnable 로 겐트리 제어로 묶어 놓은 다음에 겐트리를 홈 구동을 시작하면 구동이 되고 마스터 축에 대해서 단축 원점검색 함수를 사용하여 홈검색을 수행한다.

원점검색관련 설정변수	변수 설명
HmSig	원점센서로 사용할 신호를 설정 예) 원점센서(4), +Limit센서(0) 등
HmLev	원점센서의 Active Level을 설정 예) 0: A접점, 1: B접점

HmDir	원점센서 검출시 초기 검색진행 방향을 설정 예) 1: (+)방향, 0: (-)방향
Zphas	원점센서 검출 후 엔코더 Z상을 검출 여부 설정
VelFirst	원점검색 시 초기 고속검출 속도 (초기 원점센서가 감지되지 않았을 경우)
VelSecond	원점검색 시 1차 센서 검출 후 반대 방향으로 빠져나오는 속도
VelThird	원점검색 시 1차 센서 검출 후 재 검색 진행하는 속도
VelLast	원점검색 시 최종 검출 속도 설정 [원점검색의 정밀도 결정]
AccFirst	원점검색 시 초기 고속검출 가속도
AccSecond	원점검색 시 1차 센서 검출 후 반대 방향으로 빠져나오는 가속도
HClrTim	원점검색 완료 후 Command, Actual(Encoder)위치를 Clear전 대기시간
HOffset	원점검색 완료 후 기구원점 재 설정 시 이동 Offset 값
NSWLimit	원점검색 완료 후 적용할 (-) Software Limit 값 설정
PSWLimit	원점검색 완료 후 적용할 (+) Software Limit 값 설정 (NSWLimit값이 PSWLimit값보다 같거나 클 경우 S/W Limit 설정되지 않음)

● *Step별 원점검출 속도 설정 및 홈검색 시퀀스의 이해*
(Master, Slave 똑같은 속도사용)



1.

VelFirst, AccFirst사용

HmDir 방향으로 원점센서를 고속 검출하고 HAccF 감속도로 감속정지

2.

VelSecond, AccSecond사용

HmDir 반대 방향으로 원점센서의 Down Edge를 검출하고 HAccF감속도로 감속정지

3.

VelThird사용

HmDir 방향으로 원점센서의 Up Edge를 검출하고 급정지

4.

VelThird 사용

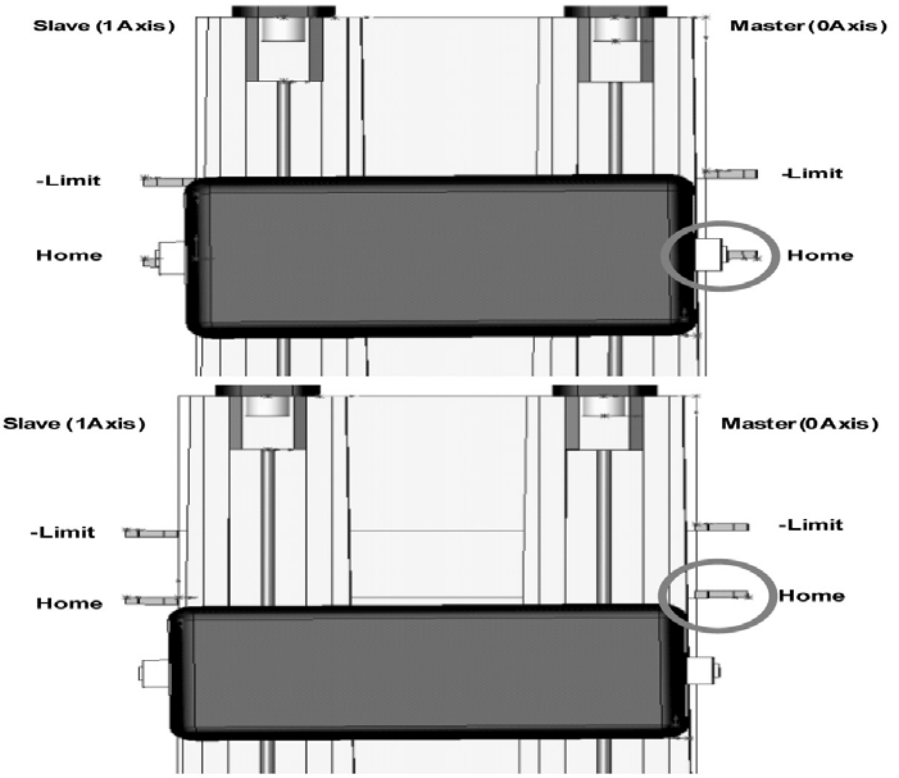
HmDir 반대 방향으로 Z상의 Down Edge를 검출하고 급정지

5.

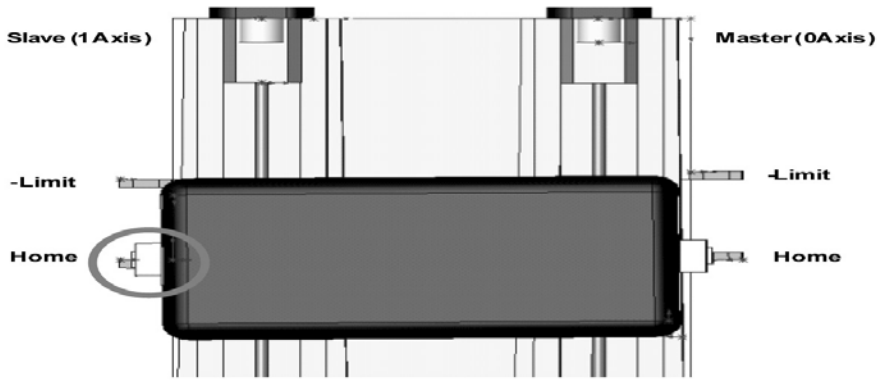
VelLast 사용

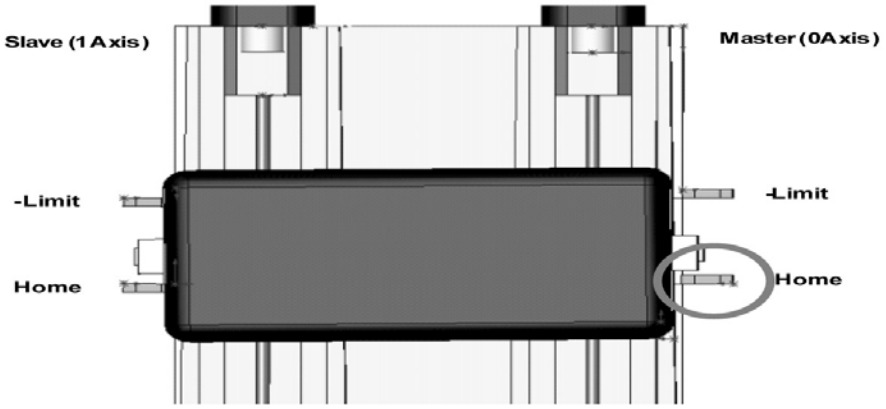
HmDir 방향으로 Z상의 Up Edge를 검출하고 급정지

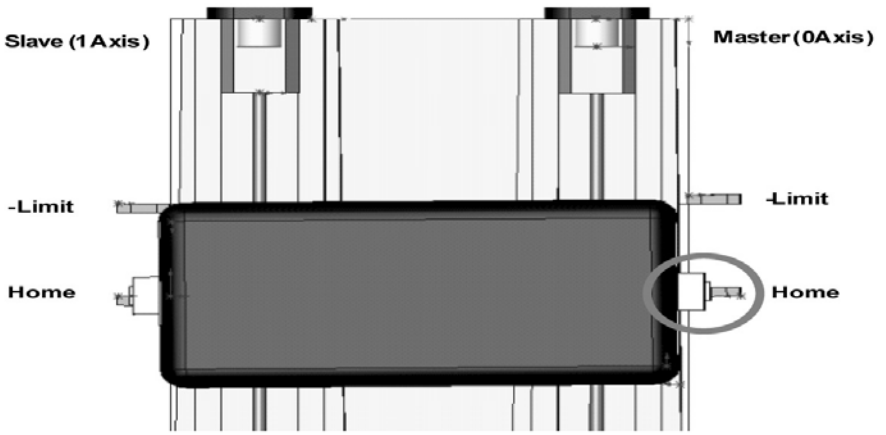
1. Master축의 홈센서를 먼저 찾는다. Maste축만 위치 0으로 만든다.



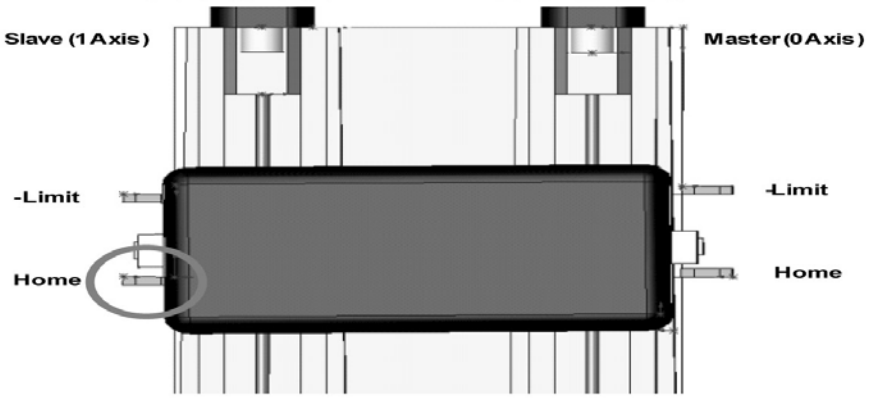
2. Slave 축의 홈센서를 찾는다. 슬레이브축만 위치 0 로 만든다. 그럼 마스터에 대한 슬레이브의 Offset 값을 얻을수있다.

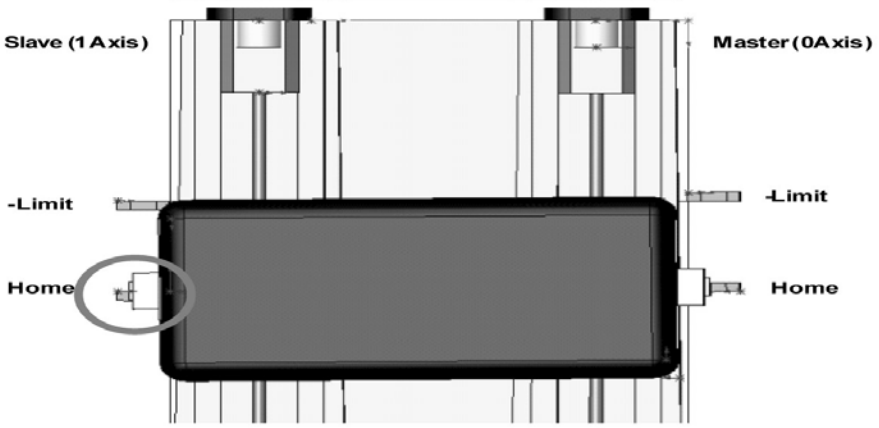




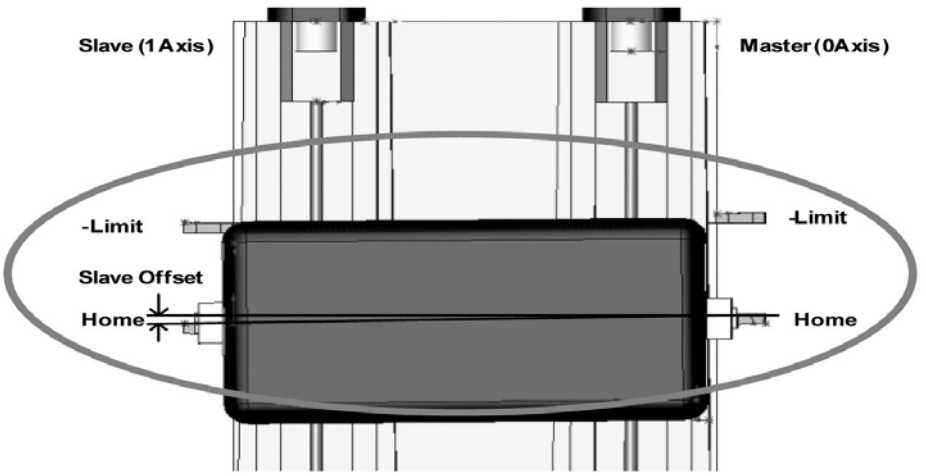


Slave (1Axis)
Master(0Axis)
-Limit
-Limit
Home
Home

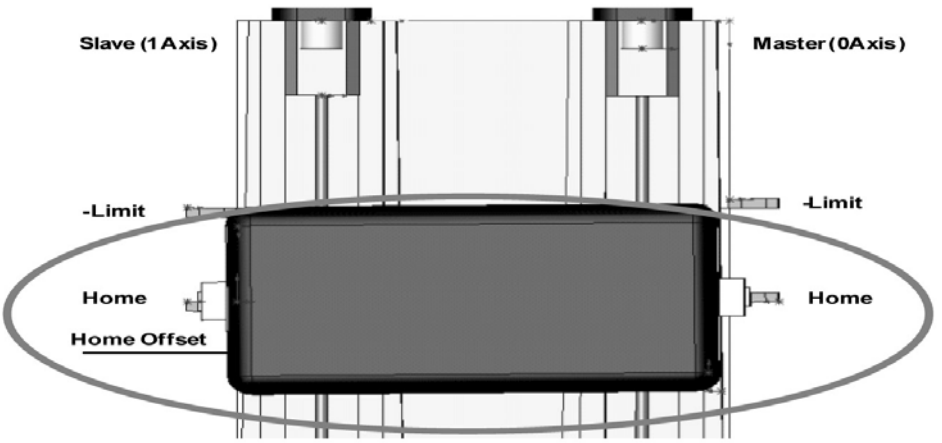

Slave (1Axis)
Master(0Axis)
-Limit
-Limit
Home
Home


Slave (1Axis)
Master(0Axis)
-Limit
-Limit
Home
Home

3. Master와 Slave의 Link를 풀고 Slave축을 Slave offset만큼 이동해준다. 그다음 Master축을 0으로 보낸다. 다음 다시 Link시킨다.



4. 홈 Offset 만큼 이동해준 다음 위치를 0로 만든다.
홈 Offset이 없을경우 이동하지 않고 위치를 0로 만든다.



5. Software Limit 설정 및 겐트리 구동하기

※ 주의 : 원점검색이 정상적으로 완료된 후 Soft Limit을 설정할 수 있다. Soft Limit은 기구적으로 Limit센서가 장착되어 있지 않거나 상황에 따라서 Limit을 변경해야 될 경우 사용하면 된다. 이때 한 가지 주의할 것은 NLimit(-)보다 PLimit(+)값이 커야 정상적으로 Soft Limit이 설정된다. 만약 NSwLimit값이 PSwlimit 값보다 같거나 클 경우 Soft Limit는 설정되지 않는다.

```
// Ex19_AXM_GentryDrive.cpp : Defines the entry point for the console application.
// 겐트리구동을 한다.

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

#define AXIS_0 0
#define AXIS_1 1

void main(void)
{
   // 라이브러리를 초기화 한다.
```

```
if (Code == AXT_RT_SUCCESS)
{
    printf("라이브러리가 초기화 되었습니다.₩n");

    //모션모듈이 있는지 검사
    DWORD dwStatus;
    Code = AxmInfoIsMotionModule(&dwStatus);
    if(Code == AXT_RT_SUCCESS)
    {
        if(dwStatus == STATUS_EXIST)
        {
            printf("모션모듈이 존재 합니다.₩n");

            for(int nAxisNo = 0; nAxisNo<2; nAxisNo++)
            {
        // 0축의 +End limit 과 -End limit 의 Active level 을 HIGH와 급정지로 설정한다.
                AxmSignalSetLimit(nAxisNo, 0, HIGH, HIGH);
                // 0축의 Inpositon 신호의 입력 레벨을 HIGH로 설정한다.
                AxmSignalSetInpos(nAxisNo, HIGH);
                // 0축의 알람 신호의 입력 레벨을 LOW로설정한다.
                AxmSignalSetServoAlarm(nAxisNo, LOW);
                // 0축의 비상정지신호(ESTOP)의 Active 입력레벨을 HIGH로 설정한다.
                AxmSignalSetStop(nAxisNo, 0, HIGH);

                //0축 지령값을 mm단위가 되도록 한다.
                AxmMotSetMoveUnitPerPulse(nAxisNo, 10, 10000);
                //0축에 초기 속도를 1로 설정한다. Default : 1
                AxmMotSetMinVel(nAxisNo, 1);
                //0축의 펄스 출력 방식을 TwoCwCcwHigh로 한다.
```

```
        AxmMotSetPulseOutMethod(nAxisNo, TwoCcwCwHigh);
        //지정 축에 Encoder 입력 방식을 4체배로 설정한다.
        AxmMotSetEncInputMethod(nAxisNo, ObverseSqr4Mode);

        // 원점검색 방법을 설정
        AxmHomeSetMethod(nAxisNo,-1,HomeSensor,1,2000.0,0.0);
        // 원점검색시 사용될 속도를 설정
        AxmHomeSetVel(nAxisNo, 100, 20, 10, 10, 200, 40);

        //Servo On
        AxmSignalServoOn(nAxisNo, ENABLE);
    }

    //갠트리 구동을 설정한다.
    long lMasterAxisNo = AXIS_0;
    long lSlaveAxisNo = AXIS_1;
    BOOL bSlHomeUse = FALSE;
    //TRUE : 마스터 홈센서를 찾은 후 슬레이브 홈센서도 찾음
    double dSlOffset = 0;              //마스터와 슬레이브 홈센서들간의 Offset
    double dSlOffsetRange = 10;
    //원점 검색시 마스터 홈센서와 슬레이브 홈센서간의 오차 한계
    AxmGantrySetEnable(lMasterAxisNo, lSlaveAxisNo, bSlHomeUse,
      dSlOffset, dSlOffsetRange);

    // help Message
    printf("[INFORMATION]********************************** ₩n");
    printf("갠트리구동이 설정되었습니다.₩n아무키나 누르면 구동을
     시작 합니다.₩n");
    printf("************************************************* ₩n");
```

```
//아무키나 누를때까지 대기
getch();

printf("₩n갠트리 구동의 원점을 검색한다.₩n");
// 원점검색 방법을 설정
AxmHomeSetMethod(lMasterAxisNo,-1,HomeSensor,1,2000.0,0.0);
// 원점검색시 사용될 속도를 설정
AxmHomeSetVel(lMasterAxisNo, 100, 20, 10, 10, 200, 40);

//원점을 검색한다.
AxmHomeSetStart(lMasterAxisNo);

//현재 원점검색 진행률을 확인
DWORD uHomeMainStepNumber , uHomeStepNumber_0;
AxmHomeGetRate(lMasterAxisNo, &uHomeMainStepNumber ,
  &uHomeStepNumber_0);
while(uHomeStepNumber_0 < 100)
{
      AxmHomeGetRate(lMasterAxisNo, &uHomeMainStepNumber ,
              &uHomeStepNumber_0);
      printf("₩r원점검색중... %d ", uHomeStepNumber_0);
}
printf("₩r원점검색 완료 ₩n");

printf("Master 축을 100만큼 구동시킨다.₩n");
AxmMoveStartPos(lMasterAxisNo, 100, 100, 200, 200);
//구동중인지 확인한다.
```

```
                DWORD uStatus;
                AxmStatusReadInMotion(lMasterAxisNo, &uStatus);
                while(uStatus) {
                    AxmStatusReadInMotion(lMasterAxisNo, &uStatus);
                }

                //갠트리구동을 종료한다.
                AxmGantrySetDisable(lMasterAxisNo,lSlaveAxisNo);

                //Servo Off
                AxmSignalServoOn(nAxisNo, DISABLE);
                AxmSignalServoOn(nAxisNo, DISABLE);

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

    // 라이브러리를 종료 한다.
    if (AxlClose())
        printf("라이브러리가 종료 되었습니다.₩n");
    else
```

```
        printf("라이브러리가 정상적으로 종료되지 않았습니다.₩n");

}
```

가상축 매핑

가상축 매핑이란 라이브러리에서 구동명령을 내릴 때 실제 해당 축번호가 아닌 임의의 축번호를 사용하여 명령을 내릴 수 있도록 하기 위한 함수이다. 예를 들어서 0번 축에 명령을 내리는데, 명령시에는 2번 축이라고 명령을 내리는 것을 말한다. 만약, 시스템이 원래 0번부터 10번까지 축이 있었는데, 특별한 이유로 가운데 3,4번 축이 제거되었다고 하면 이럴경우 축번호는 순서대로 0번부터 8번까지 당겨져서 작동되게 된다. 그러므로 프로그램상의 축번호를 일일이 변경해야 하는 상황이 발생하는데, 이러한 경우에 프로그램 소스를 건드리지 않고, 원래의 축번호로 매핑해 줌으로서 해결할 수 있게 된다. 이 경우는 3~ 8번 축을 5~10번축으로 축번호를 변경해 주면 된다.

이러한 가상 축 매핑을 사용하고자 한다면 다음과 같이 한축을 가상축으로 매핑하는 AxmVirtualSetAxisNoMap 함수를 사용하거나 여러축을 가상축으로 매핑하는 AxmVirtualSetMultiAxisNoMap 함수를 사용한다. 이때 유의할 것은 가상축으로 매핑되면 이전의 실제 축번호는 존재하지 않는것으로 되므로 이전 축번호로 내린 명령은 정상작동 하지 않으며, 설정후부터는 새로 설정한 축번호로 사용하여야 한다.

```
//0축을 2번 축으로 매핑한다.
long lRealAxisNo = 0;
long lVirtualAxisNo = 2;
AxmVirtualSetAxisNoMap(lRealAxisNo, lVirtualAxisNo);
```

```
//0축과 1축을 2번과 3번으로 매핑한다.
long lSize = 2;
long lpRealAxesNo[2] = {0,1};
long lpVirtualAxesNo[2] = {2,3};
AxmVirtualSetMultiAxisNoMap(lSize, lpRealAxesNo, lpVirtualAxesNo);
```

이렇게 매핑된 축번호를 다시 원래대로 하기 위해서는 AxmVirtualResetAxisMap 함수를 사용한다.

```
//가상축 매핑을 해제한다.
AxmVirtualResetAxisMap();
```

트리거 신호의 사용

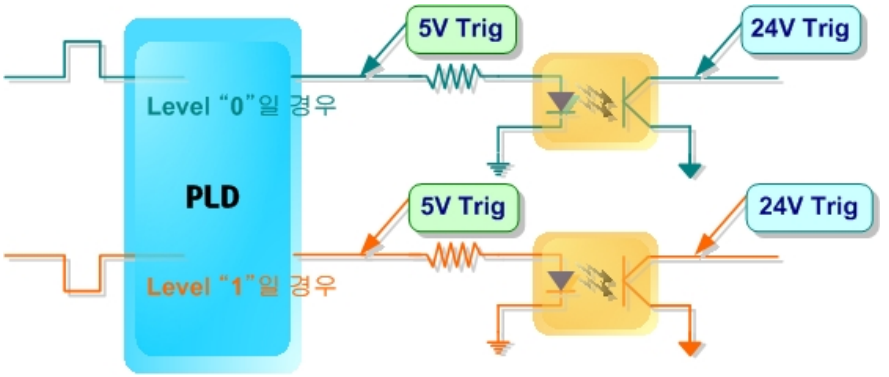
AXM에서는 모션구동중 특정한 상황에 따라 인터럽트 뿐만 아니라 5V레벨의 트리서 신호를 내보낼 수도 있다. 트리거 신호는 TRIG 신호핀에서 나오며, ActiveHIGH로 설정해두면 평소에는 0V이다가 트리거 신호가 발생하면 설정해준 시간동안 5V였다가 다시 0V가 된다. ActiveLow일 때는 그와 반대이다. 물론 기구부에서는 24V가 된다. 트리거 신호를 발생시키기 위해서는 먼저 AxmTriggerSetTimeLevel 함수로 트리거의 신호 레벨과 지속시간등을 설정하고, 특정조건에서 트리거를 발생시키는 함수들을 호출하면 된다. 트리거 발생 함수는 특정위치에서 트리거를 발생하는 AxmTriggerSetAbsPeriod / AxmTriggerOnlyAbs 함수와, 특정구간동안 일정 간격으로 트리거를 발생시키는 AxmTriggerSetBlock 함수, 그리고 임의의 시점에 한번의 트리거를 발생시키는 AxmTriggerOneShot 함수가 있다.

- ***AxmTriggerSetTimeLevel***

트리거 신호를 발생시킬 때 트리거 출력 신호 시간 설정, 트리거 출력 신호의 사용 여부 및 Active Level, 트리거 기준 위치값 입력 소스, 트리거 출력 시 인터럽트 출력 여부들을 설정하는 함수이다. 트리거 출력 신호 시간은 최소(1 = 1usec) ~ 최대(40000 = 4msec)까지 설정 가능하며, 트리거 출력 신호의 사용 여부 및 Active Level은 LOW(0), HIGH(1), UNUSED(2), USED(3)로 설정하면 된다. 트리거 기준 위치값 입력 소스는 Command Position(0)과 Actual Position(1)을 사용 할 수 있으며, 트리거 출력 시 인터럽트 출력 여부를 DISABLE(0) 또는 ENABLE(1)로 설정할 수 있다.

```
//0축에 트리거 신호의 레벨과 지속시간을 설정한다.
long lAxisNo = 0;
double dTrigTime = 100;    // 0.1ms
DWORD uTriggerLevel = HIGH;
DWORD uSelect = 0;         // Command Position을 기준으로 트리거 발생
DWORD uInterrupt = DISABLE;

AxmTriggerSetTimeLevel(lAxisNo, dTrigTime, uTriggerLevel, uSelect, uInterrupt);
```

Low(B접점) : 평상시 Close -> 감지되면 Open
High(A접점): 평상시 Open -> 감지되면 Close
Falling Edge : 평상시 High 상태를 유지하다가 트리거 시점에서 Low로 떨어졌다가 다시 High로 올라간다.
Riging Edge : 평상시 Low 상태를 유지하다가 트리거 시점에서 High로 올라갔다가 다시 Low로 떨어진다.



※ 알림 : 5V트리거 출력의 경우 Active Level을 0으로 설정했다면 평상시0V에서 트리거 발생시 5V가 되고 24V트리거의 경우 Open상태에서 트리거가 출력되면 N24V가 된다. 따라서 단자대를 통해 24V로 트리거를 사용할 경우 트리거 출력을 확인하기 위해서는 2kΩ정도의 풀업 저항을 달아 측정해야 된다.

● *AxmTriggerSetAbsPeriod*

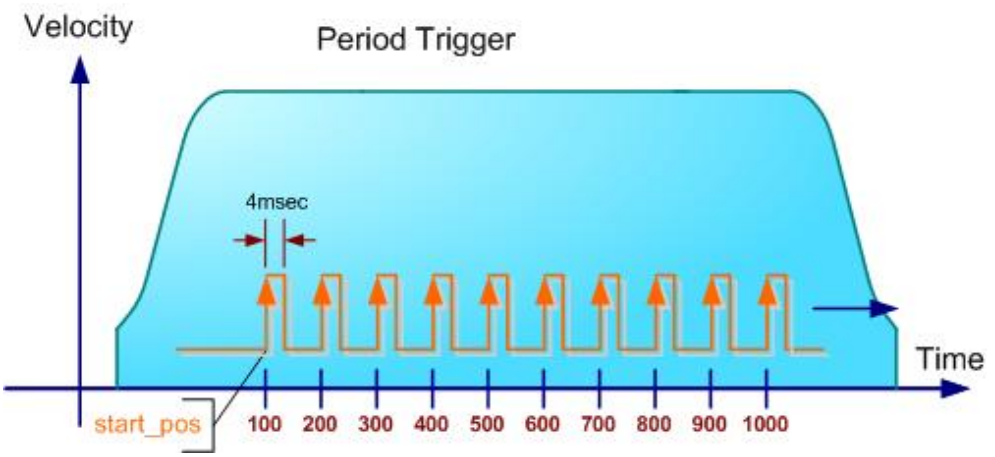
특정 위치에서 트리거를 발생시키기 위해서 AxmTriggerSetAbsPeriod 함수를 사용할 수 있다. 이 함수는 트리거 출력 방식과 위치를 입력받아서 트리거를 발생 시키는데, uMethod값을 0으로 하면 입력된 트리거 위치값을 사용한 주기 트리거 방식으로 트리거 위치값을 1000으로 설정하고 이동할경우 매 1000위치(1000,2000,3000,…)마다 트리거가 발생되고, uMethod값을 1로 하면 절대 위치 트리거 발생 방식으로 1000의 위치에서만 트리거가 발생하고 2000,3000등에서는 발생하지 않는다.

```
//0축을 이동하는 동안 매 1000위치 마다 트리거가 발생하도록 한다.
long lAxisNo = 0;
DWORD uMethod = 0;  //[0] : 주기 트리거 방식,  [1] : 절대위치 트리거 방식
```

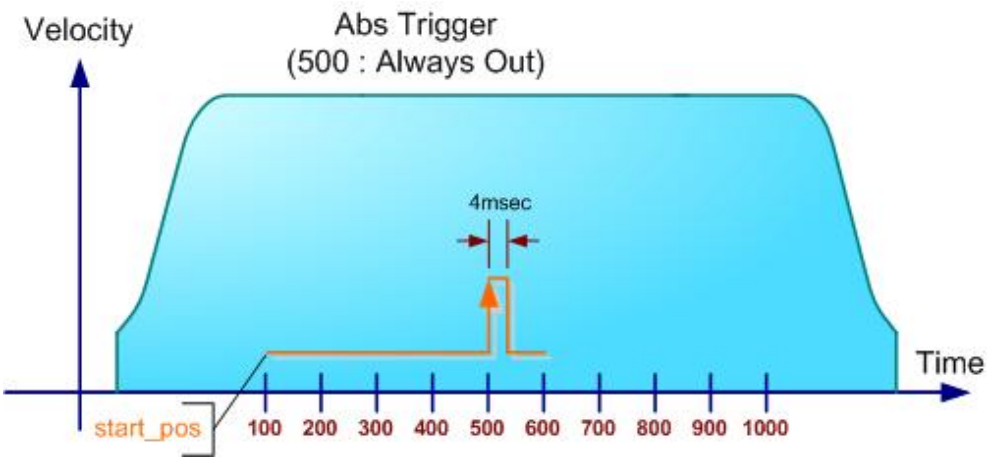
```
double dPosition = 1000;
AxmTriggerSetAbsPeriod(lAxisNo, uMethod, dPosition);
```

주의점 : 트리거 모드를 주기모드로 설정시 설정할 때 트리거가 출력된다 AxmTriggerSetBlock함수와 다르게 시작한 위치부터 카운터 변화량을 기준으로 트리거 출력된다.

주기모드 : Position이 100로 했을경우 100 펄스 마다 트리거 출력이 나온다.



절대모드 : Position이 500로 했을경우 500위치에오면 무조건 트리거가 출력된다.



● *AxmTriggerOnlyAbs*

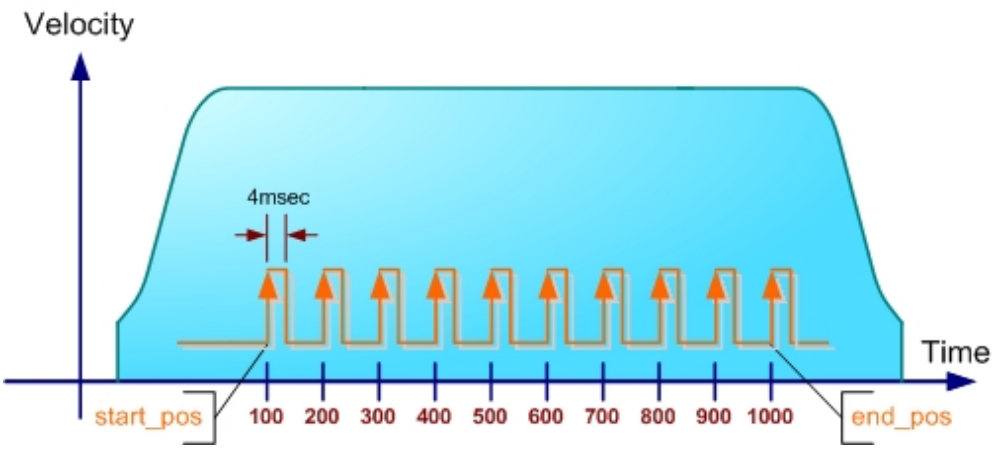
만약 트리거를 발생시키는 위치가 불규칙하면서 많이 있을경우, 앞에서 언급한 함수를 사용하면 트리거 발생 위치만큼 함수를 중복해서 호출해야 하지만, AxmTriggerOnlyAbs 함수를 사용하면 이러한 불편을 줄일 수 있다. AxmTriggerOnlyAbs 함수는 트리거를 발생 시키고자 하는 위치를 배열로 받아서 해당 위치에 트리거를 발생시켜 주는 함수 이다.

```
//0축을 이동하는 동안 지정된 10개 위치들에서 트리거를 발생시킨다.
long lAxisNo = 0;
long lTrigNum = 10;
double dTrigPos = {100, 150, 200, 250, 400, 520, 650, 700, 800, 980};
AxmTriggerOnlyAbs(lAxisNo, lTrigNum, dTrigPos);
```

● *AxmTriggerSetBlock*

트리거가 발생할 시작점과 끝점을 설정하고 그 구간내에서 일정 거리마다 한번씩 트리거를 발생 시키는 함수이다.

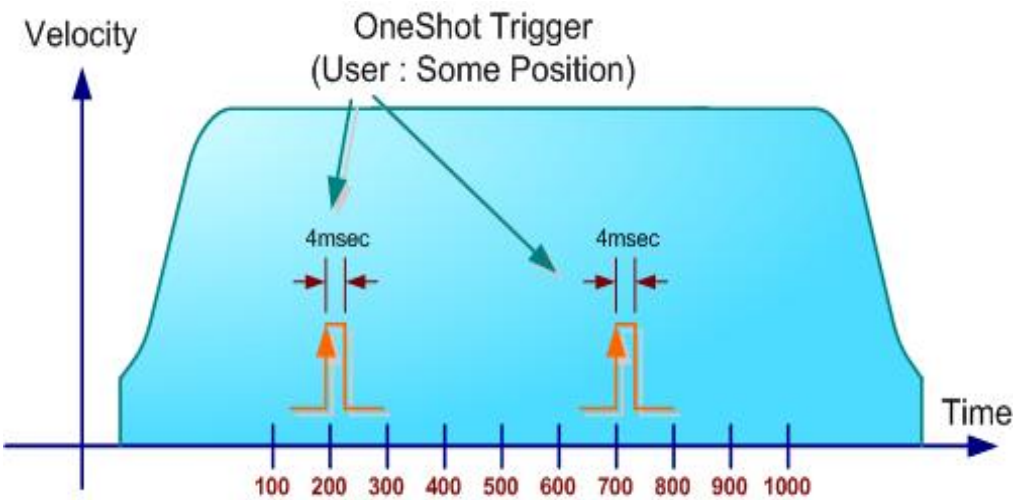
```
//0축에서 300위치부터 1000위치가 될때까지 100만큼 이동할 때마다 트리거 발생시킨다
long lAxisNo = 0;
double dStartPosition = 100;
double dEndPosition = 1000;
double dPeriodPosition = 100;
AxmTriggerSetBlock(lAxisNo, dStartPosition, dEndPosition, dPeriodPosition);
```



● *AxmTriggerOneShot*

사용자가 임의의 시점에 한 개의 트리거 신호를 출력하는 함수이다. AxmTriggerOneShot 함수는 호출하는 즉시 트리거를 발생시키는 함수이다.

```
//현시점에서 트리거를 발생시킨다.
long lAxisNo = 0;
AxmTriggerOneShot(lAxisNo);
```




```
// Ex22_AXM_Trigger.cpp : Defines the entry point for the console application.
// 구동하면서 트리거 신호를 발생시킨다.

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

#define AXIS_0 0

void main(void)
{
    // 라이브러리를 초기화 한다.
    DWORD Code = AxlOpen();
    if (Code == AXT_RT_SUCCESS)
```

```
{
    printf("라이브러리가 초기화 되었습니다.₩n");

    //모션모듈이 있는지 검사
    DWORD dwStatus;
    Code = AxmInfoIsMotionModule(&dwStatus);
    if(Code == AXT_RT_SUCCESS)
    {
        if(dwStatus == STATUS_EXIST)
        {
            printf("모션모듈이 존재 합니다.₩n");

            // 0축의 +End limit 과 -End limit 의 Active level 을 HIGH로 설정한다.
            AxmSignalSetLimit(AXIS_0, 0, HIGH, HIGH);
            // 0축의 Inpositon 신호의 입력 레벨을 HIGH로 설정한다.
            AxmSignalSetInpos(AXIS_0, HIGH);
            // 0축의 알람 신호의 입력 레벨을 LOW로설정한다.
            AxmSignalSetServoAlarm(AXIS_0, LOW);
            // 0축의 비상정지신호(ESTOP)의 Active 입력레벨을 HIGH로 설정한다.
            AxmSignalSetStop(AXIS_0, 0, HIGH);

            //0축 지령값을 mm단위가 되도록 한다.
            AxmMotSetMoveUnitPerPulse(AXIS_0, 10, 10000);
            //0축에 초기 속도를 1로 설정한다. Default : 1
            AxmMotSetMinVel(AXIS_0, 1);
            //0축의 펄스 출력 방식을 TwoCwCcwHigh로 한다.
            AxmMotSetPulseOutMethod(AXIS_0, TwoCcwCwHigh);
            //지정 축에 Encoder 입력 방식을 4체배로 설정한다.
            AxmMotSetEncInputMethod(AXIS_0, ObverseSqr4Mode);
```

```
//절대위치 구동으로 설정한다.
AxmMotSetAbsRelMode(AXIS_0, 0);
//대칭 S자 속도파일로 설정한다.
AxmMotSetProfileMode(AXIS_0, 4);

// 원점검색 방법을 설정
AxmHomeSetMethod(AXIS_0,-1,HomeSensor,1,2000.0,0.0);
// 원점검색시 사용될 속도를 설정
AxmHomeSetVel(AXIS_0, 100, 20, 10, 10, 200, 40);

//Servo On
AxmSignalServoOn(AXIS_0, ENABLE);

//원점을 검색한다.
AxmHomeSetStart(AXIS_0);

//현재 원점검색 진행률을 확인
DWORD uHomeMainStepNumber , uHomeStepNumber_0;
AxmHomeGetRate(AXIS_0, &uHomeMainStepNumber ,
                        &uHomeStepNumber_0);
while(uHomeStepNumber_0 < 100)
{
    AxmHomeGetRate(AXIS_0, &uHomeMainStepNumber ,
                        &uHomeStepNumber_0);
    printf("₩r원점검색중 : %d% ", uHomeStepNumber_0);
}

// help Message
```

```
    printf("₩n[INFORMATION]******************************************************** ₩n");

                printf("[ESC] : Exit ₩n");
                printf("[1] : (+)방향으로 200까지 가면서 50마다 트리거를 발생 ₩n");
                printf("[2] : (+)방향으로 200만큼 가면서 지정된 10개 위치들에서 트리거를 발생
₩n");
                printf("[3] : (+)방향으로 200만큼 가면서 50위치부터 150위치까지 10간격으로
트리거 발생 ₩n");

    printf("****************************************************************************** ₩n");

                //0축에 트리거 신호의 레벨과 지속시간을 설정한다.
                double dTrigTime = 4000;    // 4ms
                DWORD uTriggerLevel = HIGH;
                DWORD uSelect = 0;        // Command Position을 기준으로 트리거 발생
                DWORD uInterrupt = DISABLE;
                AxmTriggerSetTimeLevel (AXIS_0, dTrigTime, uTriggerLevel, uSelect,
uInterrupt);

                DWORD uStatus;
                BOOL fExit = FALSE;
                while (!fExit)                    // 무한 루프
                {
                    if (kbhit())              // 아무키나 누르면
                    {
                        int ch = getch();
                        DWORD uMethod;
                        double dPosition;
```

```
long lTrigNum = 10;
double dTrigPos[10];
double dStartPosition, dEndPosition, dPeriodPosition;


switch (ch)
{
    case 27:        // Esc key
        fExit = TRUE;
        AxmSignalServoOn(AXIS_0, DISABLE);
    break;

    case '1':  //이동하는 동안 매 50위치 마다 트리거가 발생
        uMethod = 0;  //[0] : 주기 트리거 방식,
                      //[1] : 절대위치 트리거 방식
        dPosition = 50;
        AxmTriggerSetAbsPeriod(AXIS_0, uMethod, dPosition);

        //200위치까지 구동시킨다.
        AxmMoveStartPos(AXIS_0, 200, 50, 100, 100);
        AxmStatusReadInMotion(AXIS_0, &uStatus);
        while(uStatus) {
            AxmStatusReadInMotion(AXIS_0, &uStatus);
        }

        //다시 원점으로 복귀한다.
        AxmMoveStartPos(AXIS_0, 0, 100, 200, 200);
        AxmStatusReadInMotion(AXIS_0, &uStatus);
        while(uStatus) {
```

```
                AxmStatusReadInMotion(AXIS_0, &uStatus);
            }
        break;

        case '2'://이동하는 동안 지정된 10개 위치들에서 트리거를 발생
            lTrigNum = 10;
            dTrigPos[0] = 10;   dTrigPos[1] = 15;
            dTrigPos[2] = 20;   dTrigPos[3] = 50;
            dTrigPos[4] = 85;   dTrigPos[5] = 100;
            dTrigPos[6] = 120; dTrigPos[7] = 150;
            dTrigPos[8] = 172; dTrigPos[9] = 190;
            AxmTriggerOnlyAbs(AXIS_0,lTrigNum, dTrigPos);

            //200위치까지 구동시킨다.
            AxmMoveStartPos(AXIS_0, 200, 50, 100, 100);
            AxmStatusReadInMotion(AXIS_0, &uStatus);
            while(uStatus) {
                AxmStatusReadInMotion(AXIS_0, &uStatus);
            }

            //다시 원점으로 복귀한다.
            AxmMoveStartPos(AXIS_0, 0, 100, 200, 200);
            AxmStatusReadInMotion(AXIS_0, &uStatus);
            while(uStatus) {
                AxmStatusReadInMotion(AXIS_0, &uStatus);
            }
        break;

        case '3':  //0축에서 50위치부터 150위치가 될때까지 10만큼
```

```
                                        //이동할 때마다 트리거 발생
                                dStartPosition = 50;
                                dEndPosition = 150;
                                dPeriodPosition = 10;
                                AxmTriggerSetBlock(AXIS_0, dStartPosition, dEndPosition,
                                                dPeriodPosition);

                                //200위치까지 구동시킨다.
                                AxmMoveStartPos(AXIS_0, 200, 50, 100, 100);
                                AxmStatusReadInMotion(AXIS_0, &uStatus);
                                while(uStatus) {
                                    AxmStatusReadInMotion(AXIS_0, &uStatus);
                                }

                                //다시 원점으로 복귀한다.
                                AxmMoveStartPos(AXIS_0, 0, 100, 200, 200);
                                AxmStatusReadInMotion(AXIS_0, &uStatus);
                                while(uStatus) {
                                    AxmStatusReadInMotion(AXIS_0, &uStatus);
                                }
                            break;
                        }
                    }
                } //End of While()
            }
            else
                printf ("AxmInfoIsMotionModule() : ERROR ( NOT STATUS_EXIST )   code
0x%x₩n",Code);
        }
```

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


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

네트워크 상태에 대한 모션 구동 에러 처리

모션 구동 중 네트워크가 끊어 졌을 때 모션 구동 상태에 대한 예외처리 방법을 설정하고 확인한다. dwNetErrorAct 인자로 0으로 설정 할 경우 현재 모션이 계속 진행이 되고, 1로 설정을 할 경우 감속 정지, 2로 설정 할 경우 급정지가 된다

이를 AxmNetWorkErrorSetAction 함수에서 설정 할 수 있으면 설정되어 진 상태는 AxmNetWorkErrorGetAction 함수에서 확인 할 수 있다.

```
// 0 축이 연결된 노드의 통신이 끊어 졌을때 급 정지 되는 설정
long lAxisNo = 0;
DWORD dwNetErrorAct= 2;
AxmNetWorkErrorSetAction  (lAxisNo, dwNetErrorAct);
```

```
// 0축이 연결된 노드의 통신이 끊어 졌을때  급 정지 되는 확인
DWORD dwpNetErrorAct;
AxmNetWorkErrorGetAction (long lAxisNo, &dwpNetErrorAct);
```

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

#define AXIS_0 0


void main(void)
{
   // 라이브러리를 초기화 한다.
   DWORD Code = AxlOpen();
   if (Code == AXT_RT_SUCCESS)
   {
         printf("라이브러리가 초기화 되었습니다.₩n");

         //모션모듈이 있는지 검사
         DWORD dwStatus;
         DWORD dwpNetErrorAct;

         Code = AxmInfoIsMotionModule(&dwStatus);
         if(Code == AXT_RT_SUCCESS)
         {
               if(dwStatus == STATUS_EXIST)
```

```
            {
                printf("모션모듈이 존재 합니다.₩n");

                // 0 축이 연결된 노드의 통신이 끊어 졌을때 감속 정지 되는 설정
                AxmNetWorkErrorSetAction  (AXIS_0, 1);
                AxmNetWorkErrorGetAction (AXIS_0, & dwpNetErrorAct);
            }
            else
                printf ("AxmInfoIsMotionModule() : ERROR ( NOT STATUS_EXIST )
                        code 0x%x₩n",Code);
        }
        else
            printf ("AxmInfoIsMotionModule() : ERROR ( Return FALSE )
                    code 0x%x₩n",Code);
    }
        else
            printf ("AxlOpen() : ERROR    code 0x%x₩n",Code);


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

}
```

Motion Command 매뉴얼 정보

Motion Cmd 매뉴얼 정보 본 매뉴얼은 EIP-M2/4Q을 포함한 상위 모션 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.

헤더 파일

본 매뉴얼의 함수 이름

헤더 파일

Visual C++
AXM.h

Visual Basic
AXM.bas

Delphi
AXM.pas

본 매뉴얼의 함수 이름

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

※ 라이브러리 함수 Prefix

Axm_ : 모션모듈 및 라이브러리 전용 함수임을 나타낸다. Axm으로 시작되는 함수들은 모두 AXM.h 에 정의 되어 있다.
Set_ : 칩의 레지스터나 변수들을 설정한다.
Get_ : Set과 한 쌍으로 Set함수에 의해 설정된 변수 값을 확인하거나 칩 레지스터의 상태를 읽는다.
Is_ : 모션 구동 상태, Servo-On 신호 등의 상태를 읽어본다.
_Move : 구동 시작 명령으로 이 명령을 포함한 함수가 실행되면 모션모듈에서 즉시 펄스가 출력되지

만 펄스 출력이 종료되는 시점에서 실행함수를 벗어나게 된다.
Start_Move : move 명령과 같으나 함수가 시작되고 모션모듈에서 펄스가 출력되는 시점에서 함수를 벗어나게 된다.

Vel_ : 속도 모드로 연속 구동하거나 속도 오버라이드를 한다.
Accel_ : 구동 중 가속 구간의 가속도(가속도의 단위는 Unit/pulse를 1/1로 한경우에 PPS[Pulses/sec]) 또는 저크값을 설정한다.
Decel_ : 구동 중 감속 구간의 가속도(가속도의 단위는 Unit/pulse를 1/1로 한경우에 PPS[Pulses/sec]) 또는 저크값을 설정한다.
.
본 매뉴얼에서 사용된 함수들의 공통적인 인자들은 다음과 같은 의미를 가진다.
long lAxis : 특정 축의 번호를 의미 하며, 초기화 함수에 의해 얻어진 채널(축) 번호(0부터 시작함)이다. 채널(축) 번호(0부터 시작함)의 시작은 '0'부터 시작한다.
DWORD uLevel : 신호 설정 등에서의 신호의 출력 레벨 또는 입력 레벨을 뜻하며 전기적으로 HIGH(로직 '1'), 또는 LOW(로직 '0')의 값을 가진다. 또는 신호의 사용 유무 (Enable/ Disable)의 값을 가지기도 한다.
double dPos : 절대 모드에서는 위치값을 의미하며, 상대 모드에서는 이동해야 할 거리를 의미한다. 단위는 단위 거리다.
double dVelocity : 이동 속도를 의미하며, 단위는 초당 단위 거리 이다. 즉, 단위 거리가 1mm로 정해져 있다면, 함수의 인자로 넣는 velocity의 값은 mm/sec의 값이 된다.
double dAcceleration : 이동 가속도(가속도의 단위는 Unit/pulse를 1/1로 한경우에 PPS[Pulses/sec]) 또는 가속시간을 의미한다. 이동 가속도(가속도의 단위는 Unit/pulse를 1/1로 한경우에 PPS[Pulses/sec])일 경우 단위는 초당 속도의 변화량이다. 여기에서의 속도는 함수의 인자로 쓰이는 velocity이다. acceleration의 값이 velocity값의 2배가 되면 최고 속도까지의 가속 시간이 0.5초라는 것을 의미한다. 가속시간일 경우 단위는 초이다. 가속시간을 설정하면 자동으로 정해진 가속률 값으로 구동하게 된다. 가속시간을 2초로 했을 경우 가속율은 최고 속도의 1/2배 값이며, 구동 시작 시점부터 최고 속도까지 도달하는 가속 시간이 2초라는 것을 의미한다.

용어 일람표

Original Word	Axl Word
Position	Pos
Velocity	Vel
Acceleration	Accel
Deceleration	Decel
Command	Cmd
Actual	Act

Revolution	Rev
Coordinate	Coord
Mapping	Map
Information	Info
Motion Parameter Motion Profile	Mot
Continuous	Conti
Absolute	Abs
Relative	Rel
Manual Pulse Generation	MPG
Current Residual Clear	Crc
Number	Num

Motion Command Quick List

에러코드 테이블 (Error Code Table) 확인

함수리스트

노드 및 모듈 확인함수(Info) – Infomation

가상축 함수(Virtual)

모션파라메타 설정함수(Mot) -> Motor Parameter

입출력 설정함수(Signal)

모션 구동후에 상태 확인함수(Status)

위치 구동 함수(Move)

위치 및 속도 오버라이드 구동 함수(Override)

MASTER, SLAVE 구동 함수(Link)

겐트리 구동 함수(Gantry)

보간 구동 함수(Line, Circle)

트리거 함수 (Trigger)

CRC(잔여 펄스 클리어) 구동 함수

MPG(Manul pulse Generator) 구동 함수

네트워크 상태에 대한 모션 구동 에러 처리

Define 문

함수리스트

노드 및 모듈 확인함수(Info) – Infomation
가상축 함수(Virtual)
모션파라메타 설정함수(Mot) -> Motor Parameter
입출력 설정함수(Signal)
모션 구동후에 상태 확인함수(Status)
위치 구동 함수(Move)
위치 및 속도 오버라이드 구동 함수(Override)
MASTER, SLAVE 구동 함수(Link)
겐트리 구동 함수(Gantry)
보간 구동 함수(Line, Circle)
트리거 함수 (Trigger)
CRC(잔여 펄스 클리어) 구동 함수
MPG(Manul pulse Generator) 구동 함수
네트워크 상태에 대한 모션 구동 에러 처리

노드 및 모듈 확인함수(Info) – Information

Function	Description	비고
AxmInfoGetAxis	해당 축의 노드번호, 모듈 위치, 모듈 아이디를 반환한다.	
AxmInfoIsMotionModule	모션 모듈이 존재하는지 반환한다.	
AxmInfoIsInvalidAxisNo	해당 축이 유효한지 반환한다.	
AxmInfoGetAxisCount	시스템에 장착된 유효한 모션 축수를 반환한다.	
AxmInfoGetFirstAxisNo	해당 노드/모듈의 첫번째 축번호를 반환한다.	

가상축 함수(Virtual)

Function	Description	비고
AxmVirtualSetAxisNoMap	가상축을 설정한다.	
AxmVirtualGetAxisNoMap	설정한 가상채널(축) 번호를 반환한다.	
AxmVirtualSetMultiAxisNoMap	멀티 가상축을 설정한다.	
AxmVirtualGetMultiAxisNoMap	설정한 멀티 가상채널(축) 번호를 반환한다.	
AxmVirtualResetAxisMap	가상축 설정을 해지한다.	

모션파라메타 설정함수(Mot) -> Motor Parameter

Function	Description	비고
AxmMotLoadParaAll	AxmMotSaveParaAll로 저장 되어진 33가지파라메타를 파일을 불러온다. 사용자가 임의로 불러서 변경가능하다.	
AxmMotSaveParaAll	현재 모든축에 대한 모든 33가지 파라메타를 축별로 저장한다. .mot파일로 저장한다. AxmM otLoadParaAll 사용해서 파일을 불러온다.	
AxmMotSetParaLoad	EIP-M2/4Q에 설정되어있는 정보를 파일로 저장 하기 위해서는 AxmMotSaveParaAll 함수를 사용한다. AxmMotSaveParaAll 함수는 모든 축의 정보를 파일로 저장하는 함수이다. 여기 파라메타중 28 - 31 까지 파라메타 dInitPos, dInitVel, dInitAccel, dInitDecel 4개 파라메타만 설정한다. 나머지 파라메타는 라이브러리를 닫을 때 현재상태를 읽어서	

	AxmMotSaveParaAll 함수를 이용하여 저장시킨다.	
AxmMotGetParaLoad	EIP-M2/4Q의 레지스터에 설정되어있는 정보를 파일로 저장 하기 위해서는 AxmMotSaveParaAll 함수를 사용한다. AxmMotSaveParaAll 함수는 모든 축의 정보를 파일로 저장한 파라메타중에 28 - 31 까지 파라메타 dInitPos, dInitVel, dInitAccel, dInitDecel, 4개 파라메타만 불러 올 때 사용하는 함수이다.	
AxmMotSetPulseOutMethod	지정 축의 펄스 출력 방식을 설정한다.	
AxmMotGetPulseOutMethod	지정 축의 펄스 출력 방식 설정을 반환한다,	
AxmMotSetEncInputMethod	지정 축의 외부(Actual) 카운트의 증가 방향 설정을 포함하여 지정 축의 Encoder 입력 방식을 설정한다.	
AxmMotGetEncInputMethod	지정 축의 외부(Actual) 카운트의 증가 방향 설정을 포함하여 지정 축의 Encoder 입력 방식을 반환한다.	
AxmMotSetMoveUnitPerPulse	지정 축의 펄스 당 움직이는 거리를 설정한다.	
AxmMotGetMoveUnitPerPulse	지정 축의 펄스 당 움직이는 거리를 반환한다.	
AxmMotSetDecelPos	지정 축에 감속 시작 포인트 검출 방식을 설정한다.	
AxmMotGetDecelPos	지정 축의 감속 시작 포인트 검출 방식을 반환한다	
AxmMotSetRemainPulse	지정 축에 수동 감속 모드에서 잔량 펄스를 설정한다.	
AxmMotGetRemainPulse	지정 축의 수동 감속 모드에서 잔량 펄스를 반환한다.	
AxmMotSetMaxVel	지정 축에 등속도 구동 함수에서의 최고 속도를 설정한다.	
AxmMotGetMaxVel	지정 축의 등속도 구동 함수에서의 최고 속도를 반환한다.	
AxmMotSetAbsRelMode	지정 축의 이동 거리 계산 모드를 설정한다. 절대위치 모드(0)로 설정한 경우 이동 함수에 사용하	

	는 위치값은 절대위치값이 된다. 예를 들어 현재 위치값이 '10'일 때 절대위치 모드에서 '100' 위치로 이동할 경우 현재 위치에서 '+90'만큼 이동하여 '100'위치에서 정지하게 된다. 상대위치 모드(1)로 설정한 경우는 이동 함수에 사용하는 위치값은 현재 위치에서 이동할 이동량이 된다. 예를 들어 현재 위치값이 '10일 대 상대 위치 모드에서 '100' 만큼 이동할 경우, 현재 위치에서 '+100'만큼 이동하여 '110'의 위치에서 정지하게 된다.	
AxmMotGetAbsRelMode	지정 축의 설정된 이동 거리 계산 모드를 반환한다	
AxmMotSetProfileMode	지정 축의 구동 속도 프로파일 모드를 설정한다. 속도 프로파일 모드에는 대칭 또는 비대칭Trapezoide / S-Curve 모드, Quasi-S 모드가 있다.	
AxmMotGetProfileMode	지정 축의 설정한 구동 속도 프로파일 모드를 반환한다.	
AxmMotSetAccelUnit	지정 축의 가속도 단위를 설정한다. 가속도 단위는 u/s^2, sec, rev/s^2 중 선택할 수 있다.	
AxmMotGetAccelUnit	지정 축의 설정된 가속도단위를 반환한다.	
AxmMotSetMinVel	지정 축에 초기 속도를 설정한다.	
AxmMotGetMinVel	지정 축의 초기 속도를 반환한다.	
AxmMotSetAccelJerk	지정 축의 가속 저크값을 설정한다.	
AxmMotGetAccelJerk	지정 축의 설정된 가속 저크값을 반환한다.	
AxmMotSetDecelJerk	지정 축의 감속 저크값을 설정한다.	
AxmMotGetDecelJerk	지정 축의 설정된 감속 저크값을 반환한다.	

입출력 설정함수(Signal)

Function	Description	비고
AxmSignalSetZphaseLevel	지정 축의 Z 상 Level을 설정한다.	
AxmSignalGetZphaseLevel	지정 축의 Z 상 Level을 반환한다.	
AxmSignalSetServoOnLevel	지정 축의 Servo-On신호의 출력 레벨을 설정한다.	
AxmSignalGetServoOnLevel	지정 축의 Servo-On신호의 출력 레벨 설정을 반환한다.	
AxmSignalSetServoAlarmResetLevel	지정 축의 Servo-Alarm Reset 신호의 출력 레벨을 설정한다.	
AxmSignalGetServoAlarmResetLevel	지정 축의 Servo-Alarm Reset 신호의 출력 레벨을 설정을 반환한다.	
AxmSignalSetInpos	지정 축의 Inpositon 신호 사용 여부 및 신호 입력 레벨을 설정한다	
AxmSignalGetInpos	지정 축의 Inpositon 신호 사용 여부 및 신호 입력 레벨을 반환한다.	
AxmSignalReadInpos	지정 축의 Inpositon 신호 입력 상태를 반환한다.	
AxmSignalSetServoAlarm	지정 축의 알람 신호 입력 시 비상 정지의 사용 여부 및 신호 입력 레벨을 설정한다.	
AxmSignalGetServoAlarm	지정 축의 알람 신호 입력 시 비상 정지의 사용 여부 및 신호 입력 레벨을 반환한다.	
AxmSignalReadServoAlarm	지정 축의 알람 신호의 입력 레벨을 반환한다.	
AxmSignalSetLimit	지정 축의 end limit sensor의 사용 유무 및 신호의 입력 레벨을 설정한다. end limit sensor 신호 입력 시 감속정지 또는 급정지에 대한 설정도 가능하다.	
AxmSignalGetLimit	지정 축의 end limit sensor의 사용 유무 및 신호의	

	입력 레벨, 신호 입력 시 정지모드를 반환한다	
AxmSignalReadLimit	지정축의 end limit sensor의 입력 상태를 반환한다.	
AxmSignalSetSoftLimit	지정 축의 Software limit의 사용 유무, 사용할 카운트, 그리고 정지 방법을 설정한다	
AxmSignalGetSoftLimit	지정 축의 Software limit의 사용 유무, 사용할 카운트, 그리고 정지 방법을 반환한다	
AxmSignalSetStop	비상 정지 신호의 정지 방법 (급정지/감속정지) 또는 사용 유무를 설정한다.	
AxmSignalGetStop	비상 정지 신호의 정지 방법 (급정지/감속정지) 또는 사용 유무를 반환한다.	
AxmSignalReadStop	비상 정지 신호의 입력 상태를 반환한다.	
AxmSignalServoOn	지정 축의 Servo-On 신호를 출력한다.	
AxmSignalIsServoOn	지정 축의 Servo-On 신호의 출력 상태를 반환한다.	
AxmSignalServoAlarmReset	지정 축의 Servo-Alarm Reset 신호를 출력한다.	
AxmSignalWriteOutput	범용 출력값을 설정한다.	
AxmSignalReadOutput	범용 출력값을 반환한다.	
AxmSignalWriteOutputBit	범용 출력값을 비트별로 설정한다.	
AxmSignalReadOutputBit	범용 출력값을 비트별로 반환한다.	
AxmSignalReadInput	범용 입력값을 반환한다.	
AxmSignalReadInputBit	범용 입력값을 비트별로 반환한다.	

모션 구동 후에 상태 확인함수(Status)

Function	Description	비고
AxmStatusReadInMotion	지정 축의 펄스 출력 상태를 반환한다.	
AxmStatusReadDrivePulseCount	지정 축의 구동 펄스 카운터 값을 반환한다.	
AxmStatusReadMotion	지정 축의 DriveStatus 레지스터를 반환한다.	
AxmStatusReadStop	지정 축의 EndStatus 레지스터를 반환한다.	
AxmStatusReadMechanical	지정 축의 Mechanical Signal Data를 반환한다.	
AxmStatusReadVel	지정 축의 현재 구동 속도를 읽어온다.	
AxmStatusReadPosError	지정 축의 Command Position과 Actual Position의 차를 반환한다.	
AxmStatusReadDriveDistance	최종 드라이브로 이동하는(이동한) 거리를 확인 한다	
AxmStatusSetActPos	지정 축의 Actual 위치를 설정한다.	
AxmStatusGetActPos	지정 축의 Actual 위치를 반환한다.	
AxmStatusSetCmdPos	지정 축의 Command 위치를 설정한다.	
AxmStatusGetCmdPos	지정 축의 Command 위치를 반환한다.	

홈(원점) 검색 함수(Home)

Function	Description	비고
AxmHomeSetSignalLevel	지정 축의 Home 센서 Level 을 설정한다.	
AxmHomeGetSignalLevel	지정 축의 Home 센서 Level 을 반환한다.	
AxmHomeReadSignal	지정 축의 Home 신호의 입력 상태를 반환한다.	

AxmHomeSetMethod	해당 축의 원점검색을 수행하기 위해서는 반드시 원점검색관련 파라메타들이 설정되어 있어야 된다. AxmMotSaveParaAll, AxmMotLoadParaAll 이용하여 각종 원점검색 파라메타를 저장하고 불러올수있다. 원점검색 방법 설정에는 검색 진행방향, 원점으로 사용할 신호, 원점센서 Active Level, 엔코더 Z상 검출 여부 등을 설정 한다.	
AxmHomeGetMethod	해당 축의 원점검색을 수행하기 위해서는 반드시 원점검색관련 파라메타들이 설정되어 있어야 된다. 설정되어있는 홈 관련 파라메타들을 반환한다.	
AxmHomeSetVel	원점을 빠르고 정밀하게 검색하기 위해 여러 단계의 스탭으로 검출한다. 이때 각 스탭에 사용 될 속도를 설정한다. 이 속도들의 설정값에 따라 원점검색 시간과, 원점검색 정밀도가 결정된다. 각 스탭별 속도들을 적절히 바꿔가면서 각 축의 원점검색 속도를 설정하면 된다.	
AxmHomeGetVel	원점을 빠르고 정밀하게 검색하기 위해 여러 단계의 스탭으로 검출한다. 이때 각 스탭에 사용 될 속도를 설정한다. 이 속도들의 설정값에 따라 원점검색 시간과, 원점검색 정밀도가 결정된다. 설정된 각각 속도를 가져온다.	
AxmHomeSetStart	원점검색 시작함수를 실행하면 라이브러리 내부에서 해당축의 원점검색을 수행 할 쓰레드가 자동 생성되어 원점검색을 순차적으로 수행한 후 자동 종료된다.	
AxmHomeSetResult	원점검색 함수를 이용해 성공적으로 원점검색이 수행되고나면 검색 결과가 HOME_SUCCESS로 설정된다. 이 함수는 사용자가 원점검색을 수행하지않고 결과를 임의로 설정할 수 있다. 실제 원점검색 완료 후 정상 구동중 예외상황이 발생	

	하면원점검색결과를 HOME_ERR_USER _BRE AK 등으로 설정해 원점검색을 다시 수행하도록 할 수 있다.	
AxmHomeGetResult	원점검색 함수의 검색 결과를 확인한다. 원점검색이 시작되면 HOME_SEARCHING으로 설정되며 원점검색에 실패하면 실패원인이 설정된다. 실패 원인을 제거한 후 다시 원점검색을 진행하면 된다.	
AxmHomeGetRate	원점검색 시작되면 진행율을 확인할 수 있다. 원점검색이 완료되면 성공여부와 관계없이 100을 반환하게 된다. 원점검색 성공여부는 GetHome Result함수를 이용해 확인할 수 있다.	

위치 구동 함수(Move)

Function	Description	비고
AxmMoveStartPos	지정 축의 절대 좌표/ 상대좌표로 설정된 위치까지 설정된 속도와 가속율로 구동을 한다. 속도 프로파일은 AxmMotSetProfileMode 함수에서 설정한다. 펄스가 출력되는 시점에서 함수를 벗어난다.	
AxmMovePos	지정 축의 절대 좌표/상대좌표로 설정된 위치까지 설정된 속도와 가속율로 구동을 한다. 속도 프로파일은 AxmMotSetProfileMode 함수에서 설정한다. 펄스 출력이 종료되는 시점에서 함수를 벗어난다.	
AxmMoveVel	지정 축에 대하여 설정된 속도와 가속율로 지속적으로 속도 모드 구동을 한다. 펄스 출력이 시작되는 시점에서 함수를 벗어난다.	
AxmMoveStartMultiVel	지정된 다축에 대하여 설정된 속도와 가속율로 지속적으로 속도 모드 구동을 한다. 펄스 출력이 시작되는 시점에서 함수를 벗어난다.	
AxmMoveSignalSearch	설정된 속도와 가속율로 설정된 신호가 검출될 때까지 구동을 한다. 펄스가 출력되는 시점에서 함수를 벗어난다.	
AxmMoveSignalCapture	지정 축에서 설정된 신호를 검출하고 그 위치를 저장하기 위해 이동하는 함수이다.	
AxmMoveGetCapturePos	‘AxmMoveSignalCapture’ 함수에서 저장된 위치값을 확인하는 함수이다.	
AxmMoveStartMultiPos	지정 축들의 절대 좌표로 설정된 위치까지 설정된 속도와 가속율로 구동을 한다. 속도 프로파일은 AxmMotSetProfileMode(함수에서 설정한다. 펄스 출력이 종료되는 시점에서 함수를 벗어	

	난다.	
AxmMoveMultiPos	지정 축들의 절대 좌표로 설정된 위치까지 설정된 속도와 가속율로 구동을 한다. 속도 프로파일은 AxmMotSetProfileMode(함수에서 설정한다. 펄스가 출력되는 시점에서 함수를 벗어난다.	
AxmMoveStop	지정 축을 설정한 감속도로 감속 정지 한다.	
AxmMoveEStop	지정 축을 급 정지 한다.	
AxmMoveSStop	지정 축을 감속 정지한다.	

위치 및 속도 오버라이드 구동 함수(Override)

Function	Description	비고
AxmOverridePos	지정 축의 구동이 종료되기 전 지정된 출력 펄스 수를 조정한다.	
AxmOverrideSetMaxVel	지정 축의 속도오버라이드 하기전에 오버라이드할 최고속도를 설정한다.	
AxmOverrideVel	지정 축의 구동 중에 속도를 가변 설정한다.	
AxmOverrideAccelVelDecel	설정 축의 모션 동작 중 구동 속도를 변경하는 함수이다	
AxmOverrideVelAtPos	지정 축에 대하여 설정된 속도와 가속율로 지속적으로 속도 모드 구동을 한다. 구동 중 설정한 위치에서 속도오버라이드를 실행한다. 펄스 출력이 시작되는 시점에서 함수를 벗어난다.	

MASTER, SLAVE 구동 함수(Link)

Function	Description	비고
AxmLinkSetMode	Master 축과 Slave축간의 전자기어비를 설정한다.	
AxmLinkGetMode	Master 축과 Slave축간의 전자기어비를 반환한다.	
AxmLinkResetMode	Master 축과 Slave축간의 전자기어비를 설정 해제 한다.	

겐트리 구동 함수(Gantry)

Function	Description	비고
AxmGantrySetEnable	모션모듈은 두 축이 기구적으로 Link되어있는 겐트리 구동시스템 제어를 지원한다. 이 함수를 이용해 Master축을 겐트리 제어로 설정하면 해당 Slave축은 Master축과 동기되어 구동된다. 만약 겐트리 설정 이후 Slave축에 구동명령이나 정지 명령등을 내려도 모두 무시된다.	
AxmGantryGetEnable	이 함수를 이용해 Master축을 겐트리 제어로 설정할 때 설정한 파라메타를 반환한다.	
AxmGantrySetDisable	겐트리 제어로 설정후 원점검색을 수행하면 Master축이 원점검색을 수행한 후 Slave축이 원점검색을 수행한다. 이 함수는 원점검색 이후 Master축 원점과 Slave축 원점간의 오차 값을 확인 할 때 사용한다.	

보간 구동 함수(Line, Circle)

Function	Description	비고
AxmLineMove	시작점과 종료점을 지정하여 다축 직선 보간 구동하는 함수이다. 구동 시작 후 함수를 벗어난다.	

AxmLineArcMove	시작점(x,y,z) , 중간점(x,y,z) , 끝점(x,y,z) , 중간점에서의 거리(d) 를 넣고 직선보간을 이용하여 원호 보간을 한다.	
AxmCircleCenterMove	시작점, 종료점과 중심점을 지정하여 원호 보간 구동하는 함수이다. 구동 시작 후 함수를 벗어난다.	
AxmCirclePointMove	중간점, 종료점을 지정하여 원호 보간 구동하는 함수이다. 구동 시작 후 함수를 벗어난다.	
AxmCircleRadiusMove	시작점, 종료점과 반지름을 지정하여 원호 보간 구동하는 함수이다. 구동 시작 후 함수를 벗어난다.	
AxmCircleAngleMove	시작점, 회전각도와 반지름을 지정하여 원호 보간 구동하는 함수이다. 구동 시작 후 함수를 벗어난다.	
AxmContiSetAxisMap	지정된 좌표계에 보간 축 맵핑을 설정한다.	
AxmContiGetAxisMap	지정된 좌표계에 보간 축 맵핑을 반환한다.	
AxmContiSetAbsRelMode	지정된 좌표계에 보간 축 절대/상대 모드를 설정한다.	
AxmContiGetAbsRelMode	지정된 좌표계에 보간 축 절대/상대 모드를 반환한다.	
AxmContiIsMotion	보간 구동 중인지 확인 하는 함수이다.	

트리거 함수 (Trigger)

Function	Description	비고
AxmTriggerSetTimeLevel	지정 축에 트리거 신호 지속 시간 및 트리거 출력 레벨, 트리거 출력방법을 설정한다.	
AxmTriggerGetTimeLevel	지정 축에 트리거 신호 지속 시간 및 트리거 출력 레벨, 트리거 출력방법을 반환한다.	

AxmTriggerSetAbsPeriod	지정 축에 트리거 기능의 사용 여부, 출력 레벨, 위치 비교기, 트리거 신호 지속 시간 및 트리거 출력 모드를 설정한다.	
AxmTriggerGetAbsPeriod	지정 축에 트리거 기능의 사용 여부, 출력 레벨, 위치 비교기, 트리거 신호 지속 시간 및 트리거 출력 모드를 반환한다.	
AxmTriggerSetBlock	사용자가 지정한 시작위치부터 종료위치까지 일정구간마다 트리거를 출력하는 한다.	
AxmTriggerGetBlock	'AxmTriggerSetBlock' 함수의 트리거 설정한 값을 읽는다..	
AxmTriggerOneShot	사용자가 한 개의 트리거 펄스를 출력한다..	
AxmTriggerOnlyAbs	절대위치 트리거 무한대 절대위치 출력한다.	
AxmTriggerSetReset	트리거 설정을 리셋한다.	

CRC(잔여 펄스 클리어) 구동 함수

Function	Description	비고
AxmCrcSetMaskLevel	지정 축에 CRC 신호 사용 여부 및 출력 레벨을 설정한다.	
AxmCrcGetMaskLevel	지정 축의 CRC 신호 사용 여부 및 출력 레벨을 반환한다.	
AxmCrcSetOutput	지정 축에 CRC 신호를 강제로 발생 시킨다.	
AxmCrcGetOutput	지정 축의 CRC 신호를 강제로 발생 여부를 반환한다.	

MPG(Manul pulse Generator) 구동 함수

Function	Description.	비고
AxmMPGSetEnable	지정 축에 MPG 입력방식, 드라이브 구동 모드, 이동	

	거리, 속도 등을 설정한다.	
AxmMPGGetEnable	지정 축에 MPG 입력방식, 드라이브 구동 모드, 이동 거리, 속도 등을 반환한다.	
AxmMPGSetRatio	지정 축에 MPG 드라이브 구동 모드에서 한펄스당 이동할 펄스 비율을 설정한다.	
AxmMPGGetRatio	지정 축에 MPG 드라이브 구동 모드에서 한펄스당 이동할 펄스 비율을 반환한다.	
AxmMPGReset	지정 축에 MPG 드라이브 설정을 해지한다.	

네트워크 상태에 대한 모션 구동 에러 처리

Function	Description	비고
AxmNetWorkErrorSetAction	지정 축이 연결된 노드의 통신이 끊어 졌을때 현재 모션 동작 상태에 대한 예외처리 방법을 설정한다	
AxmNetWorkErrorGetAction	지정 축이 연결된 노드의 통신이 끊어 졌을때 현재 모션 동작 상태에 대한 예외처리 방법을 확인한다	

Define 문

AXL Motion Library에서 사용하고 Define문 List는 다음과 같다
헤더파일 : #include "AXHS.h"

FALSE & TRUE

define	Value	Explanation
FALSE	00h	거짓
TRUE	01h	참

DISABLE & ENABLE

define	Value	Explanation
DISABLE	00h	사용안함
ENABLE	01h	사용함

AXT_MOTION_STOPMODE_DEF

#define	Value	Explanation
EMERGENCY_STOP	00h	급정지
SLOWDOWN_STOP	01h	감속정지

AXT_MOTION_EDGE_DEF

#define	Value	Explanation
SIGNAL_DOWN_EDGE	00h	다운 에지
SIGNAL_UP_EDGE	01h	업 에지

AXT_MOTION_SELECTION_DEF

#define	Value	Explanation
COMMAND	00h	목표 위치
ACTUAL	01h	실제 위치

AXT_MOTION_TRIGGER_MODE_DEF

#define	Value	Explanation

PERIOD_MODE	00h	트리거주기 모드
ABS_POS_MODE	01h	트리거절대 모드

AXT_MOTION_LEVEL_MODE_DEF

#define	Value	Explanation
LOW	00h	B 접점(NORMAL CLOSE)
HIGH	01h	A 접점(NORMAL OPEN)
UNUSED	02h	사용안함
USED	03h	현상태유지

AXT_MOTION_ABSREL_MODE_DEF

#define	Value	Explanation
POS_ABS_MODE	00h	위치구동 절대모드
POS_REL_MODE	01h	위치구동 상대모드

AXT_MOTION_PROFILE_MODE_DEF

#define	Value	Explanation
SYM_TRAPEZOIDE_MODE	00h	대칭 Trapezode
ASYM_TRAPEZOIDE_MODE	01h	비대칭 Trapezode
QUASI_S_CURVE_MODE	02h	대칭 Quasi-S Curve (지원 안함)
SYM_S_CURVE_MODE	03h	대칭 S Curve
ASYM_S_CURVE_MODE	04h	비대칭 S Curve

AXT_MOTION_SELECTION_DEF

#define	Value	Explanation
INACTIVE	00h	비활성화
ACTIVE	01h	활성화

AXT_MOTION_HOME_RESULT_DEF

#define	Value	Explanation
HOME_SUCCESS	01h	원점 검색이 성공적으로 종료 됐을 경우

HOME_SEARCHING	02h	현재 원점검색이 진행중일 경우
HOME_ERR_GNT_RANGE	10h	겐트리 구동축의 Master축과 Slave축의 원점검색 결과가 설정한 OffsetRange를 벗어났을경우
HOME_ERR_USER_BREAK	11h	원점 검색중 사용자가 정지 명령을 실행했을 경우
HOME_ERR_VELOCITY	12h	원점 검색속도 설정이 되어있지 않을경우
HOME_ERR_AMP_FAULT	13h	원점 검색중 서보팩 알람이 발생할 경우
HOME_ERR_NEG_LIMIT	14h	(+)방향으로 원점센서 검출 중 (-)리미트 센서가 감지될 경우
HOME_ERR_POS_LIMIT	15h	(-)방향으로 원점센서 검출 중 (+)리미트 센서가 감지될 경우
HOME_ERR_NOT_DETECT	16h	원점센서가 감지되지 않을경우
HOME_ERR_UNKNOWN	FFh	알수없는 채널(축) 번호(0부터 시작함)로 원점검색을 시작하려 했을경우

AXT_MOTION_UNIV_INPUT_DEF

#define	Value	Explanation
UIO_INP0	00h	범용 입력 0
UIO_INP1	01h	범용 입력 1
UIO_INP2	02h	범용 입력 2
UIO_INP3	03h	범용 입력 3
UIO_INP4	04h	범용 입력 4

AXT_MOTION_UNIV_OUTPUT_DEF

#define	Value	Explanation
UIO_OUT0	00h	범용 출력 0
UIO_OUT1	01h	범용 출력 1
UIO_OUT2	02h	범용 출력 2
UIO_OUT3	03h	범용 출력 3
UIO_OUT4	04h	범용 출력 4

AXT_MOTION_DETECT_DOWN_START_POINT_DEF
감속시작 포인트 검출방식 지정

#define	Value	Explanation
AutoDetect	00h	자동 검출 방식
RestPulse	01h	나머지 검출 방식(수동가감속)

AXT_MOTION_PULSE_OUTPUT_DEF

#define	Value	Explanation
OneHighLowHigh	00h	1펄스 방식, PULSE(Active High), 정방향 (DIR=Low) / 역방향(DIR=High)
OneHighHighLow	01h	1펄스 방식, PULSE(Active High), 정방향 (DIR=High) / 역방향(DIR=Low)
OneLowLowHigh	02h	1펄스 방식, PULSE(Active Low), 정방향 (DIR=Low) / 역방향(DIR=High)
OneLowHighLow	03h	1펄스 방식, PULSE(Active Low), 정방향 (DIR=High) / 역방향(DIR=Low)
TwoCcwCwHigh	04h	2펄스 방식, PULSE(CCW:역방향), DIR(CW:정방향), Active High
TwoCcwCwLow	05h	2펄스 방식, PULSE(CCW:역방향), DIR(CW:정방향), Active Low
TwoCwCcwHigh	06h	2펄스 방식, PULSE(CW:정방향), DIR(CCW:역방향), Active High
TwoCwCcwLow	07h	2펄스 방식, PULSE(CW:정방향), DIR(CCW:역방향), Active Low
TwoPhase	08h	2상(90' 위상차), PULSE lead DIR(CW: 정방향), PULSE lag DIR(CCW:역방향)
TwoPhaseReverse	09h	2상(90' 위상차), PULSE lead DIR(CCW: 정방향), PULSE lag DIR(CW:역방향)

AXT_MOTION_EXTERNAL_COUNTER_INPUT_DEF

Encoder 입력 방식

#define	Value	Explanation
ObverseUpDownMode	00h	정방향 Up / Down
ObverseSqr1Mode	01h	정방향 1체배
ObverseSqr2Mode	02h	정방향 2체배
ObverseSqr4Mode	03h	정방향 4체배
ReverseUpDownMode	04h	역방향 Up / Down
ReverseSqr1Mode	05h	역방향 1체배
ReverseSqr2Mode	06h	역방향 2체배
ReverseSqr4Mode	07h	역방향 4체배

AXT_MOTION_ACC_UNIT_DEF

#define	Value	Explanation
UNIT_SEC2	00h	unit/sec^2
SEC	01h	sec

AXT_MOTION_MOVE_DIR_DEF

#define	Value	Explanation
DIR_CCW	00h	반시계방향
DIR_CW	01h	시계방향

AXT_MOTION_RADIUS_DISTANCE_DEF

#define	Value	Explanation
SHORT_DISTANCE	00h	짧은 거리의 원호 이동
LONG_DISTANCE	01h	긴 거리의 원호 이동

AXT_MOTION_INTERPOLATION_AXIS_DEF

#define	Value	Explanation
INTERPOLATION_AXIS2	00h	2축을 보간으로 사용할 때
INTERPOLATION_AXIS3	01h	3축을 보간으로 사용할 때
INTERPOLATION_AXIS4	04h	4축을 보간으로 사용할 때

AXT_MOTION_HOME_DETECT_SIGNAL_DEF

#define	Value	Explanation
PosEndLimit	00h	+Elm(End limit) +방향 리미트 센서 신호
NegEndLimit	01h	-Elm(End limit) -방향 리미트 센서 신호
PosSloLimit	02h	+Slm(Slow Down limit) 신호 - 사용하지 않음
NegSloLimit	03h	-Slm(Slow Down limit) 신호 - 사용하지 않음
HomeSensor	04h	IN0(ORG) 원점 센서 신호
EncodZPhase	05h	IN1(Z상) Encoder Z상 신호
UniInput02	06h	IN2(범용) 범용 입력 2번 신호
UniInput03	07h	IN3(범용) 범용 입력 3번 신호

AXT_MOTION_IPDETECT_DESTINATION_SIGNAL

#define	Value	Explanation
PElmNegativeEdge	00h	+Elm(End limit) 하강 edge
NElmNegativeEdge	01h	-Elm(End limit) 하강 edge
PSlmNegativeEdge	02h	+Slm(Slow Down limit) 신호 - 사용하지 않음
NSlmNegativeEdge	03h	-Slm(Slow Down limit) 신호 - 사용하지 않음
In0DownEdge	04h	IN0(ORG) 하강 edge
In1DownEdge	05h	IN1(Z상) 하강 edge
In2DownEdge	06h	IN2(범용) 하강 edge
In3DownEdge	07h	IN3(범용) 하강 edge
PElmPositiveEdge	00h	+Elm(End limit) 상승 edge
NElmPositiveEdge	01h	-Elm(End limit) 상승 edge
PSlmPositiveEdge	02h	+Slm(Slow Down limit) 신호 - 사용하지 않음
NSlmPositiveEdge	03h	-Slm(Slow Down limit) 신호 - 사용하지 않음
In0UpEdge	04h	IN0(ORG) 상승 edge
In1UpEdge	05h	IN1(Z상) 상승 edge
In2UpEdge	06h	IN2(범용) 상승 edge
In3UpEdge	07h	IN3(범용) 상승 edge

AXT_MOTION_QIDETECT_DESTINATION_SIGNAL

#define	Value	Explanation
Signal_ PosEndLimit	00h	+Elm(End limit) +방향 리미트 센서 신호
Signal_ NegEndLimit	01h	-Elm(End limit) -방향 리미트 센서 신호
Signal_ PosSloLimit	02h	+Slm(Slow Down limit) 신호 - 사용하지 않음
Signal_ NegSloLimit	03h	-Slm(Slow Down limit) 신호 - 사용하지 않음
Signal_ HomeSensor	04h	IN0(ORG) 원점 센서 신호
Signal_ EncodZPhase	05h	IN1(Z상) Encoder Z상 신호
Signal_ UniInput02	06h	IN2(범용) 범용 입력 2번 신호
Signal_ UniInput03	07h	IN3(범용) 범용 입력 3번 신호

AXT_MOTION_MPG_INPUT_METHOD

#define	Value	Explanation
MPG_DIFF_ONE_PHASE	00h	MPG 입력 방식 One Phase
MPG_DIFF_TWO_PHASE_1X	01h	MPG 입력 방식 TwoPhase1
MPG_DIFF_TWO_PHASE_2X	02h	MPG 입력 방식 TwoPhase2
MPG_DIFF_TWO_PHASE_4X	03h	MPG 입력 방식 TwoPhase4
MPG_LEVEL_ONE_PHASE	04h	MPG 입력 방식 Level One Phase
MPG_LEVEL_TWO_PHASE_1X	05h	MPG 입력 방식 Level Two Phase1
MPG_LEVEL_TWO_PHASE_2X	06h	MPG 입력 방식 Level Two Phase2
MPG_LEVEL_TWO_PHASE_4X	07h	MPG 입력 방식 Level Two Phase4

AXT_MOTION_SENSOR_INPUT_METHOD

#define	Value	Explanation
SENSOR_METHOD1	00h	일반 구동
SENSOR_METHOD2	01h	센서 신호 검출 전은 저속 구동. 신호 검출 후 일반 구동
SENSOR_METHOD3	02h	저속 구동

AXT_MOTION_HOME_CRC_SELECT

#define	Value	Explanation
CRC_SELECT1	00h	위치클리어 사용않함, 잔여펄스 클리어 사용 안함
CRC_SELECT2	01h	위치클리어 사용함, 잔여펄스 클리어 사용 안함
CRC_SELECT3	02h	위치클리어 사용안함, 잔여펄스 클리어 사용함
CRC_SELECT4	03h	위치클리어 사용함, 잔여펄스 클리어 사용함

AXT_MOTION_QIMECHANICAL_SIGNAL_DEF

Mechanical Signal 레지스터

#define	Value	Explanation
QIMECHANICAL_PELM_LEVEL	0001h	Bit 0, +Limit 급정지 신호 현재 상태
QIMECHANICAL_NELM_LEVEL	0002h	Bit 1, -Limit 급정지 신호 현재 상태
QIMECHANICAL_PSLM_LEVEL	0004h	Bit 2, +limit 감속정지 현재 상태.
QIMECHANICAL_NSLM_LEVEL	0008h	Bit 3, -limit 감속정지 현재 상태
QIMECHANICAL_ALARM_LEVEL	0010h	Bit 4, Alarm 신호 신호 현재 상태
QIMECHANICAL_INP_LEVEL	0020h	Bit 5, InPos 신호 현재 상태
QIMECHANICAL_ESTOP_LEVEL	0040h	Bit 6, 비상 정지 신호(ESTOP) 현재 상태.
QIMECHANICAL_ORG_LEVEL	0080h	Bit 7, 원점 신호 헌재 상태
QIMECHANICAL_ZPHASE_LEVEL	0100h	Bit 8, Z 상 입력 신호 현재 상태
QIMECHANICAL_ECUP_LEVEL	0200h	Bit 9, ECUP 터미널 신호 상태.
QIMECHANICAL_ECDN_LEVEL	0400h	Bit 10, ECDN 터미널 신호 상태.
QIMECHANICAL_EXPP_LEVEL	0800h	Bit 11, EXPP 터미널 신호 상태
QIMECHANICAL_EXMP_LEVEL	1000h	Bit 12, EXMP 터미널 신호 상태
QIMECHANICAL_SQSTR1_LEVEL	2000h	Bit 13, SQSTR1 터미널 신호 상태
QIMECHANICAL_SQSTR2_LEVEL	4000h	Bit 14, SQSTR2 터미널 신호 상태
QIMECHANICAL_SQSTP1_LEVEL	8000h	Bit 15, SQSTP1 터미널 신호 상태
QIMECHANICAL_SQSTP2_LEVEL	10000h	Bit 16, SQSTP2 터미널 신호 상태
QIMECHANICAL_MODE_LEVEL	20000h	Bit 17, MODE 터미널 신호 상태

AXT_MOTION_QIEND_STATUS_DEF

축의 종료 상태

#define	Value	Explanation
	00000000h	구동하지 않음 또는 초기 상태
QIEND_STATUS_0	00000001h	Bit 0, 정방향 리미트 신호(PELM)에 의한 종료
QIEND_STATUS_1	00000002h	Bit 1, 역방향 리미트 신호(NELM)에 의한 종료
QIEND_STATUS_2	00000004h	Bit 2, 정방향 부가 리미트 신호(PSLM)에 의한 구동 종료
QIEND_STATUS_3	00000008h	Bit 3, 역방향 부가 리미트 신호(NSLM)에 의한 구동 종료
QIEND_STATUS_4	00000010h	Bit 4, 정방향 소프트 리미트 급정지 기능에 의한 구동 종료
QIEND_STATUS_5	00000020h	Bit 5, 역방향 소프트 리미트 급정지 기능에 의한 구동 종료
QIEND_STATUS_6	00000040h	Bit 6, 정방향 소프트 리미트 감속정지 기능에 의한 구동 종료
QIEND_STATUS_7	00000080h	Bit 7, 역방향 소프트 리미트 감속정지 기능에 의한 구동 종료
QIEND_STATUS_8	00000100h	Bit 8, 서보 알람 기능에 의한 구동 종료.
QIEND_STATUS_9	00000200h	Bit 9, 비상 정지 신호 입력에 의한 구동 종료.
QIEND_STATUS_10	00000400h	Bit 10, 급 정지 명령에 의한 구동 종료
QIEND_STATUS_11	00000800h	Bit 11, 감속 정지 명령에 의한 구동 종료.
QIEND_STATUS_12	00001000h	Bit 12, 전축 급정지 명령에 의한 구동 종료
QIEND_STATUS_13	00002000h	Bit 13, 동기 정지 기능 #1(SQSTP1)에 의한

		구동 종료
QIEND_STATUS_14	00004000h	Bit 14, 동기 정지 기능 #2(SQSTP2)에 의한 구동 종료.
QIEND_STATUS_15	00008000h	Bit 15, 인코더 입력(ECUP,ECDN) 오류 발생
QIEND_STATUS_16	00010000h	Bit 16, MPG 입력(EXPP,EXMP) 오류 발생
QIEND_STATUS_17	00020000h	Bit 17, 원점 검색 성공 종료
QIEND_STATUS_18	00040000h	Bit 18, 신호 검색 성공 종료.
QIEND_STATUS_19	00080000h	Bit 19, 보간 데이터 이상으로 구동 종료.
QIEND_STATUS_20	00100000h	Bit 20, 비정상 구동 정지발생.
QIEND_STATUS_21	00200000h	Bit 21, MPG 기능 블록 펄스 버퍼 오버플로우 발생
QIEND_STATUS_22 – 27		Bit 22,- Bit27 DON'CARE
QIEND_STATUS_28	10000000h	Bit 28, 현재/마지막 구동 드라이브 방향
QIEND_STATUS_29	20000000h	Bit 29, 잔여 펄스 제거 신호 출력 중
QIEND_STATUS_30	40000000h	Bit 30, 비정상 구동 정지 원인 상태
QIEND_STATUS_31	80000000h	Bit 31, 보간 드라이브 데이타 오류 상태.

AXT_MOTION_QIDRIVE_STATUS_DEF

축의 구동 상태

#define	Value	Explanation
QIDRIVE_STATUS_0	00000001h	Bit 0, BUSY(드라이브 구동 중)
QIDRIVE_STATUS_1	00000002h	Bit 1, DOWN(감속 중)
QIDRIVE_STATUS_2	00000004h	Bit 2, CONST(등속 중)
QIDRIVE_STATUS_3	00000008h	Bit 3, UP(가속 중)

QIDRIVE_STATUS_4	00000010h	Bit 4, 연속 드라이브 구동 중
QIDRIVE_STATUS_5	00000020h	Bit 5, 지정 거리 드라이브 구동 중
QIDRIVE_STATUS_6	00000040h	Bit 6, MPG 드라이브 구동 중
QIDRIVE_STATUS_7	00000080h	Bit 7, 원점검색 드라이브 구동중
QIDRIVE_STATUS_8	00000100h	Bit 8, 신호 검색 드라이브 구동 중
QIDRIVE_STATUS_9	00000200h	Bit 9, 보간 드라이브 구동 중
QIDRIVE_STATUS_10	00000400h	Bit 10, Slave 드라이브 구동중
QIDRIVE_STATUS_11	00000800h	Bit 11, 현재 구동 드라이브 방향(보간 드라이브에서는 표시 정보 다름)
QIDRIVE_STATUS_12	00001000h	Bit 12, 펄스 출력후 서보위치 완료 신호 대기중.
QIDRIVE_STATUS_13	00002000h	Bit 13, 직선 보간 드라이브 구동중.
QIDRIVE_STATUS_14	00004000h	Bit 14, 원호 보간 드라이브 구동중.
QIDRIVE_STATUS_15	00008000h	Bit 15, 펄스 출력 중.
QIDRIVE_STATUS_16	00010000h	Bit 16, 구동 예약 데이터 개수(처음)(0-7)
QIDRIVE_STATUS_17	00020000h	Bit 17, 구동 예약 데이터 개수(중간)(0-7)
QIDRIVE_STATUS_18	00040000h	Bit 18, 구동 예약 데이터 갯수(끝)(0-7)
QIDRIVE_STATUS_19	00080000h	Bit 19, 구동 예약 Queue 비어 있음.
QIDRIVE_STATUS_20	00100000h	Bit 20, 구동 예약 Queue 가득 찮
QIDRIVE_STATUS_21	00200000h	Bit 21, 현재 구동 드라이브의 속도 모드(처음)
QIDRIVE_STATUS_22	00400000h	Bit 22, 현재 구동 드라이브의 속도 모드(끝)
QIDRIVE_STATUS_23	00800000h	Bit 23, MPG 버퍼 #1 Full
QIDRIVE_STATUS_24	01000000h	Bit 24, MPG 버퍼 #2 Full
QIDRIVE_STATUS_25	02000000h	Bit 25, MPG 버퍼 #3 Full
QIDRIVE_STATUS_26	04000000h	Bit 26, MPG 버퍼 데이터 OverFlow

Motion Command Function List

노드 및 모듈 초기화

가상축 함수

모션 파라메타 설정함수

입출력 레벨 설정 함수

모션 구동후에 상태 확인 함수

원점 검색 함수

위치구동 함수

위치 및 속도 오버라이드 구동 함수

MASTER, SLAVE 구동 함수

보간 구동 함수**(Line, Circle)**

Trigger 함수

겐트리 구동 함수

CRC(잔여 펄스 클리어) 함수

MPG 구동 함수

네트워크 상태에 대한 모션 구동 에러 처리

노드 및 모듈 초기화

본 단원에서는 라이브러리를 초기화하는데 있어 필요한 노드 및 모듈의 Load 또는 초기화와 관련된 함수들 소개한다.

노드 및 모듈 확인함수

Function	Description
AxmInfoGetAxis	해당 축의 노드번호, 모듈 위치, 모듈 아이디를 반환한다.
AxmInfoIsMotionModule	모션 모듈이 존재하는지 반환한다.
AxmInfoIsInvalidAxisNo	해당 축이 유효한지 반환한다.
AxmInfoGetAxisCount	시스템에 장착된 유효한 모션 축수를 반환한다.
AxmInfoGetFirstAxisNo	해당 노드/모듈의 첫번째 축번호를 반환한다.

AxmInfoGetAxis

Purpose

지정 축의 노드번호, 모듈위치, 모듈 아이디를 확인 할 수 있다.

Format

C

```
DWORD AxmInfoGetAxis (long lAxisNo, long * lpNodeNo, long *lpModulePos, DWORD
*upModuleID);
```

Visual Basic

```
Function AxmInfoGetAxis (ByVal lAxisNo As Long, ByRef lpNodeNo As Long, ByRef lpModulePos
As Long, ByRef upModuleID As Long) As Long
```

Delphi

```
function AxmInfoGetAxis (lAxisNo : LongInt; lpNodeNo: PLongInt; lpModulePos : PLongInt;
upModuleID : PDWord) : DWord; stdcall;
```

Input / Output

Name	in/out	Format	Init Value	Explanation
AxisNo	in	long	-	채널(축) 번호(0 ~ (최대축수 - 1))
lpNodeNo	out	long *	-	설정 축이 포함되어 있는 노드 위치값
lpModulePos	out	long*	-	설정 축이 포함되어 있는 모듈 위치값
upModuleID	out	DWORD*	-	설정 축이 포함되어 있는 모듈 아이디값

ModuleID

#define	Value	Explanation
AXT_NMC_2V04	B1h	EIP-M2Q, 2 Axis
AXT_NMC_4V04	B2h	EIP-M4Q, 4 Axis
AXT_NIO_DC16	B3h	Digital 16점 Configurable
AXT_NIO_AI4	B4h	AI 4Ch, 16 bit(reserved)

AXT_NIO_DI32	B5h	Digital IN 32점(reserved)
AXT_NIO_DO32	B6h	Digital OUT 32점(reserved)
AXT_NIO_DB32	B7h	Digital IN 16점 / OUT 16점
AXT_NIO_AO4	B8h	AO 4Ch, 16 bit (reserved)
AXT_FN_EMPTY	FEh	Empty module area
AXT_FN_UNKNOWN	FFh	Unkown module

Return AXT_RT_SUCCESS(0000) : Successful execution of API
* See error code Table more information on status error codes

Description

*참고 : 필요한 데이터만 가지고 오고싶을 때는 필요하지 않는 변수에 NULL 넣어 사용 가능하다.
노드 및 모듈 아이디 값은 제품 모델명과 동일하여 실제 노드정보와 모듈정보를 확인할 때 사용한다.

C Example

```
// 0축의 노드넘버, 모듈 위치, 모듈 아이디를 반환.
Long lpNodeNo, ModulePos;
DWORD ModuleID;
AxmInfoGetAxis (0, &lpNodeNo, &ModulePos, &ModuleID);
```

VB Example

```
'0축의 노드넘버, 모듈 위치, 모듈 아이디를 반환.
Dim lpNodeNo, ModulePos, ModuleID As Long
AxmInfoGetAxis 0, lpNodeNo, ModulePos, ModuleID
```

Delphi Example

```
{0축의 노드넘버, 모듈 위치, 모듈 아이디를 반환. }
var
ModuleID: DWord;
lpNodeNo, ModulePos: LongInt;

begin
AxmInfoGetAxis (0, @lpNodeNo, @ModulePos, @ModuleID);
end;
```

See Also

AxmInfoIsMotionModule, AxmInfoIsInvalidAxisNo, AxmInfoGetAxisCount, AxmInfoGetFirstAxisNo

AxmInfoIsMotionModule

Purpose

모션 모듈이 장착되어 있는지 여부를 반환한다.

Format

C

```
DWORD AxmInfoIsMotionModule (DWORD *upStatus);
```

Visual Basic

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

Delphi

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

Input / Output

Name	in/out	Format	Init Value	Explanation
Status	out	DWORD*	-	모션모듈의 장착 여부값

Status

#define	Value	Explanation
FALSE	00h	모션 모듈이 장착 되지 않음
TRUE	01h	모션 모듈이 장착 됨

Return AXT_RT_SUCCESS(0000) : Successful execution of API

* See error code Table more information on status error codes

Description

모션 모듈의 장착 여부는 *upStatus값으로 반환한다.

C Example

```
//  모션 모듈 장착 여부 확인
DWORD Status;
AxmInfoIsMotionModule (&Status);
```

VB Example

```
' 모션 모듈 장착 여부 확인
```

```
Dim Status As Long
AxmInfoIsMotionModule Status
```

Delphi Example

```
{ 모션 모듈 장착 여부 확인 }
var
Status: DWord;
begin
AxmInfoIsMotionModule (@Status);
end;
```

See Also

AxmInfoGetAxis, AxmInfoIsInvalidAxisNo, AxmInfoGetAxisCount, AxmInfoGetFirstAxisNo

AxmInfoIsInvalidAxisNo

Purpose

해당 축이 사용 가능한 축인지 반환한다.

Format

C

DWORD AxmInfoIsInvalidAxisNo(long lAxisNo);

Visual Basic

Function AxmInfoIsInvalidAxisNo (ByVal szInvalidAxisNo As String) As Long

Delphi

function AxmInfoIsInvalidAxisNo (lAxisNo : LongInt) : DWord; stdcall;

Input / Output

Name	in/out	Format	Init Value	Explanation
lAxisNo	in	long*	-	사용가능한지 테스트할 축번호

Return AXT_RT_SUCCESS(0000) : Successful execution of API

* See error code Table more information on status error codes

Description

해당축이 사용할 수 있는 축인지 확인한다. 사용 할 수 없다면 에러코드를 띄워 준다

C Example

```
//  9축이 사용가능한 축인지 에러코드를 확인한다.
DWORD    dwReturn;
dwReturn = AxmInfoIsInvalidAxisNo(9);
```

VB Example

```
//  9축이 사용가능한 축인지 에러코드를 확인한다.
DWORD    dwReturn;
dwReturn = AxmInfoIsInvalidAxisNo(9);
```

Delphi Example

```
//  9축이 사용가능한 축인지 에러코드를 확인한다.
DWORD    dwReturn;
dwReturn = AxmInfoIsInvalidAxisNo(9);
```

See Also

AxmInfoGetAxis, AxmInfoIsMotionModule, AxmInfoGetAxisCount, AxmInfoGetFirstAxisNo

AxmInfoGetAxisCount

Purpose

장착된 노드의 총 축의 개수를 반환한다.

Format

C

```
DWORD AxmInfoGetAxisCount (long *lpAxisCount);
```

Visual Basic

```
Function AxmInfoGetAxisCount (ByVal lAxisNo As Long) As Long
```

Delphi

```
function AxmInfoGetAxisCount (lpAxisCount : PLongInt) : DWord; stdcall;
```

Input / Output

Name	in/out	Format	Init Value	Explanation
AxisCount	out	long *	-	장착 된 노드 총 축의 개수

Return AXT_RT_SUCCESS(0000) : Successful execution of API

* See error code Table more information on status error codes

Description

장착된 노드의 연결 된 축의 개수를 반환한다.

C Example

```
//  장착된 모션 보듈 축수 확인
long   AxisCount;
AxmInfoGetAxisCount (&AxisCount);
```

VB Example

```
'장착된 모션 보듈 축수 확인
Dim AxisCount As Long
AxmInfoGetAxisCount AxisCount
```

Delphi Example

```
{ 장착된 모션 보듈 축수 확인 }
var
AxisCount: LongInt;

begin
AxmInfoGetAxisCount (@AxisCount);
end;
```

See Also

AxmInfoGetAxis, AxmInfoIsMotionModule, AxmInfoIsInvalidAxisNo, AxmInfoGetFirstAxisNo

AxmInfoGetFirstAxisNo

Purpose

해당 노드의 특정 위치 모듈에서 가장 빠른 축번호를 반환한다.

Format

C

```
DWORD AxmInfoGetFirstAxisNo (long lNodeNum, long lModulePos, long *lpAxisNo);
```

Visual Basic

```
Function AxmInfoGetFirstAxisNo (ByVal lNodeNum As Long, ByVal lModulePos As Long, ByRef
lpAxisNo As Long) As Long
```

Delphi

```
function AxmInfoGetFirstAxisNo (lNodeNum: LongInt; lModulePos : LongInt; lpAxisNo : PLongInt) :
DWord; stdcall;
```

Input / Output

Name	in/out	Format	Init Value	Explanation
lNodeNum	in	long	-	노드 ID 값
ModulePos	In	long	-	사용자가 로터리 스위치로 설정한 모듈의 위치
AxisNo	out	long *	-	해당 노드/모듈의 축번호값

Return AXT_RT_SUCCESS(0000) : Successful execution of API

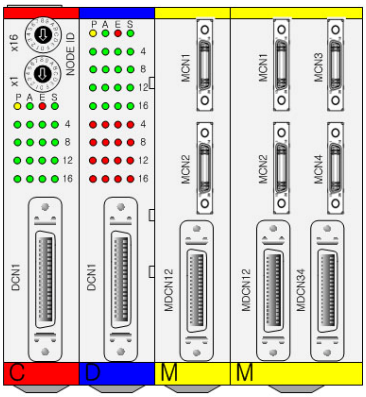
* See error code Table more information on status error codes

Description

EIP-M2Q, EIP-M4Q의 경우는 모듈타입이므로, 노드에서 가장 작은 모듈 ID(로터리 스위치로 조정)부터 시작되는 축번호가 시작 된다.

노드번호는 노드 ID(CPU 로터리 스위치) 값으로 절대 위치에 해당되며, 모듈 위치는 모션 모듈 상단의 로터리 스위치 순서의 값으로 설정된다.

*lpAxisNo값은 해당 모듈에서 빠른 축번호값이 된다.



1. 축 번호는 같은 모듈끼리 각각 카운터된다.
AXLNET은 EIP-M2/4Q모듈이상의 버전만을 지원한다

2. 같은 노드에서는 모션 모듈의 로터리 스위치 값이 작은 순서 대로 축번호가 카운터 된다.

3. 여러 개의 노드가 있을 경우 CPU 모듈 로터리 스위치가 작은 노드 부터 축 번호가 카운터 된다.

C Example

```
// Node ID 0 노드의 Module ID 1로 설정된 모듈의 가장 빠른 축번호 확인
long   AxisNo;
AxmInfoGetFirstAxisNo (0, 1, &AxisNo);
```

VB Example

```
'Node ID 0 노드의 Module ID 1로 설정된 모듈의 가장 빠른 축번호 확인
Dim AxisNo As Long
AxmInfoGetFirstAxisNo 0, 1, AxisNo
```

Delphi Example

```
{ Node ID 0 노드의 Module ID 1로 설정된 모듈의 가장 빠른 축번호 확인 }
Var AxisNo: LongInt;

begin
AxmInfoGetFirstAxisNo (0, 1, @AxisNo);
end;
```

See Also

AxmInfoGetAxis, AxmInfoIsMotionModule, AxmInfoIsInvalidAxisNo, AxmInfoGetAxisCount

가상축 함수

본단원에서는 가상축 매핑을 사용하여 라이브러리에서 구동명령을 내릴 때 실제 해당 축번호가 아닌 임의의 축번호를 사용하여 명령을 내릴 수 있도록 하기 위한 함수들을 소개한다. 예를 들어서 0번 축에 명령을 내리는데, 명령시에는 2번 축이라고 명령을 내리는 것을 말한다. 만약, 시스템이 원래 0번부터 10번까지 축이 있었는데, 축특별한 이유로 가운데 3,4번 축이 제거되었다고 하면 이럴경우 축번호는 순서대로 0번부터 8번까지 당겨져서 작동되게 된다. 그러므로 프로그램상의 축번호를 일일이 변경해야 하는 상황이 발생하는데, 이러한 경우에 프로그램 소스를 건드리지 않고, 원래의 축번호로 매핑해 줌으로서 해결할 수 있게 된다. 이 경우는 3~ 8번 축을 5~10번축으로 축번호를 변경해 주면 된다.

Function	Description.
AxmVirtualSetAxisNoMap	가상축을 설정한다.
AxmVirtualGetAxisNoMap	설정한 가상채널(축) 번호(0부터 시작함)를 반환한다.
AxmVirtualSetMultiAxisNoMap	멀티 가상축을 설정한다.
AxmVirtualGetMultiAxisNoMap	설정한 멀티 가상채널(축) 번호(0부터 시작함)를 반환한다.
AxmVirtualResetAxisMap	가상축 설정을 해지한다.

AxmVirtualSetAxisNoMap

Purpose

실제 축번호와 가상축번호를 매핑하는 함수이다.

Format

C

```
DWORD AxmVirtualSetAxisNoMap (long lRealAxisNo, long lVirtualAxisNo);
```

Visual Basic

```
Function AxmVirtualSetAxisNoMap (ByVal lRealAxisNo As Long, ByVal lVirtualAxisNo As Long ) As Long
```

Delphi

```
function AxmVirtualSetAxisNoMap (lRealAxisNo: LongInt ; lVirtualAxisNo : LongInt) : DWord; stdcall;
```

Input / Output

Name	in/out	Format	Init Value	Explanation
RealAxisNo	In	long	-	시스템에 장착된 실제 축번호 (0 ~ (시스템 장착된 축수 - 1))
VirtualAxisNo	In	long	-	매핑할 가상 축번호(0 ~ (최대축수 - 1))

Return AXT_RT_SUCCESS(0000) : Successful execution of API

* See error code Table more information on status error codes

Description

초기 상태에서 AXM 모든 함수의 축 설정은 0 ~ (실제 시스템에 장착된 축수 - 1) 범위에서 유효하지만 이 함수를 사용하면, 실제 장착된 축 번호 대신 임의의 축 번호로 바꿀 수 있다.
제어 시스템의 H/W 변경사항 발생시 기존 프로그램에 할당된 축 번호를 그대로 유지하고 실제 제어 축의 물리적인 위치를 변경하여 사용을 위해 만들어진 함수이다.

주의사항 : 여러 개의 실제 축번호에 대하여 같은 번호로 가상 축을 중복해서 맵핑하지 말아야 한다. 중복 맵핑된 경우 실제 축번호가 낮은 축만 가상 축번호로 제어 할 수 있으며, 나머지 같은 가상축 번호로 맵핑된 축은 제어가 불가능하다.

C Example

```
// 실제축 0축을 가상축 5축으로 설정
AxmVirtualSetAxisNoMap (0, 5);

// 설정 가상축 확인
long   VirtualAxisNo;
AxmVirtualGetAxisNoMap (0, &VirtualAxisNo);
```

VB Example

```
'실제축 0축을 가상축 5축으로 설정
AxmVirtualSetAxisNoMap 0, 5

'설정 가상축 확인
Dim VirtualAxisNo As Long
AxmVirtualGetAxisNoMap 0, VirtualAxisNo
```

Delphi Example

```
var
VirtualAxisNo: LongInt;

Begin
{ 실제축 0축을 가상축 5축으로 설정 }
AxmVirtualSetAxisNoMap (0, 5);
{ 설정 가상축 확인 }
AxmVirtualGetAxisNoMap (0, @VirtualAxisNo);
end;
```

See Also

AxmVirtualGetAxisNoMap, AxmVirtualSetMultiAxisNoMap, AxmVirtualGetMultiAxisNoMap, AxmVirtualResetAxisMap

AxmVirtualGetAxisNoMap

Purpose

실제 축번호와 매핑된 가상축번호를 확인하는 함수이다.

Format

C

DWORD AxmVirtualGetAxisNoMap (long lRealAxisNo, long *lpVirtualAxisNo);

Visual Basic

Function AxmVirtualGetAxisNoMap (ByVal lAxisNo As Long, ByRef lVirtualAxisNo As Long) As Long

Delphi

function AxmVirtualGetAxisNoMap (lRealAxisNo : LongInt; lpVirtualAxisNo : PLongInt) : DWord; stdcall;

Input / Output

Name	in/out	Format	Init Value	Explanation
RealAxisNo	In	long	-	시스템에 장착된 실제 축번호 (0 ~ (시스템 장착된 축수 - 1))
VirtualAxisNo	out	long *	-	매핑할 가상 축번호(0 ~ (최대축수 - 1))

Return AXT_RT_SUCCESS(0000) : Successful execution of API

* See error code Table more information on status error codes

Description

AxmVirtualSetAxisNoMap 함수로 설정된 실제 축번호에 대한 가상축번호를 확인하는 함수이다. 맵핑이 안되어있을 경우에는 -1을 반환하고 맵핑이 되어있을경우에 해당축이 반환된다.

C Example

```
// 실제축 0축을 가상축 5축으로 설정
AxmVirtualSetAxisNoMap (0, 5);

// 설정 가상축 확인
long   VirtualAxisNo;
AxmVirtualGetAxisNoMap (0, &VirtualAxisNo);
```

VB Example

```
'실제축 0축을 가상축 5축으로 설정
AxmVirtualSetAxisNoMap 0, 5

'설정 가상축 확인
Dim VirtualAxisNo As Long
AxmVirtualGetAxisNoMap 0, VirtualAxisNo
```

Delphi Example

```
var
VirtualAxisNo: LongInt;

Begin
{ 실제축 0축을 가상축 5축으로 설정 }
AxmVirtualSetAxisNoMap (0, 5);
{ 설정 가상축 확인 }
AxmVirtualGetAxisNoMap (0, @VirtualAxisNo);
end;
```

See Also

AxmVirtualSetAxisNoMap, AxmVirtualSetMultiAxisNoMap, AxmVirtualGetMultiAxisNoMap, AxmVirtualResetAxisMap

AxmVirtualSetMultiAxisNoMap

Purpose

한 축 이상의 실제 축 번호와 가상 축 버호를 맵핑하는 함수이다.

Format

C

```
DWORD AxmVirtualSetMultiAxisNoMap (long lSize, long *lpRealAxisNo, long
*lpVirtualAxisNo);
```

Visual Basic

```
Function AxmVirtualSetMultiAxisNoMap (ByVal lSize As Long, ByRef lpRealAxesNo As Long, ByRef
lpVirtualAxesNo As Long) As Long
```

Delphi

```
function AxmVirtualSetMultiAxisNoMap (lSize : LongInt; lpRealAxesNo : PLongInt; lpVirtualAxesNo :
PLongInt) : DWord; stdcall;
```

Input / Output

Name	in/out	Format	Init Value	Explanation
Size	in	long	-	설정할 축 수(1 ~ (최대축수 - 1))
RealAxisNo	In	long *	-	시스템에 장착된 실제 축번호 배열
VirtuallAxisNo	In	long *	-	매핑할 가상 축번호 배열

Return AXT_RT_SUCCESS(0000) : Successful execution of API

* See error code Table more information on status error codes

Description

실제 축 번호와 가상 축 번호를 다축에 대하여 배열로 맵핑하는 함수이다..

초기 상태에서 AXM 모든 함수의 축번호 설정은 0 ~ (실제 시스템에 장착된 축수 - 1) 범위에서 유효하지만 이 함수를 사용하여 실제 장착된 축번호 대신 임의의 축번호로 바꿀 수 있다.
제어 시스템의 H/W 변경사항 발생시 기존 프로그램에 할당된 축번호를 그대로 유지하고 실제 제어 축의 물리적인 위치를 변경하여 사용을 위해 만들어진 함수이다

주의사항 : 여러 개의 실제 축번호에 대하여 같은 번호로 가상 축을 중복해서 맵핑하지 말아야 한다.
중복 맵핑된 경우 실제 축번호가 낮은 축만 가상 축번호로 제어 할 수 있으며, 나머지 같은 가상축 번

호로 맵핑된 축은 제어가 불가능하다

C Example

```
// 가상축 설정
long Realaxis[4],Virtualaxis[4];
Realaxis[0] = 0;
Realaxis[1] = 1;
Realaxis[2] = 2;
Realaxis[3] = 3;

Virtualaxis[0] = 4;
Virtualaxis[1] = 5;
Virtualaxis[2] = 6;
Virtualaxis[3] = 7;
// 0축을 4축으로 , 1축을 5축으로, 2축을 6축으로 3축을 7축으로 설정한다.
AxmVirtualSetMultiAxisNoMap(4,Realaxis,Virtualaxis);

// 가상축 설정 확인
long lReadRealAxisNo[4];
AxmVirtualGetMultiAxisNoMap (4, Realaxis, lReadRealAxisNo);
```

VB Example

```
‘ 가상축 설정
Dim Realaxis(0 To 3) As Long
Dim Virtualaxis(0 To 3) As Long
Realaxis(0) = 0
Realaxis(1) = 1
Realaxis(2) = 2
Realaxis(3) = 3
Virtualaxis(0) = 4
Virtualaxis(1) = 5
Virtualaxis(2) = 6
Virtualaxis(3) = 7

AxmVirtualSetMultiAxisNoMap 4, Realaxis(0), Virtualaxis(0)

‘ 가상축 설정 확인
Dim RealAxisNo (0 To 3) As Long
AxmVirtualGetMultiAxisNoMap 4, VirtualAxisNo(0), RealAxisNo(0)
```

Delphi Example

```
var
Realaxis: array [0..3] of Longint;
Virtualaxis: array [0..3] of Longint;
RealAxisNo: array [0..3] of Longint;

Begin
Realaxis[0]           := 0;
```

```
Realaxis[1]           := 1;
Realaxis[2]           := 2;
Realaxis[3]           := 3;

Virtualaxis[0] := 4;
Virtualaxis[1] := 5;
Virtualaxis[2] := 6;
Virtualaxis[3] := 7;

{ 가상축 설정 }
AxmVirtualSetMultiAxisNoMap (4, @Realaxis, @Virtualaxis);
{ 가상축 설정 확인 }

AxmVirtualGetMultiAxisNoMap (4, @Realaxis, @RealAxisNo);
end;
```

See Also

AxmVirtualSetAxisNoMap, AxmVirtualGetAxisNoMap, AxmVirtualGetMultiAxisNoMap, AxmVirtualResetAxisMap

AxmVirtualGetMultiAxisNoMap

Purpose

한 축이상 실제 축번호와 매핑된 가상축번호를 확인하는 함수이다.

Format

C

DWORD AxmVirtualGetMultiAxisNoMap (long lSize, long *lpRealAxisNo, long *lpVirtualAxisNo);

Visual Basic

Function AxmVirtualGetMultiAxisNoMap ByVal lSize As Long, ByRef lpRealAxesNo As Long, ByRef lpVirtualAxesNo As Long) As Long

Delphi

function AxmVirtualGetMultiAxisNoMap (lSize : LongInt; lpRealAxesNo : PLongInt; lpVirtualAxesNo : PLongInt) : DWord; : Byte; stdcall;

Input / Output

Name	in/out	Format	Init Value	Explanation
Size	in	long		설정한 축 수
RealAxisNo	In	long *		시스템에 장착된 실제 축번호
VirtuallAxisNo	Out	long *		매핑한 가상 축번호

Return AXT_RT_SUCCESS(0000) : Successful execution of API

* See error code Table more information on status error codes

Description

AxmVirtualSetMultiAxisNoMap 를 사용해서 실제 매핑된 다수의 가상축번호를 확인한다.

C Example

```
// 가상축 설정
long Realaxis[4],Virtualaxis[4];
Realaxis[0] = 0;
Realaxis[1] = 1;
Realaxis[2] = 2;
Realaxis[3] = 3;
```

```
Virtualaxis[0] = 4;
Virtualaxis[1] = 5;
Virtualaxis[2] = 6;
Virtualaxis[3] = 7;
// 0축을 4축으로 , 1축을 5축으로, 2축을 6축으로 3축을 7축으로 설정한다.
AxmVirtualSetMultiAxisNoMap(4,Realaxis,Virtualaxis);

// 가상축 설정 확인
long lReadRealAxisNo[4];
AxmVirtualGetMultiAxisNoMap (4, Realaxis, lReadRealAxisNo);
```

VB Example

```
' 가상축 설정
Dim Realaxis(0 To 3) As Long
Dim Virtualaxis(0 To 3) As Long

Realaxis(0) = 0
Realaxis(1) = 1
Realaxis(2) = 2
Realaxis(3) = 3

Virtualaxis(0) = 4
Virtualaxis(1) = 5
Virtualaxis(2) = 6
Virtualaxis(3) = 7

AxmVirtualSetMultiAxisNoMap 4, Realaxis(0), Virtualaxis(0)

' 가상축 설정 확인
Dim RealAxisNo (0 To 3) As Long
AxmVirtualGetMultiAxisNoMap 4, Realaxis(0), RealAxisNo(0)
```

Delphi Example

```
var
Realaxis: array [0..3] of Longint;
Virtualaxis: array [0..3] of Longint;
RealAxisNo: array [0..3] of Longint;

Begin
Realaxis[0]           := 0;
Realaxis[1]           := 1;
Realaxis[2]           := 2;
Realaxis[3]           := 3;

Virtualaxis[0] := 4;
Virtualaxis[1] := 5;
Virtualaxis[2] := 6;
Virtualaxis[3] := 7;

{ 가상축 설정 }
AxmVirtualSetMultiAxisNoMap (4, @Realaxis, @Virtualaxis);
```

```
{ 가상축 설정 확인 }
AxmVirtualGetMultiAxisNoMap (4, @Realaxis, @RealAxisNo);
end;
```

See Also

AxmVirtualSetAxisNoMap, AxmVirtualGetAxisNoMap, AxmVirtualSetMultiAxisNoMap, AxmVirtualResetAxisMap

AxmVirtualResetAxisMap

Purpose

현재 설정되어 있는 가상축 설정을 해지한다.

Format

C

DWORD AxmVirtualResetAxisMap ();

Visual Basic

Function AxmVirtualResetAxisMap () As Long

Delphi

function AxmVirtualResetAxisMap () : DWord; stdcall;

Input/Output

None

Return AXT_RT_SUCCESS(0000) : Successful execution of API

* See error code Table more information on status error codes

Description

현재 설정되어 있는 가상 맵핑 내용을 초기화 하여 함수 내에서 사용하는 축번호들은 실제 시스템에 장착된 축번호와 같아 진다.

C Example

```
// 가상축 설정 해지
AxmVirtualResetAxisMap ();
```

VB Example

```
' 가상축 설정 해지
AxmVirtualResetAxisMap
```

Delphi Example

```
{ 가상축 설정 해지 }
begin
AxmVirtualResetAxisMap ();
End;
```

See Also

AxmVirtualSetAxisNoMap, AxmVirtualGetAxisNoMap, AxmVirtualSetMultiAxisNoMap, AxmVirtualGetMultiAxisNoMap

모션 파라메타 설정함수

단원에서 소개 되어지는 함수들은 모션에 관련된 모든 파라메터 설정에 대해 설명한다. 모션을 구동하기 전에 모션 파라메타 함수들을 자세히 읽어보고 구동하기 바란다.

Function	Description
AxmMotLoadParaAll	AxmMotSaveParaAll로 저장 되어진 33가지파라메타를 파일을 불러온다. 사용자가 임의로 불러서 변경가능하다.
AxmMotSaveParaAll	현재 모든축에 대한 모든 33가지 파라메타를 축별로 저장한다. .mot파일로 저장한다. AxmMotLoadParaAll 사용해서 파일을 불러온다.
AxmMotSetParaLoad	설정되어있는 정보를 파일로 저장 하기 위해서는 AxmMotSaveParaAll 함수를 사용한다. AxmMotSaveParaAll 함수는 모든 축의 정보를 파일로 저장하는 함수이다. 여기 파라메타중 28 – 33 까지 파라메타 dInitPos, dInitVel, dInitAccel, dInitDecel, uInitAbsRelMode, uInitProfileMode 6개 파라메타만 설정한다. 나머지 파라메타는 라이브러리를 닫을 때 현재상태를 읽어서 AxmMotSaveParaAll 함수를 이용하여 저장시킨다.
AxmMotGetParaLoad	설정되어있는 정보를 파일로 저장 하기 위해서는 AxmMotSaveParaAll 함수를 사용한다. AxmMotSaveParaAll 함수는 모든 축의 정보를 파일로 저장한 파라메타중에 28 – 33 까지 파라메타 dInitPos, dInitVel, dInitAccel, dInitDecel, uInitAbsRelMode, uInitProfileMode 6개 파라메타만 불러 올 때 사용하는 함수이다.
AxmMotSetPulseOutMethod	지정 축의 펄스 출력 방식을 설정한다.
AxmMotGetPulseOutMethod	지정 축의 펄스 출력 방식 설정을 반환한다,
AxmMotSetEncInputMethod	지정 축의 외부(Act) 카운트의 증가 방향 설정을 포함하여 지정 축의 Encoder 입력 방식을 설정한다.
AxmMotGetEncInputMethod	지정 축의 외부(Act) 카운트의 증가 방향 설정을 포함하여 지정

	축의 Encoder 입력 방식을 반환한다.
AxmMotSetMoveUnitPerPulse	지정 축의 펄스 당 움직이는 거리를 설정한다.
AxmMotGetMoveUnitPerPulse	지정 축의 펄스 당 움직이는 거리를 반환한다.
AxmMotSetDecelPos	지정 축에 감속 시작 포인트 검출 방식을 설정한다.
AxmMotGetDecelPos	지정 축의 감속 시작 포인트 검출 방식을 반환한다
AxmMotSetRemainPulse	지정 축에 수동 감속 모드에서 잔량 펄스를 설정한다.
AxmMotGetRemainPulse	지정 축의 수동 감속 모드에서 잔량 펄스를 반환한다.
AxmMotSetMaxVel	지정 축에 등속도 구동 함수에서의 최고 속도를 설정한다.
AxmMotGetMaxVel	지정 축의 등속도 구동 함수에서의 최고 속도를 반환한다.
AxmMotSetAbsRelMode	지정 축의 이동 거리 계산 모드를 설정한다. 절대위치 모드(0)로 설정한 경우 이동 함수에 사용하는 위치값은 절대위치값이 된다. 예를 들어 현재 위치값이 '10'일 때 절대위치 모드에서 '100' 위치로 이동할 경우 현재 위치에서 '+90'만큼 이동하여 '100'위치에서 정지하게 된다. 상대위치 모드(1)로 설정한 경우는 이동 함수에 사용하는 위치값은 현재 위치에서 이동할 이동량이 된다. 예를 들어 현재 위치값이 '10일 대 상대 위치 모드에서 '100' 만큼 이동할 경우, 현재 위치에서 '+100'만큼 이동하여 '110'의 위치에서 정지하게 된다.
AxmMotGetAbsRelMode	지정 축의 설정된 이동 거리 계산 모드를 반환한다
AxmMotSetProfileMode	지정 축의 구동 속도 프로파일 모드를 설정한다. 속도 프로파일 모드에는 대칭 또는 비대칭Trapezoide / S-Curve 모드, Quasi-S 모드가 있다.
AxmMotGetProfileMode	지정 축의 설정한 구동 속도 프로파일 모드를 반환한다.
AxmMotSetAccelUnit	지정 축의 가속도(가속도의 단위는 Unit/pulse를 1/1로 한경우에 PPS[Pulses/sec]) 단위를 설정한다. 가속도(가속도의 단위는 Unit/pulse를 1/1로 한경우에 PPS[Pulses/sec]) 단위는 u/s^2, sec, rev/s^2 중 선택할 수 있다.

AxmMotGetAccelUnit	지정 축의 설정된 가속도(가속도의 단위는 Unit/pulse를 1/1로 한 경우에 PPS[Pulses/sec]) 단위를 반환한다.
AxmMotSetMinVel	지정 축에 초기 속도를 설정한다.
AxmMotGetMinVel	지정 축의 초기 속도를 반환한다.
AxmMotSetAccelJerk	지정 축의 가속 저크값을 설정한다.
AxmMotGetAccelJerk	지정 축의 설정된 가속 저크값을 반환한다.
AxmMotSetDecelJerk	지정 축의 감속 저크값을 설정한다.
AxmMotGetDecelJerk	지정 축의 설정된 감속 저크값을 반환한다.

AxmMotLoadParaAll

Purpose

모든 축의 초기 값을 지정한 파일에서 읽어 설정한다. AxmMotSaveParaAll로 저장 되어진 33가지 파라메타를 파일을 불러온다. 사용자가 임의로 편집 가능하다.

Format

C

```
DWORD AxmMotLoadParaAll (char *szFilePath);
```

Visual Basic

```
Function AxmMotLoadParaAll (ByRef szFilePath As String) As Long
```

Delphi

```
function AxmMotLoadParaAll (szFilePath: PChar) : DWord ; stdcall;
```

Input / Output

Name	in/out	Format	Init Value	Explanation
szFilePath	in	char *	–	불러올 파일 명

Return AXT_RT_SUCCESS(0000) : Successful execution of API
* See error code Table more information on status error codes

Description

모든 축의 초기 값을 지정한 *spFilePath 파일에서 읽어 설정한다. 로드되는 파라메타는 AxmMotSaveParaAll의 표를 참고한다. 초기에는 .MOT 설정파일이 없기 때문에 기본설정값으로 설정된다. 직접설정하기위해서는 AxmMotSaveParaAll함수를 이용하여 .MOT파일을 저장시켜 AxmMotLoadParaAll 로 불러온다.

파일 설정되는 내부변수 초기값

No	신호관련 변수	파라메타 설명	설정 값
00	AXIS_NO	축 번호	설정 축번호
01	PULSE_OUT_METHOD	모션노드에서 서보팩 지령으로 사용될 펄스의 출력방식을 설정	TwoCwCcwHigh(6)
02	Enc[Encoder]	Encoder신호의 입력방식을 설정	ObverseSqr4M

			ode (3)
03	InP[InPos]	서보팩 위치결정 완료신호의 사용여부 및 Active Level을 설정	2
04	Alm[Alarm]	서보팩 알람 신호의 사용여부 및 Active level을 설정	1
05	NLimit	기구 (-)리미트 센서의 사용여부 및 Active level을 설정	1
06	PLimit	기구 (+)리미트 센서의 사용여부 및 Active level을 설정	1
07	MinSpd	모터 구동 시 초기 시작구동 속도 설정	1.0
08	MaxSpd	모터 구동 시 최대구동 속도 설정	700000.0
09	HmSig	원점센서로 사용할 신호를 설정	4
10	HmLev	원점센서의 Active Level을 설정	1
11	HmDir	원점센서 검출시 초기 검색진행 방향을 설정	0
12	Zphas_Lev	엔코더 Z상 Active level	1
13	Zphas_Use	원점센서 검출 후 엔코더 Z상을 검출 여부 설정	0
14	Stop Signal Mode	ESTOP, SSTOP 사용시 모드	0
15	Stop Signal level	ESTOP, SSTOP 사용 레벨	0
16	VelFirst	원점검색 시 초기 고속검출 속도(속도의 단위는 Unit/pulse를 1/1로 한경우에 PPS[Pulses/sec])	10000.0
17	VelSecond	원점검색 시 1차 센서 검출 후 반대 방향으로 빠져나오는 속도	10000.0
18	VelThird	원점검색 시 1차 센서 검출 후 재 검색 진행하는 속도	2000.0
19	VelLast	원점검색 시 최종 검출 속도 설정 [원점검색의 정밀도 결정]	100.0
20	AccFirst	원점검색 시 초기 고속검출 가속도(가속도의 단위는 Unit/pulse를 1/1로 한경우에 PPS[Pulses/sec^2])	40000.0
21	AccSecond	원점검색 시 1차 센서 검출 후 반대 방향으로 빠져나오는 가속도(가속도의 단위는 Unit/pulse를 1/1로 한경우에 PPS[Pulses/sec^2])	40000.0
22	HClrTim	원점검색 완료 후 Cmd, Act위치를 Clear전 대기시간	1000.0
23	HOffset	원점검색 완료 후 기구원점 재 설정 시 이동 Offset 값	0.0

24	NSWLimit	원점검색 완료 후 적용할 (-) Software Limit 값 설정	0.0
25	PSWLimit	원점검색 완료 후 적용할 (+) Software Limit 값 설정	0.0
26	OenPulse	모터 1회전에 필요한 펄스 수 설정 [서보팩 전자 기어 비 참조]	1.0
27	OneUnit	모터 1회전 시 이동되는 이동 량 [mm, 도, 등등]	1
28	InitPosition	초기화시 위치	200
29	InitVel	초기화시 속도(속도의 단위는 Unit/pulse를 1/1로 한경우에 PPS[Pulses/sec])	200
30	InitAccel	초기화시 가속도(가속도의 단위는 Unit/pulse를 1/1로 한경우에 PPS[Pulses/sec^2])	400
31	InitDecel	초기화시 감속도(감속도의 단위는 Unit/pulse를 1/1로 한경우에 PPS[Pulses/sec^2])	400
32	AbsReMode	초기화시 절대/상대 모드	0
33	ProfMode	초기화시 속도 프로파일 모드	4

C Example

```
char szFileName[50];

strcpy(szFileName, “C:\\Data\\CamCIP.mot”);

AxmMotLoadParaAll(szFileName); // 모든 축의 초기 값을 파일에 읽어 설정한다.
```

VB Example

```
Dim szFileName As String

szFileName = “C:\\Data\\CamCIP.mot”

AxmMotLoadParaAll szFileName  ‘ 모든 축의 초기 값을 파일에 읽어 설정한다.
```

Delphi Example

```
Var
szFileName : String;

begin
szFileName := ‘C:\\Data\\CamCIP.mot’;

AxmMotLoadParaAll(@szFileName);{ 모든 축의 초기 값을 파일에 읽어 설정한다. }
```

```
end;
```

See Also

AxmMotSaveParaAll, AxmMotSetParaLoad, AxmMotGetParaLoad
AxmMotSetPulseOutMethod, AxmMotSetEncInputMethod, AxmMotGetEncInputMethod,
AxmMotSetMoveUnitPerPulse, AxmMotGetMoveUnitPerPulse, AxmMotSetDecelPos,
AxmMotGetDecelPos AxmMotSetRemainPulse, AxmMotGetRemainPulse

AxmMotSaveParaAll

Purpose

현재 모든 축에 대한 파라메타를 축 별로 .mot파일에 저장한다. AxmMotLoadParaAll을 사용해서 파일을 불러온다.

Format

C

```
DWORD AxmMotSaveParaAll (char * szFilePath);
```

Visual Basic

```
Function AxmMotSaveParaAll (ByRef szFilePath As String) As Long
```

Delphi

```
function AxmMotSaveParaAll (szFilePath: PChar) : DWord ; stdcall;
```

Input / Output

Name	in/out	Format	Init Value	Explanation
szFilePath	in	char *	–	저장할 파일 명

Return AXT_RT_SUCCESS(0000) : Successful execution of API
* See error code Table more information on status error codes

Description

모든 축의 초기 값을 *szFilePath 파일에 저장한다. 로드되는 파라메타는 AxmMotSaveParaAll의 표를 참고한다. 초기에는 .MOT 설정파일이 없다. 기본설정값이 내부에 설정된다. 직접설정하기위해서는 AxmMotSaveParaAll함수로 이용하여 .MOT파일을 저장시켜 AxmMotLoadParaAll 로 불러온다.

파일 설정되는 내부변수 초기값

No	신호관련 변수	파라메타 설명	설정 값
00	AXIS_NO	축 번호	설정 축번호
01	PULSE_OUT_METHOD	모션노드에서 서보팩 지령으로 사용될 펄스의 출력방식을 설정	TwoCwCcwHigh(6)
02	Enc[Encoder]	Encoder 신호의 입력방식을 설정	ObverseSqr4Mode (3)

03	InP[InPos]	서보팩 위치결정 완료신호의 사용여부 및 Active Level을 설정	2
04	Alm[Alarm]	서보팩 알람 신호의 사용여부 및 Active level을 설정	1
05	NLimit	기구 (-)리미트 센서의 사용여부 및 Active level을 설정	1
06	PLimit	기구 (+)리미트 센서의 사용여부 및 Active level을 설정	1
07	MinSpd	모터 구동 시 초기 시작구동 속도 설정	1.0
08	MaxSpd	모터 구동 시 최대구동 속도 설정	700000.0
09	HmSig	원점센서로 사용할 신호를 설정	4
10	HmLev	원점센서의 Active Level을 설정	1
11	HmDir	원점센서 검출시 초기 검색진행 방향을	0
12	Zphas_Lev	엔코더 Z상 Active level	1
13	Zphas_Use	원점센서 검출 후 엔코더 Z상을 검출 여부 설정	0
14	Stop Signal Mode	ESTOP, SSTOP 사용시 모드	0
15	Stop Signal level	ESTOP, SSTOP 사용 레벨	0
16	VelFirst	원점검색 시 초기 고속검출 속도(속도의 단위는 Unit/pulse를 1/1로 한경우에 PPS[Pulses/sec])	10000.0
17	VelSecond	원점검색 시 1차 센서 검출 후 반대 방향으로 빠져나오는 속도	10000.0
18	VelThird	원점검색 시 1차 센서 검출 후 재 검색 진행하는 속도	2000.0
19	VelLast	원점검색 시 최종 검출 속도 설정 [원점검색의 정밀도 결정]	100.0
20	AccFirst	원점검색 시 초기 고속검출 가속도(가속도의 단위는 Unit/pulse를 1/1로 한경우에 PPS[Pulses/sec^2])	40000.0
21	AccSecond	원점검색 시 1차 센서 검출 후 반대 방향으로 빠져나오는 가속도(가속도의 단위는 Unit/pulse를 1/1로 한경우에 PPS[Pulses/sec^2])	40000.0
22	HClrTim	원점검색 완료 후 Cmd, Act위치를 Clear전 대기시간	1000.0

23	HOffset	원점검색 완료 후 기구원점 재 설정 시 이동 Offset 값	0.0
24	NSWLimit	원점검색 완료 후 적용할 (-) Software Limit 값 설정	0.0
25	PSWLimit	원점검색 완료 후 적용할 (+) Software Limit 값 설정	0.0
26	OenPulse	모터 1회전에 필요한 펄스 수 설정 [서보팩 전자 기어 비 참조]	1.0
27	OneUnit	모터 1회전 시 이동되는 이동 량 [mm, 도, 등등]	1
28	InitPosition	초기화시 위치	200
29	InitVel	초기화시 속도(속도의 단위는 Unit/pulse를 1/1로 한경우에 PPS[Pulses/sec])	200
30	InitAccel	초기화시 가속도(가속도의 단위는 Unit/pulse를 1/1로 한경우에 PPS[Pulses/sec^2])	400
31	InitDecel	초기화시 감속도(감속도의 단위는 Unit/pulse를 1/1로 한경우에 PPS[Pulses/sec^2])	400
32	AbsReMode	초기화시 절대/상대 모드	0
33	ProfMode	초기화시 속도 프로파일 모드	4

C Example

```
char szFileName[50];
strcpy(szFileName, “C:\\Data\\CamCIP.mot”);
AxmMotSaveParaAll(szFileName);   // 모든 축의 초기 값을 파일에 저장 한다.
```

VB Example

```
Dim szFileName As String
szFileName = “C:\\Data\\CamCIP.mot”
AxmMotSaveParaAll szFileName  ‘ 모든 축의 초기 값을 파일에 저장 한다.
```

Delphi Example

```
Var
szFileName : String;

begin
szFileName := 'C:\\Data\\CamCIP.mot';

AxmMotSaveParaAll(@szFileName);    { 모든 축의 초기 값을 파일에 저장 한다. }
end;
```

See Also

AxmMotLoadParaAll , AxmMotSetParaLoad , AxmMotGetParaLoad
AxmMotSetPulseOutMethod, AxmMotSetEncInputMethod, AxmMotGetEncInputMethod,
AxmMotSetMoveUnitPerPulse, AxmMotGetMoveUnitPerPulse, AxmMotSetDecelPos,
AxmMotGetDecelPos, AxmMotSetRemainPulse, AxmMotGetRemainPulse,

AxmMotSetParaLoad

Purpose

AxmMotSaveParaAll로 저장되는 파라메타중 28 – 31 까지 파라메타 dInitPos, dInitVel, dInitAccel, dInitDecel 를 설정한다.

Format

C

```
DWORD AxmMotSetParaLoad (long lAxisNo, double dInitPos, double dInitVel, double dInitAccel,
double dInitDecel);
```

Visual Basic

```
Function AxmMotSetParaLoad (ByVal lAxisNo As Long, ByVal dInitPos As Double, ByVal dInitVel As
Double, ByVal dInitAccel As Double, ByVal dInitDecel As Double) As Long
```

Delphi

```
function AxmMotSetParaLoad (lAxisNo : LongInt; dInitPos : Double; dInitVel : Double; dInitAccel :
Double; dInitDecel : Double) : DWord; stdcall;
```

Input / Output

Name	in/out	Format	Init Value	Explanation
AxisNo	In	long	–	채널(축) 번호(0 ~ (최대축수 – 1))
InitPos	in	double	1000	초기화위치 (사용자가 라이브러리 Open과 동시에 이값을 가져온다.)
InitVel	in	double	200	초기화속도 (사용자가 라이브러리 Open과 동시에 이값을 가져온다.)
InitAccel	In	double	400	초기화가속도(사용자가 라이브러리 Open 과 동시에 이값을 가져온다.)
InitDecel	In	double	400	초기화감속도 (사용자가 라이브러리 Open 과 동시에 이값을 가져온다.)

Return AXT_RT_SUCCESS(0000) : Successful execution of API.

* See error code Table more information on status error codes

Description

파일로 저장되는 인자중 라이브러리 내부에서만 사용하는 파라미터(InitPos, InitVel, InitAccel,

InitDecel)을 AxmMotSetParaLoad 로 설정한 다음 AxmMotSaveParaAll함수를 이용하여 저장시킨다. 만약 설정을 안한다면 초기 설정 값이 저장된다. 해당 파라미터는 모션 구동에 전혀 영향을 주지 않으며, 프로그램 작성시 축마다의 사용자 설정값 저장의 용도로 활용된다.

파일 설정되는 내부변수 초기값

No	신호관련 변수	파라메타 설명	설정 값
00	AXIS_NO	축 번호	설정 축번호
01	PULSE_OUT_METHOD	모션보드에서 서보팩 지령으로 사용될 펄스의 출력방식을 설정	TwoCwCcwHigh(6)
02	Enc[Encoder]	Encoder신호의 입력방식을 설정	ObverseSqr4Mode (3)
03	InP[InPos]	서보팩 위치결정 완료신호의 사용여부 및 Active Level을 설정	2
04	Alm[Alarm]	서보팩 알람 신호의 사용여부 및 Active level을 설정	1
05	NLimit	기구 (-)리미트 센서의 사용여부 및 Active level을 설정	1
06	PLimit	기구 (+)리미트 센서의 사용여부 및 Active level을 설정	1
07	MinSpd	모터 구동 시 초기 시작구동 속도 설정	1.0
08	MaxSpd	모터 구동 시 최대구동 속도 설정	700000.0
09	HmSig	원점센서로 사용할 신호를 설정	4
10	HmLev	원점센서의 Active Level을 설정	1
11	HmDir	원점센서 검출시 초기 검색진행 방향을	0
12	Zphas_Lev	엔코더 Z상 Active level	1
13	Zphas_Use	원점센서 검출 후 엔코더 Z상을 검출 여부 설정	0
14	Stop Signal Mode	ESTOP, SSTOP 사용시 모드	0
15	Stop Signal level	ESTOP, SSTOP 사용 레벨	0
16	VelFirst	원점검색 시 초기 고속검출 속도(속도의 단위는 Unit/pulse를 1/1로 한경우에 PPS[Pulses/sec])	10000.0

17	VelSecond	원점검색 시 1차 센서 검출 후 반대 방향으로 빠져나오는 속도	10000.0
18	VelThird	원점검색 시 1차 센서 검출 후 재 검색 진행하는 속도	2000.0
19	VelLast	원점검색 시 최종 검출 속도 설정 [원점검색의 정밀도 결정]	100.0
20	AccFirst	원점검색 시 초기 고속검출 가속도(가속도의 단위는 Unit/pulse를 1/1로 한경우에 PPS[Pulses/sec^2])	40000.0
21	AccSecond	원점검색 시 1차 센서 검출 후 반대 방향으로 빠져나오는 가속도(가속도의 단위는 Unit/pulse를 1/1로 한경우에 PPS[Pulses/sec^2])	40000.0
22	HClrTim	원점검색 완료 후 Cmd, Act위치를 Clear전 대기시간	1000.0
23	HOffset	원점검색 완료 후 기구원점 재 설정 시 이동 Offset 값	0.0
24	NSWLimit	원점검색 완료 후 적용할 (-) Software Limit 값 설정	0.0
25	PSWLimit	원점검색 완료 후 적용할 (+) Software Limit 값 설정	0.0
26	OenPulse	모터 1회전에 필요한 펄스 수 설정 [서보팩 전자 기어비 참조]	1.0
27	OneUnit	모터 1회전 시 이동되는 이동 량 [mm, 도, 등등]	1
28	InitPosition	초기화시 위치	200
29	InitVel	초기화시 속도(속도의 단위는 Unit/pulse를 1/1로 한경우에 PPS[Pulses/sec])	200
30	InitAccel	초기화시 가속도(가속도의 단위는 Unit/pulse를 1/1로 한경우에 PPS[Pulses/sec^2])	400
31	InitDecel	초기화시 감속도(감속도의 단위는 Unit/pulse를 1/1로 한경우에 PPS[Pulses/sec^2])	400
32	AbsReMode	초기화시 절대/상대 모드	0
33	ProfMode	초기화시 속도 프로파일 모드	4

28 – 33 까지 파라메타를 강제적으로 설정한다.

C Example

```
double dInitpos = 1000, dInitvel = 200,  dInitaccel = 400, dInitdecel =
       400;
AxmMotSetParaLoad(0, dInitpos, dInitvel, dInitaccel, dInitdecel);

double dReadInitpos,dReadInitvel,dReadInitaccel,dReadInitdecel;
AxmMotGetParaLoad(0,&dReadInitpos,&dReadInitvel, &dReadInitaccel
, &dReadInitdecel);
```

VB Example

```
Dim dInitpos ,dInitvel,dInitaccel,dInitdecel As Double
dInitpos=1000 :dInitvel = 200 :dInitaccel = 400: dInitdecel = 400
AxmMotSetParaLoad 0, dInitpos, dInitvel, dInitaccel, dInitdecel

Dim dReadInitpos,dReadInitvel,dReadInitaccel,dReadInitdecel As Double
AxmMotGetParaLoad 0, dReadInitpos, dReadInitvel, dReadInitaccel,
 dReadInitdecel
```

Delphi Example

```
var
dInitpos,dInitvel,dInitaccel,dInitdecel : Double;
dReadInitpos,dReadInitvel,dReadInitaccel,dReadInitdecel : Double;
begin

dInitpos := 1000; dInitvel := 200; dInitaccel := 400; dInitdecel := 400;
AxmMotSetParaLoad(0, dInitpos, dInitvel, dInitaccel, dInitdecel);

AxmMotGetParaLoad(0, @dReadInitpos,@dReadInitvel, @dReadInitaccel,
@dReadInitdecel);

End;
```

See Also

AxmMotLoadParaAll, AxmMotGetParaLoad, AxmMotSetPulseOutMethod, AxmMotSetEncInputMethod, AxmMotGetEncInputMethod, AxmMotSetMoveUnitPerPulse, AxmMotGetMoveUnitPerPulse, AxmMotSetDecelMode, AxmMotGetDecelMode, AxmMotSetSRate, AxmMotGetSRate, AxmMotSetRemainPulse, AxmMotGetRemainPulse,

AxmMotGetParaLoad

Purpose

AxmMotSaveParaAll로 저장한 파라메타중 28 – 33 까지 파라메타 dInitPos, dInitVel, dInitAccel, dInitDecel를 확인하는 함수이다.

Format

C

```
DWORD AxmMotGetParaLoad(long lAxisNo, double *dpInitPos, double *dpInitVel, double
*dpInitAccel, double *dpInitDecel);
```

Visual Basic

```
Function AxmMotGetParaLoad (ByVal lAxisNo As Long, ByRef dpInitPos As Double, ByRef
dpInitVel As Double, ByRef dpInitAccel As Double, ByRef dpInitDecel As Double) As Long
```

Delphi

```
function AxmMotGetParaLoad (lAxisNo : LongInt; dpInitPos : PDouble; dpInitVel : PDouble;
dpInitAccel : PDouble; dpInitDecel : PDouble) : DWord; stdcall;
```

Input / Output

Name	in/out	Format	Init Value	Explanation
AxisNo	in	long	–	채널(축) 번호(0 ~ (최대축수 – 1))
InitPos	out	double*	1000	초기화위치 (사용자가 라이브러리 Open과 동시에 이값을 가져온다.)
InitVel	out	double*	200	초기화속도 (사용자가 라이브러리 Open과 동시에 이값을 가져온다.)
InitAccel	out	double*	400	초기화가속도(사용자가 라이브러리 Open 과 동시에 이값을 가져온다.)
InitDecel	out	double*	400	초기화감속도 (사용자가 라이브러리 Open 과 동시에 이값을 가져온다.)

Return AXT_RT_SUCCESS(0000) : Successful execution of API.
* See error code Table for more information on status error codes

Description

내부 파라메타중에 28 – 31 까지 파라메타 dInitPos, dInitVel, dInitAccel, dInitDecel의 현재 라이브러리 내부 설정 값을 읽어올 때 사용한다. 이 함수와 관계된 내부 파라미터(dInitPos, dInitVel, dInitAccel, dInitDecel)은 AxmMotSetParaLoad 로 설정 가능하며, AxmMotLoadParaAll로 .Mot 파일에 저장된 값으로 설정 할 수 있다.

파일 설정되는 내부변수 초기값

No	신호관련 변수	파라메타 설명	설정 값
00	AXIS_NO	축 번호	설정 축번호
01	PULSE_OUT_METHOD	모션보드에서 서보팩 지령으로 사용될 펄스의 출력방식을 설정	TwoCwCcwHigh(6)
02	Enc[Encoder]	Encoder신호의 입력방식을 설정	ObverseSqr4Mode (3)
03	InP[InPos]	서보팩 위치결정 완료신호의 사용여부 및 Active Level을 설정	2
04	Alm[Alarm]	서보팩 알람 신호의 사용여부 및 Active level을 설정	1
05	NLimit	기구 (–)리미트 센서의 사용여부 및 Active level을 설정	1
06	PLimit	기구 (+)리미트 센서의 사용여부 및 Active level을 설정	1
07	MinSpd	모터 구동 시 초기 시작구동 속도 설정	1.0
08	MaxSpd	모터 구동 시 최대구동 속도 설정	700000.0
09	HmSig	원점센서로 사용할 신호를 설정	4
10	HmLev	원점센서의 Active Level을 설정	1
11	HmDir	원점센서 검출시 초기 검색진행 방향을	0
12	Zphas_Lev	엔코더 Z상 Active level	1
13	Zphas_Use	원점센서 검출 후 엔코더 Z상을 검출 여부 설정	0
14	Stop Signal Mode	ESTOP, SSTOP 사용시 모드	0
15	Stop Signal level	ESTOP, SSTOP 사용 레벨	0
16	VelFirst	원점검색 시 초기 고속검출 속도(속도의 단위는 Unit/pulse를 1/1로 한경우에 PPS[Pulses/sec])	10000.0

17	VelSecond	원점검색 시 1차 센서 검출 후 반대 방향으로 빠져나오는 속도	10000.0
18	VelThird	원점검색 시 1차 센서 검출 후 재 검색 진행하는 속도	2000.0
19	VelLast	원점검색 시 최종 검출 속도 설정 [원점검색의 정밀도 결정]	100.0
20	AccFirst	원점검색 시 초기 고속검출 가속도(가속도의 단위는 Unit/pulse를 1/1로 한경우에 PPS[Pulses/sec^2])	40000.0
21	AccSecond	원점검색 시 1차 센서 검출 후 반대 방향으로 빠져나오는 가속도(가속도의 단위는 Unit/pulse를 1/1로 한경우에 PPS[Pulses/sec^2])	40000.0
22	HClrTim	원점검색 완료 후 Cmd, Act위치를 Clear전 대기시간	1000.0
23	HOffset	원점검색 완료 후 기구원점 재 설정 시 이동 Offset 값	0.0
24	NSWLimit	원점검색 완료 후 적용할 (-) Software Limit 값 설정	0.0
25	PSWLimit	원점검색 완료 후 적용할 (+) Software Limit 값 설정	0.0
26	OenPulse	모터 1회전에 필요한 펄스 수 설정 [서보팩 전자 기어비 참조]	1.0
27	OneUnit	모터 1회전 시 이동되는 이동 량 [mm, 도, 등등]	1
28	InitPosition	초기화시 위치	200
29	InitVel	초기화시 속도(속도의 단위는 Unit/pulse를 1/1로 한경우에 PPS[Pulses/sec])	200
30	InitAccel	초기화시 가속도(가속도의 단위는 Unit/pulse를 1/1로 한경우에 PPS[Pulses/sec^2])	400
31	InitDecel	초기화시 감속도(감속도의 단위는 Unit/pulse를 1/1로 한경우에 PPS[Pulses/sec^2])	400
32	AbsReMode	초기화시 절대/상대 모드	0
33	ProfMode	초기화시 속도 프로파일 모드	4

28 – 31 까지 파라메타를 반환한다.

C Example

```
double dInitpos = 1000, dInitvel = 200,  dInitaccel = 400, dInitdecel =
       400;
AxmMotSetParaLoad(0, dInitpos, dInitvel, dInitaccel, dInitdecel);

double dReadInitpos,dReadInitvel,dReadInitaccel,dReadInitdecel;
AxmMotGetParaLoad(0,&dReadInitpos,&dReadInitvel, &dReadInitaccel
, &dReadInitdecel);
```

VB Example

```
Dim dInitpos ,dInitvel,dInitaccel,dInitdecel As Double
dInitpos=1000 :dInitvel = 200 :dInitaccel = 400: dInitdecel = 400
AxmMotSetParaLoad 0, dInitpos, dInitvel, dInitaccel, dInitdecel

Dim dReadInitpos,dReadInitvel,dReadInitaccel,dReadInitdecel As Double
AxmMotGetParaLoad 0, dReadInitpos, dReadInitvel, dReadInitaccel,
       dReadInitdecel
```

Delphi Example

```
var
dInitpos,dInitvel,dInitaccel,dInitdecel : Double;
dReadInitpos,dReadInitvel,dReadInitaccel,dReadInitdecel : Double;
begin

dInitpos := 1000; dInitvel := 200; dInitaccel := 400; dInitdecel := 400;
AxmMotSetParaLoad(0, dInitpos, dInitvel, dInitaccel, dInitdecel);

AxmMotGetParaLoad(0, @dReadInitpos,@dReadInitvel, @dReadInitaccel,
@dReadInitdecel);

End;
```

See Also

AxmMotLoadParaAll, AxmMotSetParaLoad, AxmMotSetParaLoad
AxmMotSetPulseOutMethod, AxmMotSetEncInputMethod, AxmMotGetEncInputMethod, AxmMotSetMoveUnitPerPulse, AxmMotGetMoveUnitPerPulse, AxmMotSetDecelMode, AxmMotGetDecelMode, AxmMotSetSRate, AxmMotGetSRate, AxmMotSetRemainPulse, AxmMotGetRemainPulse,

AxmMotSetPulseOutMethod

Purpose

지정 축에 펄스 출력 방식을 설정한다.

Format

C

```
DWORD AxmMotSetPulseOutMethod (long lAxisNo, DWORD uMethod);
```

Visual Basic

```
Function AxmMotSetPulseOutMethod (ByVal lAxisNo As Long, ByVal uMethod As Long) As Long
```

Delphi

```
function AxmMotSetPulseOutMethod (lAxisNo : LongInt; uMethod : DWord) : DWord; stdcall;
```

Input / Output

Name	in/out	Format	Init Value	Explanation
AxisNo	in	long	-	축번호(0 ~ (최대축수 - 1))
Method	In	DWORD	6	펄스 출력 방식

Method

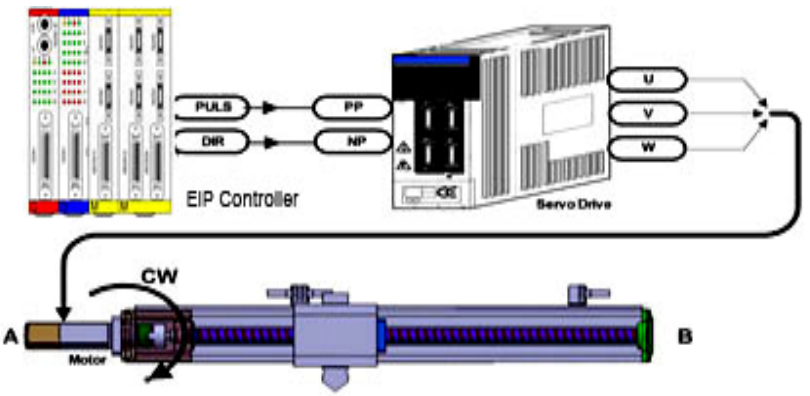
#define	Value	Explanation
OneHighLowHigh	00h	1펄스 방식, PULSE(Active High), 정방향(DIR=Low) / 역방향(DIR=High)
OneHighHighLow	01h	1펄스 방식, PULSE(Active High), 정방향(DIR=High) / 역방향(DIR=Low)
OneLowLowHigh	02h	1펄스 방식, PULSE(Active Low), 정방향(DIR=Low) / 역방향(DIR=High)
OneLowHighLow	03h	1펄스 방식, PULSE(Active Low), 정방향(DIR=High) / 역방향(DIR=Low)
TwoCcwCwHigh	04h	2펄스 방식, PULSE(CCW:역방향), DIR(CW:정방향), Active High
TwoCcwCwLow	05h	2펄스 방식, PULSE(CCW:역방향), DIR(CW:정방향), Active Low

TwoCwCcwHigh	06h	2펄스 방식, PULSE(CW:정방향), DIR(CCW:역방향), Active High
TwoCwCcwLow	07h	2펄스 방식, PULSE(CW:정방향), DIR(CCW:역방향), Active Low
TwoPhase	08h	2상(90' 위상차), PULSE lead DIR(CW: 정방향), PULSE lag DIR(CCW:역방향)
TwoPhaseReverse	09h	2상(90' 위상차), PULSE lead DIR(CCW: 정방향), PULSE lag DIR(CW:역방향)

Return AXT_RT_SUCCESS(0000) : Successful execution of API
* See error code Table more information on status error codes

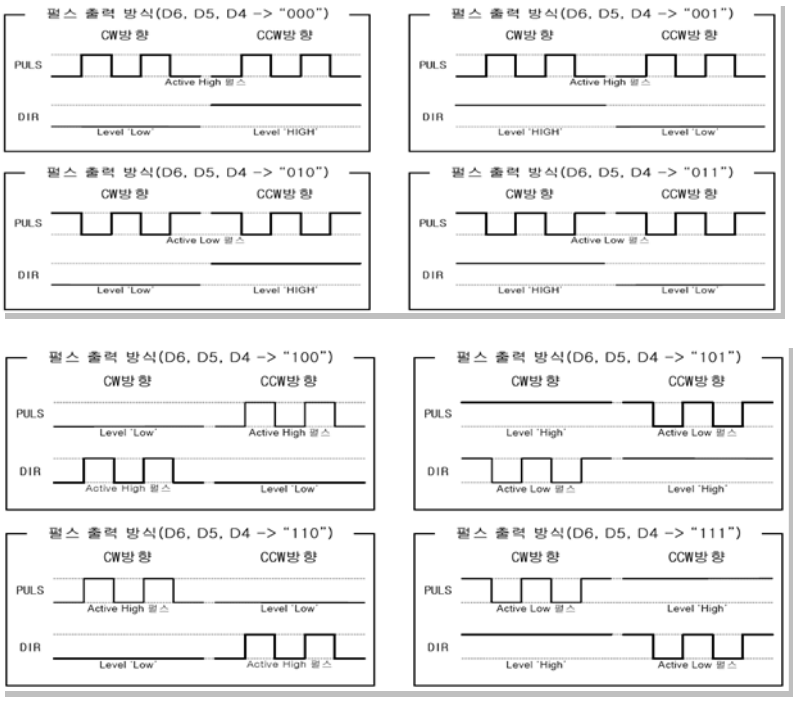
Description

펄스 출력방식에는 크게 one pulse 출력 방식과 two pulse 출력방식이 있고, 세부적으로 각각 4가지 방식으로 나누어져 있다. 사용하는 서보 드라이버와 같은 펄스 출력방식을 설정해야 한다. 펄스 출력 방식은 크게 1Pulse방식과 2Pulse방식으로 나뉘어 지며 각각의 방식은 아래와 같다.

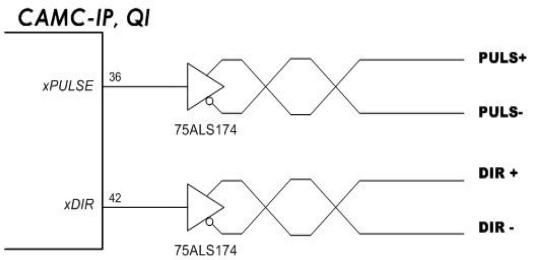


서보드라이브로 출력되는 모션제어기의 PULS, DIR의 펄스 출력방식은 One Pulse방식과 Two Pulse방식으로 구분한다.

One Pulse 방식



설정값	방향 신호(DIR)		펄스 신호(Pulse)		방식
	CW	CCW			
0	L	H	Active H Pulse		1 Pulse 방식
1	H	L			
2	L	H	Active L Pulse		
3	H	L			
4	CW Pulse	Active H	CCW Pulse	Active H	2 Pulse 방식
5		Active L		Active L	
6	CCW Pulse	Active H	CW Pulse	Active H	
7		Active L		Active L	



C Example

```
// 0축에 Two pulse 방식으로 설정
AxmMotSetPulseOutMethod(0, TwoCwCcwHigh);

// 0축의 펄스 출력 방식을 확인
DWORD Method;
AxmMotGetPulseOutMethod (0, &Method);
```

VB Example

```
' 0축에 Two pulse 방식으로 설정
AxmMotSetPulseOutMethod 0, TwoCwCcwHigh

' 0축의 펄스 출력 방식을 확인
Dim Method As Long
AxmMotGetPulseOutMethod 0, Method
```

Delphi Example

```
{ 0축의 펄스 출력 방식을 확인 }
var
Method: DWord;

Begin
{ 0축에 Two pulse 방식으로 설정 }
AxmMotSetPulseOutMethod(0, TwoCwCcwHigh);

AxmMotGetPulseOutMethod (0, @Method);
end;
```

See Also

AxmMotLoadParaAll, AxmMotSaveParaAll , AxmMotGetParaLoad, AxmMotSetParaLoad, AxmMotGetPulseOutMethod, AxmMotSetEncInputMethod, AxmMotGetEncInputMethod, AxmMotSetMoveUnitPerPulse, AxmMotGetMoveUnitPerPulse, AxmMotSetDecelPos, AxmMotGetDecelPos, AxmMotSetRemainPulse, AxmMotGetRemainPulse

AxmMotGetPulseOutMethod

Purpose

지정 축의 펄스 출력 방식 설정을 반환한다.

Format

C

```
DWORD AxmMotGetPulseOutMethod (long lAxisNo, DWORD *upMethod);
```

Visual Basic

```
Function AxmMotGetPulseOutMethod (ByVal lAxisNo As Long, ByRef upMethod As Long) As Long
```

Delphi

```
function AxmMotGetPulseOutMethod (lAxisNo : LongInt; upMethod : PDWord) : DWord; stdcall;
```

Input / Output

Name	in/out	Format	Init Value	Explanation
AxisNo	in	long	-	축번호(0 ~ (최대축수 - 1))
Method	out	DWORD*	6	펄스 출력 방식

Method

#define	Value	Explanation
OneHighLowHigh	00h	1펄스 방식, PULSE(Active High), 정방향(DIR=Low) / 역방향 (DIR=High)
OneHighHighLow	01h	1펄스 방식, PULSE(Active High), 정방향(DIR=High) / 역방향 (DIR=Low)
OneLowLowHigh	02h	1펄스 방식, PULSE(Active Low), 정방향(DIR=Low) / 역방향 (DIR=High)
OneLowHighLow	03h	1펄스 방식, PULSE(Active Low), 정방향(DIR=High) / 역방향 (DIR=Low)
TwoCcwCwHigh	04h	2펄스 방식, PULSE(CCW:역방향), DIR(CW:정방향), Active High
TwoCcwCwLow	05h	2펄스 방식, PULSE(CCW:역방향), DIR(CW:정방향), Active Low

TwoCwCcwHigh	06h	2펄스 방식, PULSE(CW:정방향), DIR(CCW:역방향), Active High
TwoCwCcwLow	07h	2펄스 방식, PULSE(CW:정방향), DIR(CCW:역방향), Active Low
TwoPhase	08h	2상(90' 위상차), PULSE lead DIR(CW: 정방향), PULSE lag DIR(CCW:역방향)
TwoPhaseReverse	09h	2상(90' 위상차), PULSE lead DIR(CCW: 정방향), PULSE lag DIR(CW:역방향)

Return AXT_RT_SUCCESS(0000) : Successful execution of API
* See error code Table more information on status error codes

Description

AxmMotSetPulseOutMethod로 설정한 지정 축의 펄스 출력 방식 설정을 반환한다.

C Example

```
// 0축에 Two pulse 방식으로 설정
AxmMotSetPulseOutMethod(0, TwoCwCcwHigh);

// 0축의 펄스 출력 방식을 확인
DWORD Method;
AxmMotGetPulseOutMethod (0, &Method);
```

VB Example

```
' 0축에 Two pulse 방식으로 설정
AxmMotSetPulseOutMethod 0, TwoCwCcwHigh

' 0축의 펄스 출력 방식을 확인
Dim Method As Long
AxmMotGetPulseOutMethod 0, Method
```

Delphi Example

```
{ 0축의 펄스 출력 방식을 확인 }
var
Method: DWord;

Begin
{ 0축에 Two pulse 방식으로 설정 }
```

```
AxmMotSetPulseOutMethod(0, TwoCwCcwHigh);

AxmMotGetPulseOutMethod (0, @Method);
end;
```

See Also

AxmMotSetPulseOutMethod, AxmMotSetEncInputMethod, AxmMotGetEncInputMethod, AxmMotSetMoveUnitPerPulse, AxmMotGetMoveUnitPerPulse, AxmMotSetDecelPos, AxmMotGetDecelPos, AxmMotSetRemainPulse, AxmMotGetRemainPulse,

AxmMotSetEncInputMethod

Purpose

지정 축에 Encoder 입력 방식을 설정한다.

Format

C

```
DWORD AxmMotSetEncInputMethod (long lAxisNo, DWORD uMethod);
```

Visual Basic

```
Function AxmMotSetEncInputMethod (ByVal lAxisNo As Long, ByVal uMethod As Long) As Long
```

Delphi

```
function AxmMotSetEncInputMethod (lAxisNo : LongInt; uMethod : DWord) : DWord; stdcall;
```

Input / Output

Name	in/out	Format	Init Value	Explanation
AxisNo	in	long	-	축번호(0 ~ (최대축수 - 1))
Method	In	DWORD	3	Encoder 입력 방식

Method

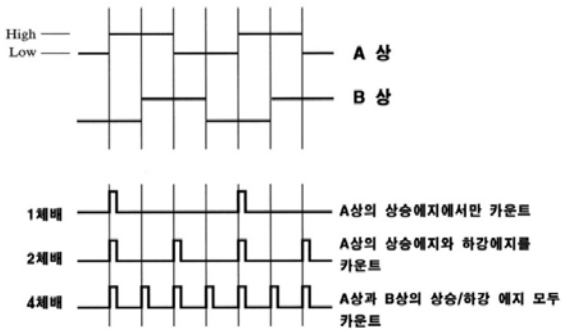
#define	Value	Explanation
ObverseUpDownMode	00h	정방향 Up / Down
ObverseSqr1Mode	01h	정방향 1체배
ObverseSqr2Mode	02h	정방향 2체배
ObverseSqr4Mode	03h	정방향 4체배
ReverseUpDownMode	04h	역방향 Up / Down
ReverseSqr1Mode	05h	역방향 1체배
ReverseSqr2Mode	06h	역방향 2체배
ReverseSqr4Mode	07h	역방향 4체배

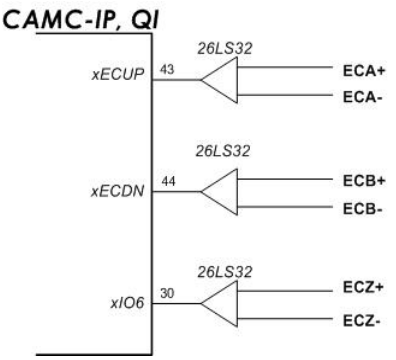
Return AXT_RT_SUCCESS(0000) : Successful execution of API

* See error code Table more information on status error codes

Description

Encorder 입력 방식은 8가지로 구분되어 있으며, 대 부분의 드라이버가 4체배 방식을 사용하고 있다.





C Example

```
// 0축에 Encoder 입력 방식을 4체배 방식으로 설정
AxmMotSetEncInputMethod(0, ObverseSqr4Mode);

// 0축의 Encoder 입력 방식을 확인
DWORD Method;
AxmMotGetEncInputMethod (0, &Method);
```

VB Example

```
‘ 0축에 Encoder 입력 방식을 4체배 방식으로 설정
AxmMotSetEncInputMethod 0, ObverseSqr4Mode

‘ 0축의 Encoder 입력 방식을 확인
Dim Method As Long
AxmMotGetEncInputMethod 0, Method
```

Delphi Example

```
{ 0축의 Encoder 입력 방식을 확인 }
var
Method: DWord;

Begin
{ 0축에 Encoder 입력 방식을 4체배 방식으로 설정 }
AxmMotSetEncInputMethod(0, ObverseSqr4Mode);

AxmMotGetEncInputMethod (0, @Method);
end;
```

See Also

AxmMotLoadParaAll, AxmMotSaveParaAll, AxmMotGetParaLoad, AxmMotSetParaLoad, AxmMotSetPulseOutMethod, AxmMotGetPulseOutMethod, AxmMotGetEncInputMethod, AxmMotSetMoveUnitPerPulse, AxmMotGetMoveUnitPerPulse, AxmMotSetDecelPos, AxmMotGetDecelPos, AxmMotSetRemainPulse, AxmMotGetRemainPulse,

AxmMotGetEncInputMethod

Purpose

지정 축의 Encoder 입력 방식 설정을 반환한다.

Format

C

```
DWORD AxmMotGetEncInputMethod (long lAxisNo, DWORD *upMethod);
```

Visual Basic

```
Function AxmMotGetEncInputMethod (ByVal lAxisNo As Long, ByRef upMethod As Long) As Long
```

Delphi

```
function AxmMotGetEncInputMethod (lAxisNo : LongInt; upMethod : PDWord) : DWord; stdcall;
```

Input / Output

Name	in/out	Format	Init Value	Explanation
AxisNo	in	long	–	축번호(0 ~ (최대축수 – 1))
Method	out	DWORD*	3	Encoder 입력 방식

Method

#define	Value	Explanation
ObverseUpDownMode	00h	정방향 Up / Down
ObverseSqr1Mode	01h	정방향 1체배
ObverseSqr2Mode	02h	정방향 2체배
ObverseSqr4Mode	03h	정방향 4체배
ReverseUpDownMode	04h	역방향 Up / Down
ReverseSqr1Mode	05h	역방향 1체배
ReverseSqr2Mode	06h	역방향 2체배
ReverseSqr4Mode	07h	역방향 4체배

Return AXT_RT_SUCCESS(0000) : Successful execution of API

* See error code Table more information on status error codes

Description

AxmMotSetEncInputMethod로 설정한 지정 축의 Encoder 입력 방식 설정을 반환한다.

C Example

```
// 0축에 Encoder 입력 방식을 4체배 방식으로 설정
AxmMotSetEncInputMethod(0, ObverseSqr4Mode);

// 0축의 Encoder 입력 방식을 확인
DWORD Method;
AxmMotGetEncInputMethod (0, &Method);
```

VB Example

```
' 0축에 Encoder 입력 방식을 4체배 방식으로 설정
AxmMotSetEncInputMethod 0, ObverseSqr4Mode

' 0축의 Encoder 입력 방식을 확인
Dim Method As Long
AxmMotGetEncInputMethod 0, Method
```

Delphi Example

```
{ 0축의 Encoder 입력 방식을 확인 }
var
Method: DWord;

Begin
{ 0축에 Encoder 입력 방식을 4체배 방식으로 설정 }
AxmMotSetEncInputMethod(0, ObverseSqr4Mode);

AxmMotGetEncInputMethod (0, @Method);
end;
```

See Also

AxmMotLoadParaAll, AxmMotSaveParaAll, AxmMotGetParaLoad, AxmMotSetParaLoad, AxmMotSetPulseOutMethod, AxmMotGetPulseOutMethod, AxmMotSetEncInputMethod, AxmMotGetEncInputMethod, AxmMotSetMoveUnitPerPulse, AxmMotGetMoveUnitPerPulse AxmMotSetDecelPos, AxmMotGetDecelPos, AxmMotSetRemainPulse, AxmMotGetRemainPulse

AxmMotSetMoveUnitPerPulse

Purpose

지정 축에 움직이는 거리 당 출력되는 펄스를 설정한다.

Format

C

```
DWORD AxmMotSetMoveUnitPerPulse(long lAxisNo, double dUnit, long lPulse);
```

Visual Basic

```
Function AxmMotSetMoveUnitPerPulse(ByVal lAxisNo As Long, ByVal dUnit As Double, ByVal lPulse As Long) As Long
```

Delphi

```
procedure AxmMotSetMoveUnitPerPulse(lAxisNo : LongInt; dUnit : Double; lPulse : LongInt) : DWord; stdcall;
```

Input / Output

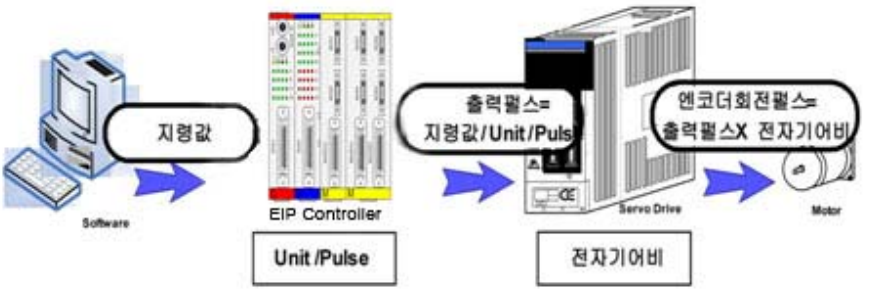
Name	in/out	Format	Init Value	Explanation
AxisNo	in	long	–	축번호(0 ~ (최대축수 – 1))
dUnit	In	double	1	단위 이동거리
lPulse	In	long	1	단위 펄스값

Return AXT_RT_SUCCESS(0000) : Successful execution of API

* See error code Table more information on status error codes

Description

기본적으로 사용자가 라이브러리의 명령어에서 지정해주는 이동거리 지령값은 아래의 그림에서와 같이 두 번의 단위변환 과정을 거치게 된다.



소프트웨어 상에서 이동량을 입력하면, 설정된 Unit/Pulse라는 파라미터에 의해 모션 제어기의 출력펄스 수가 결정된다. 그리고 이 출력펄스는 서보드라이브에 설정되어있는 전자기어비 라는 파라미터에 의해 또 한번 변환되어 실제 모터를 구동시키는 구동펄스 수가 결정된다.

여기서 Unit/Pulse는 모션 구동함수를 실행함에 있어서 주어지는 값들의 단위를 지정해주기 위한 파라미터이며 이 값을 설정하는 함수가 AxmMotSetMoveUnitPerPulse 이다. 쉽게 말해서 Unit에 10을 설정하고, Pulse에 100을 설정한다면, 모션구동함수에서 10을 명령하면 100개의 펄스가 서보드라이브로 출력된다는 뜻이다.

※ 서보드라이브의 Electronic gear Ratio (전자기어비) 설명

Unit/Pulse는 모션제어기에서 설정하는 파라미터인데 반하여 전자기어비는 서보드라이브에서 설정하는 파라미터이다.

$$[\text{전자 기어비}] = \frac{[\text{서보모터가 1회전 할 때 발생하는 엔코더 펄스 수}]}{[\text{서보모터가 일회전 할 때 모션제어기의 출력지령펄스 수}]}$$

17비트의 엔코더를 사용한다고 하면, 위의 식으로 131,072개의 엔코더의 펄스 수는 모션제어기에서 10,000로 줄일 수 있다. 즉 서보모터 엔코더의 회전 정밀도를 기구부의 정밀도 만큼으로 낮추는 것이 전자 기어비 이다.

예를 들어 20mm 볼 스크루, 기어비 : 1/2이고 모터 1회전에 필요한 펄스 수가 10000 일때, lPulse = 10000과 dUnit = 10으로 설정하면 1mm 움직이는 거리 당 출력되는 펄스 수 = 1000펄스가 설정된다

논리적 단위 속도에 대한 실제 펄스 출력속도는 Pulses/sec를 설정한다. 여기서 논리적 단위 속도라는 것은 속도 지정함수에서 사용하는 속도,가속도에 대한 단위량을 의미한다. 하지만 사용자의 특성에 따라 속도에 대한 단위가 다를수있다. 즉, 어떤사용자는 속도 단위를 RPM으로 표현하는 것이 용이할수 있고, 어떤사용자는 m/sec로 표현하는 것이 용이할 수 있다. 1회전에 필요한 펄스 수가 3600 펄스인 경우에 인자 lPulse에 3600을 넣으면된다. 내부에서는 3600/60, 즉 60PPS로 설정되는데 그이유는 RPM은 분당 회전수이므로 60을 나눠준다. 그래서 출력해야 될 펄스는 1분에 3600펄스가 나가게 된다.

설정 속도 단위가 RPM(Revolution Per Minute)으로 맞추고 싶다면.
ex> rpm 계산:
4500 rpm ?
unit/ pulse = 1 : 1이면 pulse/ sec 초당 펄스수가 되는데
4500 rpm에 맞추고 싶다면 4500 / 60 초 : 75회전/ 1초
모터가 1회전에 몇 펄스인지 알아야 된다. 이것은 Encoder에 Z상을 검색해보면 알수있다.
1회전:1800 펄스라면 75 x 1800 = 135000 펄스가 필요하게 된다.
AxmMotSetMoveUnitPerPulse에 Unit = 1, Pulse = 1800 넣어 동작시킨다.

C Example

```
// 0축에 움직이는 거리 당 출력되는 펄스 수를 100으로 설정
AxmMotSetMoveUnitPerPulse(0, 1, 100);

// 0축에 움직이는 거리 당 출력되는 펄스 수를 반환한다
long lpPulse;
double  dpUnit;
AxmMotGetMoveUnitPerPulse (0, &dpUnit, &lpPulse);
```

VB Example

```
' 0축에 움직이는 거리 당 출력되는 펄스 수를 100으로 설정
AxmMotSetMoveUnitPerPulse 0, 1, 100

' 0축에 움직이는 거리 당 출력되는 펄스 수를 반환한다
Dim lpPulse As Long
Dim dpUnit As Double
AxmMotGetMoveUnitPerPulse 0, dpUnit, lpPulse
```

Delphi Example

```
{ 0축에 움직이는 거리 당 출력되는 펄스 수를 반환한다 }
var
lpPulse: LongInt;
dpUnit: Double;

begin
{ 0축에 움직이는 거리 당 출력되는 펄스 수를 100으로 설정 }
AxmMotSetMoveUnitPerPulse(0, 1, 100);
AxmMotGetMoveUnitPerPulse (0, @dpUnit, @lpPulse);
end;
```

See Also

AxmMotLoadParaAll, AxmMotSaveParaAll, AxmMotGetParaLoad, AxmMotSetParaLoad, AxmMotSetPulseOutMethod, AxmMotGetPulseOutMethod, AxmMotSetEncInputMethod, AxmMotGetEncInputMethod, AxmMotGetMoveUnitPerPulse, AxmMotSetDecelPos, AxmMotGetDecelPos, AxmMotSetRemainPulse, AxmMotGetRemainPulse,

AxmMotGetMoveUnitPerPulse

Purpose

지정 축에 움직이는 거리 당 출력되는 펄스를 반환한다.

Format

C

DWORD AxmMotGetMoveUnitPerPulse (long lAxisNo, double *dpUnit, long *lpPulse);

Visual Basic

Function AxmMotGetMoveUnitPerPulse (ByVal lAxisNo As Long, ByRef dpUnit As Double, ByRef lpPulse As Long) As Long

Delphi

function AxmMotGetMoveUnitPerPulse (lAxisNo : LongInt; dpUnit : PDouble; lpPulse : PLongInt) : DWord; stdcall;

Input / Output

Name	in/out	Format	Init Value	Explanation
AxisNo	in	long	–	축번호(0 ~ (최대축수 – 1))
dUnit	out	double*	1	단위 이동거리
lPulse	out	long*	1	단위 펄스값

Return AXT_RT_SUCCESS(0000) : Successful execution of API

* See error code Table more information on status error codes

Description

AxmMotSetMoveUnitPerPulse로 설정한 지정 축에 움직이는 거리 당 출력되는 펄스를 반환한다

*참고 : 필요한 데이터만 가지고 오고싶을때는 필요하지 않는 변수에 NULL 넣어 사용가능하다.

C Example

```
// 0축에 움직이는 거리 당 출력되는 펄스 수를 100으로 설정
AxmMotSetMoveUnitPerPulse(0, 1, 100);

// 0축에 움직이는 거리 당 출력되는 펄스 수를 반환한다
long lpPulse;
double  dpUnit;
AxmMotGetMoveUnitPerPulse (0, &dpUnit, &lpPulse);
```

VB Example

```
' 0축에 움직이는 거리 당 출력되는 펄스 수를 100으로 설정
AxmMotSetMoveUnitPerPulse 0, 1, 100

' 0축에 움직이는 거리 당 출력되는 펄스 수를 반환한다
Dim lpPulse As Long
Dim dpUnit As Double
AxmMotGetMoveUnitPerPulse 0, dpUnit, lpPulse
```

Delphi Example

```
{ 0축에 움직이는 거리 당 출력되는 펄스 수를 반환한다 }
var
lpPulse: LongInt;
dpUnit: Double;

begin
{ 0축에 움직이는 거리 당 출력되는 펄스 수를 100으로 설정 }
AxmMotSetMoveUnitPerPulse(0, 1, 100);
AxmMotGetMoveUnitPerPulse (0, @dpUnit, @lpPulse);
end;
```

See Also

AxmMotLoadParaAll, AxmMotSaveParaAll, AxmMotGetParaLoad, AxmMotSetParaLoad, AxmMotSetPulseOutMethod, AxmMotGetPulseOutMethod, AxmMotSetEncInputMethod, AxmMotGetEncInputMethod, AxmMotSetMoveUnitPerPulse, AxmMotSetDecelPos, AxmMotGetDecelPos, AxmMotSetRemainPulse, AxmMotGetRemainPulse,

AxmMotSetDecelPos

Purpose

지정 축에 감속 시작 포인트 검출 방식을 설정한다.

Format

C

```
DWORD AxmMotSetDecelPos (long lAxisNo, DWORD uMethod);
```

Visual Basic

```
Function AxmMotSetDecelPos (ByVal lAxisNo As Long, ByVal uMethod As Long) As Long
```

Delphi

```
function AxmMotSetDecelPos (lAxisNo : LongInt; uMethod : DWord) : DWord; stdcall;
```

Input / Output

Name	in/out	Format	Init Value	Explanation
AxisNo	in	long	-	축번호(0 ~ (최대축수 - 1))
Method	In	DWORD	-	감속 시작 포인트 검출 방식

Method

#define	Value	Explanation
AutoDetect	00h	자동 검출 방식
RestPulse	01h	나머지 검출 방식

Return AXT_RT_SUCCESS(0000) : Successful execution of API

* See error code Table more information on status error codes

Description

검출 방식은 자동 검출 방식(Autodetect)과, 나머지 검출 방식(RestPulse)이 있다. 초기 값은 자동 검출 방식으로 설정되어 있다.

C Example

```
// 0축에 감속 시작 포인트 검출 방식을 자동 검출 방식으로 설정
AxmMotSetDecelPos(0, AutoDetect);
```

```
// 0축의 감속 시작 포인트 검출 방식을 확인
DWORD Method;
AxmMotGetDecelPos(0, &Method);
```

VB Example

```
' 0축에 감속 시작 포인트 검출 방식을 자동 검출 방식으로 설정
AxmMotSetDecelPos 0, AutoDetect

' 0축의 감속 시작 포인트 검출 방식을 확인
Dim Method As Long
AxmMotGetDecelPos 0, Method
```

Delphi Example

```
{ 0축의 감속 시작 포인트 검출 방식을 확인 }
var
Method: DWord;

Begin
{ 0축에 감속 시작 포인트 검출 방식을 자동 검출 방식으로 설정 }
AxmMotSetDecelPos(0, AutoDetect);

AxmMotGetDecelPos(0, @Method);
end;
```

See Also

AxmMotLoadParaAll, AxmMotSaveParaAll, AxmMotGetParaLoad, AxmMotSetParaLoad, AxmMotSetPulseOutMethod, AxmMotGetPulseOutMethod, AxmMotSetEncInputMethod, AxmMotGetEncInputMethod, AxmMotSetMoveUnitPerPulse, AxmMotGetMoveUnitPerPulse, AxmMotGetDecelPos, AxmMotSetRemainPulse, AxmMotGetRemainPulse,

AxmMotGetDecelPos

Purpose

지정 축의 감속 시작 포인트 검출 방식을 반환한다

Format

C

```
DWORD AxmMotGetDecelPos (long lAxisNo, DWORD *upMethod);
```

Visual Basic

```
Function AxmMotGetDecelPos (ByVal lAxisNo As Long, ByRef upMethod As Long) As Long
```

Delphi

```
function AxmMotGetDecelPos (lAxisNo : LongInt; upMethod : PDWord) : DWord; stdcall;
```

Input / Output

Name	in/out	Format	Init Value	Explanation
AxisNo	in	long	-	축번호(0 ~ (최대축수 - 1))
Method	out	long*	-	감속 시작 포인트 검출 방식

Method

#define	Value	Explanation
AutoDetect	00h	자동 검출 방식
RestPulse	01h	나머지 검출 방식

Return AXT_RT_SUCCESS(0000) : Successful execution of API

* See error code Table more information on status error codes

Description

AxmMotSetDecelPos 로 설정한 지정 축의 감속 시작 포인트 검출 방식을 반환한다

C Example

```
// 0축에 감속 시작 포인트 검출 방식을 자동 검출 방식으로 설정
AxmMotSetDecelPos(0, AutoDetect);

// 0축의 감속 시작 포인트 검출 방식을 확인
DWORD Method;
AxmMotGetDecelPos(0, &Method);
```

VB Example

```
' 0축에 감속 시작 포인트 검출 방식을 자동 검출 방식으로 설정
AxmMotSetDecelPos 0, AutoDetect

' 0축의 감속 시작 포인트 검출 방식을 확인
Dim Method As Long
AxmMotGetDecelPos 0, Method
```

Delphi Example

```
{ 0축의 감속 시작 포인트 검출 방식을 확인 }
var
Method: DWord;

Begin
{ 0축에 감속 시작 포인트 검출 방식을 자동 검출 방식으로 설정 }
AxmMotSetDecelPos(0, AutoDetect);

AxmMotGetDecelPos(0, @Method);
end;
```

See Also

AxmMotLoadParaAll, AxmMotSaveParaAll, AxmMotGetParaLoad, AxmMotSetParaLoad, AxmMotSetPulseOutMethod, AxmMotGetPulseOutMethod, AxmMotSetEncInputMethod, AxmMotGetEncInputMethod, AxmMotSetMoveUnitPerPulse, AxmMotGetMoveUnitPerPulse AxmMotSetDecelPos, AxmMotSetRemainPulse, AxmMotGetRemainPulse,

AxmMotSetRemainPulse

Purpose

지정 축에 수동 감속 모드에서 잔량 펄스를 설정한다.

Format

C

```
DWORD AxmMotSetRemainPulse (long lAxisNo, DWORD uData);
```

Visual Basic

```
Function AxmMotSetRemainPulse (ByVal lAxisNo As Long, ByVal uData As Long) As Long
```

Delphi

```
Function AxmMotSetRemainPulse (lAxisNo : LongInt; uData : DWord); DWord; stdcall;
```

Input / Output

Name	in/out	Format	Init Value	Explanation
AxisNo	in	long	-	축번호(0 ~ (최대축수 - 1)))
Data	In	DWORD	-	설정된 잔량 펄스

Return AXT_RT_SUCCESS(0000) : Successful execution of API

* See error code Table more information on status error codes

Description

AxmMotSetDecelPos 에서 AutoDetect (0)로 Setting되면 AxmMotSetRemainPulse함수를 이용하여 구동 감속후 구동 펄스를 설정한다.
예를들어 AxmMoveStartPos를 이용하여 1000펄스를 보내고 5펄스 잔량펄스를 설정하게되면 마지막 995펄스부터 AxmMotSetMinVel 로 설정한 속도로 감속하여 그속도로 5펄스 만큼 유지하면서 감속하게 된다. 마지막 시점에서 감속을 유지할 때 사용한다.

AxmMotSetDecelMode 에서 RestPulse (1)로 Setting되면 AxmMotSetRemainPulse함수를 이용하여 감속 시점에 대한 남은 펄스 수를 설정한다.

예를들어 AxmMoveStartPos를 이용하여 1000펄스를 설정하고 가속시 100 펄스가 필요한 구동을 하였다면, AxmMotSetRemainPulse를 사용하여 설정한 값에 따라 다음과 같이 동작한다.

나머지 펄스 설정값이 100 보다 작을 경우 : 출발 정지 속도까지 감속하지 전에 구동이 완료된다.

나머지 펄스 설정값이 100 보다 클 경우 : 출발 정지 속도로 감속하여 100 보다 큰 값 만큼 구동 후 완료한다.

주의점) 입력 값이 펄스 단위이기 때문에 AxmMotSetMoveUnitPerPulse 로 설정한 값을 사용하여 위치값(Unit)을 펄스 단위로 변환하여야 한다.

C Example

```
// 0축에 수동 감속 모드에서 잔량 펄스를 설정
AxmMotSetRemainPulse(0, 1000);

// 0축의 수동 감속 모드에서 잔량 펄스를 확인
DWORD Data;
AxmMotGetRemainPulse (0, &Data);
```

VB Example

```
' 0축에 수동 감속 모드에서 잔량 펄스를 설정
AxmMotSetRemainPulse 0, 1000

' 0축의 수동 감속 모드에서 잔량 펄스를 확인
Dim Data As Long
AxmMotGetRemainPulse 0, Data
```

Delphi Example

```
{ 0축의 수동 감속 모드에서 잔량 펄스를 확인 }
var
Data: DWord;
```

```
Begin
{ 0축에 수동 감속 모드에서 잔량 펄스를 설정 }
AxmMotSetRemainPulse(0, 1000);
AxmMotGetRemainPulse (0, @Data);
end;
```

See Also

AxmMotLoadParaAll, AxmMotSaveParaAll, AxmMotGetParaLoad, AxmMotSetParaLoad, AxmMotSetPulseOutMethod, AxmMotGetPulseOutMethod, AxmMotSetEncInputMethod, AxmMotGetEncInputMethod, AxmMotSetMoveUnitPerPulse, AxmMotGetMoveUnitPerPulse AxmMotSetDecelPos, AxmMotGetDecelPos, AxmMotGetRemainPulse,

AxmMotGetRemainPulse

Purpose

지정 축의 수동 감속 모드에서 잔량 펄스를 반환한다.

Format

C

DWORD AxmMotGetRemainPulse (long lAxisNo, DWORD *upData);

Visual Basic

Function AxmMotGetRemainPulse (ByVal lAxisNo As Long,ByRef upData As Long) As Long

Delphi

function AxmMotGetRemainPulse (lAxisNo : LongInt; upData : PDWord) : DWord; stdcall;

Input / Output

Name	in/out	Format	Init Value	Explanation
AxisNo	in	long	-	축번호(0 ~ (최대축수 - 1))
Data	out	DWORD*	-	설정된 잔량 펄스

Return AXT_RT_SUCCESS(0000) : Successful execution of API

* See error code Table more information on status error codes

Description

AxmMotSetRemainPulse 함수를 사용해서 지정한 축의 수동 감속 모드에서 잔량 펄스를 반환한다.

C Example

```
// 0축에 수동 감속 모드에서 잔량 펄스를 설정
AxmMotSetRemainPulse(0, 1000);

// 0축의 수동 감속 모드에서 잔량 펄스를 확인
DWORD Data;
AxmMotGetRemainPulse (0, &Data);
```

VB Example

```
' 0축에 수동 감속 모드에서 잔량 펄스를 설정
AxmMotSetRemainPulse 0, 1000
```

```
' 0축의 수동 감속 모드에서 잔량 펄스를 확인
Dim Data As Long
AxmMotGetRemainPulse 0, Data
```

Delphi Example

```
{ 0축의 수동 감속 모드에서 잔량 펄스를 확인 }
var
Data: DWord;

Begin
{ 0축에 수동 감속 모드에서 잔량 펄스를 설정 }
AxmMotSetRemainPulse(0, 1000);
AxmMotGetRemainPulse (0, @Data);
end;
```

See Also

AxmMotLoadParaAll, AxmMotSaveParaAll, AxmMotGetParaLoad, AxmMotSetParaLoad, AxmMotSetPulseOutMethod, AxmMotGetPulseOutMethod, AxmMotSetEncInputMethod, AxmMotGetEncInputMethod, AxmMotSetMoveUnitPerPulse, AxmMotGetMoveUnitPerPulse AxmMotSetDecelPos, AxmMotGetDecelPos, AxmMotSetRemainPulse

AxmMotSetMaxVel

Purpose

지정 축에 등속도 구동 함수에서의 최고 속도를 설정한다.

Format

C

```
DWORD AxmMotSetMaxVel (long lAxisNo, double dVel);
```

Visual Basic

```
Function AxmMotSetMaxVel (ByVal lAxisNo As Long, ByVal dVel As Doubel) As Long
```

Delphi

```
function AxmMotSetMaxVel (lAxisNo : LongInt; dVel : Double); DWord; stdcall;
```

Input / Output

Name	in/out	Format	Init Value	Explanation
AxisNo	in	long	-	축번호(0 ~ (최대축수 - 1))
Vel	In	double	700000	최고 구동 속도 (구동속도의 단위는 Unit/pulse를 1/1로 한 경우에 PPS[Pulses/sec])

Return AXT_RT_SUCCESS(0000) : Successful execution of API

* See error code Table more information on status error codes

Description

지정한 축에 대한 모든 구동함수에서 입력되는 속도값을 AxmMotSetMaxVel에서 설정한 속도로 제한한다.

주의사항: 최대 10MPPS까지 설정이 가능하다. 즉 AxmMotSetMoveUnitPerPulse 함수로 설정된 Unit/Pulse 값에 따라 최대 속도 값이 다르게 적용되는데, 예를 들어 Unit/Pulse 가 0.01이라면 AxmMotSetMaxVel로 설정할 수 있는 최대 값은 (10M/100) 이다.

C Example

```
// 0축에 정속도 구동을 위한 최고 속도 1000으로 설정
AxmMotSetMaxVel (0, 1000);

// 0축에 정속도 구동 최고 속도 확인
double Vel;
AxmMotGetMaxVel (0, &Vel);
```

VB Example

```
' 0축에 정속도 구동을 위한 최고 속도 1000으로 설정
AxmMotSetMaxVel 0, 1000

' 0축에 정속도 구동 최고 속도 확인
Dim Vel As Double
AxmMotGetMaxVel 0, Vel
```

Delphi Example

```
{ 0축에 정속도 구동 최고 속도 확인 }
var
Vel: Double;

Begin
{ 0축에 정속도 구동을 위한 최고 속도 1000으로 설정 }
AxmMotSetMaxVel (0, 1000);
AxmMotGetMaxVel (0, @Vel);
end;
```

See Also

AxmMotLoadParaAll, AxmMotSaveParaAll, AxmMotGetParaLoad, AxmMotSetParaLoad, AxmMotGetMaxVel, AxmMotSetAbsRelMode, AxmMotGetAbsRelMode, AxmMotSetProfileMode, AxmMotGetProfileMode, AxmMotSetAccelUnit, AxmMotGetAccelUnit, AxmMotSetMinVel, AxmMotGetMinVel, AxmMotSetAccelJerk, AxmMotGetAccelJerk, AxmMotSetDecelJerk, AxmMotGetDecelJerk

AxmMotGetMaxVel

Purpose

지정 축의 등속도 구동 함수에서의 최고 속도를 반환한다.

Format

C

DWORD AxmMotGetMaxVel (long lAxisNo, double *dpVel);

Visual Basic

Function AxmMotGetMaxVel (ByVal lAxisNo As Long, ByRef dpVel As Double) As Long

Delphi

function AxmMotGetMaxVel (lAxisNo : LongInt; dpVel : PDouble); DWord; stdcall;

Input / Output

Name	in/out	Format	Init Value	Explanation
AxisNo	in	long	-	축번호(0 ~ (최대축수 - 1))
Vel	out	double*	700000	최고 구동 속도 (구동속도의 단위는 Unit/pulse를 1/1로 한 경우에 PPS[Pulses/sec])

Return AXT_RT_SUCCESS(0000) : Successful execution of API

* See error code Table more information on status error codes

Description

지정 축의 등속도 구동 함수에서의 최고 속도를 반환한다.

C Example

```
// 0축에 정속도 구동을 위한 최고 속도 1000으로 설정
AxmMotSetMaxVel (0, 1000);

// 0축에 정속도 구동 최고 속도 확인
double Vel;
AxmMotGetMaxVel (0, &Vel);
```

VB Example

```
' 0축에 정속도 구동을 위한 최고 속도 1000으로 설정
AxmMotSetMaxVel 0, 1000

' 0축에 정속도 구동 최고 속도 확인
Dim Vel As Double
AxmMotGetMaxVel 0, Vel
```

Delphi Example

```
{ 0축에 정속도 구동 최고 속도 확인 }
var
Vel: Double;

Begin
{ 0축에 정속도 구동을 위한 최고 속도 1000으로 설정 }
AxmMotSetMaxVel (0, 1000);
AxmMotGetMaxVel (0, @Vel);
end;
```

See Also

AxmMotLoadParaAll, AxmMotSaveParaAll, AxmMotGetParaLoad, AxmMotSetParaLoad, AxmMotSetMaxVel, AxmMotSetAbsRelMode, AxmMotGetAbsRelMode, AxmMotSetProfileMode, AxmMotGetProfileMode, AxmMotSetAccelUnit, AxmMotGetAccelUnit, AxmMotSetMinVel, AxmMotGetMinVel, AxmMotSetAccelJerk, AxmMotGetAccelJerk, AxmMotSetDecelJerk, AxmMotGetDecelJerk

AxmMotSetAbsRelMode

Purpose

지정 축의 이동 거리 계산 모드를 설정한다.

Format

C

```
DWORD AxmMotSetAbsRelMode (long lAxisNo, DWORD uAbsRelMode);
```

Visual Basic

```
Function AxmMotSetAbsRelMode (ByVal lAxisNo As Long, ByVal uAbsRelMode As Long) As Long
```

Delphi

```
function AxmMotSetAbsRelMode(lAxisNo : LongInt; uAbsRelMode : DWord) : DWord; stdcall;
```

Input/Output

Name	in/out	Format	Init Value	Explanation
AxisNo	in	long	-	축번호(0 ~ (최대축수 - 1))
AbsRelMode	In	DWORD	0	펄스 출력 방식

AbsRelMode

#define	Value	Explanation
POS_ABS_MODE	00h	위치구동 절대모드
POS_REL_MODE	01h	위치구동 상대모드

Return AXT_RT_SUCCESS(0000) : Successful execution of API

* See error code Table more information on status error codes

Description

절대위치 모드(0)로 설정한 경우 이동 함수에 사용하는 위치값은 절대위치값이 된다. 예를 들어 현재 위치값이 '10'일 때 절대위치 모드에서 '100' 위치로 이동할 경우 현재 위치에서 '+90'만큼 이동하여 '100'위치에서 정지하게 된다.

상대위치 모드(1)로 설정한 경우는 이동 함수에 사용하는 위치값은 현재 위치에서 이동할 이동량이 된다. 예를 들어 현재 위치값이 '10일 대 상대 위치 모드에서 '100' 만큼 이동할 경우, 현재 위치에서 '+100'만큼 이동하여 '110'의 위치에서 정지하게 된다.

C Example

```
// 0축을 절대모드로 설정.
AxmMotSetAbsRelMode(0, POS_ABS_MODE);

// 0축의 이동 모드 확인.
DWORD Mode;
AxmMotGetAbsRelMode(0, &Mode);
```

VB Example

```
' 0축을 절대모드로 설정.
AxmMotSetAbsRelMode 0, POS_ABS_MODE

' 0축의 이동 모드 확인
Dim Mode As Long
AxmMotGetAbsRelMode 0, Mode
```

Delphi Example

```
{ 0축의 이동 모드 확인 }
var
Mode : DWord;

Begin
{ 0축을 절대모드로 설정. }
AxmMotSetAbsRelMode(0, POS_ABS_MODE);

AxmMotGetAbsRelMode (0, @Mode);
end;
```

See Also

AxmMotLoadParaAll, AxmMotSaveParaAll, AxmMotGetParaLoad, AxmMotSetParaLoad, AxmMotSetMaxVel, AxmMotGetMaxVel, AxmMotGetAbsRelMode, AxmMotSetProfileMode, AxmMotGetProfileMode, AxmMotSetAccelUnit, AxmMotGetAccelUnit, AxmMotSetMinVel, AxmMotGetMinVel, AxmMotSetAccelJerk, AxmMotGetAccelJerk, AxmMotSetDecelJerk, AxmMotGetDecelJerk

AxmMotGetAbsRelMode

Purpose

지정 축의 설정한 이동 거리 계산 모드를 반환한다.

Format

C

DWORD AxmMotGetAbsRelMode (long lAxisNo, DWORD *upAbsRelMode);

Visual Basic

Function AxmMotGetAbsRelMode (ByVal lAxisNo As Long, ByRef upAbsRelMode As Long) As Long

Delphi

function AxmMotGetAbsRelMode (lAxisNo : LongInt; upAbsRelMode : PDWord) : DWord;; stdcall;

Input/Output

Name	in/out	Format	Init Value	Explanation
AxisNo	in	long	-	축번호(0 ~ (최대축수 - 1))
AbsRelMode	out	DWORD*	0	펄스 출력 방식

AbsRelMode

#define	Value	Explanation
POS_ABS_MODE	00h	위치구동 절대모드
POS_REL_MODE	01h	위치구동 상대모드

Return AXT_RT_SUCCESS(0000) : Successful execution of API

* See error code Table more information on status error codes

Description

AxmMotSetAbsRelMode로 설정한 이동 거리 계산 모드를 반환한다.

C Example

```
// 0축을 절대모드로 설정.
AxmMotSetAbsRelMode(0, POS_ABS_MODE);

// 0축의 이동 모드 확인.
```

```
DWORD Mode;
AxmMotGetAbsRelMode(0, &Mode);
```

VB Example

```
' 0축을 절대모드로 설정.
AxmMotSetAbsRelMode 0, POS_ABS_MODE

' 0축의 이동 모드 확인
Dim Mode As Long
AxmMotGetAbsRelMode 0, Mode
```

Delphi Example

```
{ 0축의 이동 모드 확인 }
var
Mode : DWord;

Begin
{ 0축을 절대모드로 설정. }
AxmMotSetAbsRelMode(0, POS_ABS_MODE);
AxmMotGetAbsRelMode (0, @Mode);
end;
```

See Also

AxmMotLoadParaAll, AxmMotSaveParaAll, AxmMotGetParaLoad, AxmMotSetParaLoad, AxmMotSetMaxVel, AxmMotGetMaxVel, AxmMotSetAbsRelMode, AxmMotSetProfileMode, AxmMotGetProfileMode, AxmMotSetAccelUnit, AxmMotGetAccelUnit, AxmMotSetMinVel, AxmMotGetMinVel, AxmMotSetAccelJerk, AxmMotGetAccelJerk, AxmMotSetDecelJerk, AxmMotGetDecelJerk

AxmMotSetProfileMode

Purpose

지정 축의 구동 속도 프로파일을 설정한다.

Format

C

```
DWORD AxmMotSetProfileMode (long lAxisNo, DWORD uProfileMode);
```

Visual Basic

```
Function AxmMotSetProfileMode (ByVal lAxisNo As Long, ByVal uProfileMode As Long ) As Long
```

Delphi

```
function AxmMotSetProfileMode (lAxisNo : LongInt; uProfileMode : DWord) : DWord; stdcall;
```

Input/Output

Name	in/out	Format	Init Value	Explanation
AxisNo	in	long	-	축번호(0 ~ (최대축수 - 1))
ProfileMode	In	DWORD	4	속도 프로파일 모드

ProfileMode

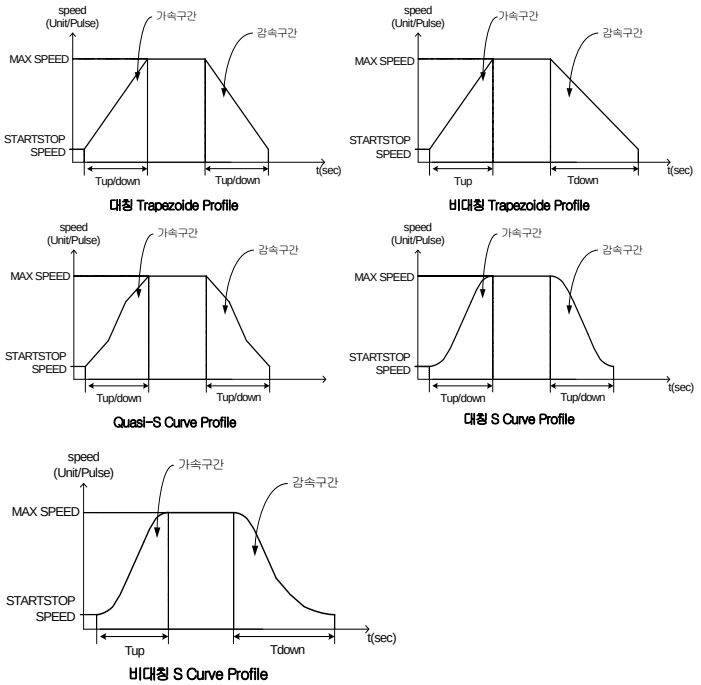
#define	Value	Explanation
SYM_TRAPEZOIDE_MODE	00h	대칭 Trapezode
ASYM_TRAPEZOIDE_MODE	01h	비대칭 Trapezode
QUASI_S_CURVE_MODE	02h	대칭 Quasi-S Curve(지원 안함)
SYM_S_CURVE_MODE	03h	대칭 S Curve
ASYM_S_CURVE_MODE	04h	비대칭 S Curve

Return AXT_RT_SUCCESS(0000) : Successful execution of API

* See error code Table more information on status error codes

Description

구동 속도 프로파일은 대칭 Trapezoid 모드, 비대칭 Trapezoid 모드, 대칭 S Curve 모드, 비대칭 S Curve 모드가 있다. 각 속도 프로파일 모드의 그래프는 아래 그림과 같다.



C Example

```
// 대칭 S-Curve로 0축의 속도 프로파일 설정.
AxmMotSetProfileMode (0, 3);

// 0축의 속도 프로파일 확인.
DWORD uProfile;
AxmMotGetProfileMode (0, &uProfile);
```

VB Example

```
'대칭 S-Curve로 0축의 속도 프로파일 설정.
AxmMotSetProfileMode 0, 3

' 0 축의 속도 프로파일 확인.
Dim uProfile As Long
AxmMotGetProfileMode 0, uProfile
```

Delphi Example

```
{ 0축의 속도 프로파일 확인. }
var
uProfile : DWord;
```

```
begin
{ 대칭 S-Curve로 0축의 속도 프로파일 설정. }
AxmMotSetProfileMode (0, 3);
AxmMotGetProfileMode (0, @uProfile);
end;
```

See Also

AxmMotLoadParaAll, AxmMotSaveParaAll, AxmMotGetParaLoad, AxmMotSetParaLoad, AxmMotSetMaxVel, AxmMotGetMaxVel, AxmMotSetAbsRelMode, AxmMotGetAbsRelMode, AxmMotGetProfileMode, AxmMotSetAccelUnit, AxmMotGetAccelUnit, AxmMotSetMinVel, AxmMotGetMinVel, AxmMotSetAccelJerk, AxmMotGetAccelJerk, AxmMotSetDecelJerk, AxmMotGetDecelJerk

AxmMotGetProfileMode

Purpose

지정 축의 설정된 구동 속도 프로파일을 반환한다.

Format

C

```
DWORD AxmMotGetProfileMode (long lAxisNo, DWORD *upProfileMode);
```

Visual Basic

```
Function AxmMotGetProfileMode (ByVal lAxisNo As Long, ByRef upProfileMode As Long) As Long
```

Delphi

```
function AxmMotGetProfileMode (lAxisNo : LongInt; upProfileMode : PDWord) : DWord; stdcall;
```

Input/Output

Name	in/out	Format	Init Value	Explanation
AxisNo	in	long	-	축번호(0 ~ (최대축수 - 1))
ProfileMode	out	DWORD*	4	속도 프로파일 모드

ProfileMode

#define	Value	Explanation
SYM_TRAPEZOIDE_MODE	00h	대칭 Trapezode
ASYM_TRAPEZOIDE_MODE	01h	비대칭 Trapezode
QUASI_S_CURVE_MODE	02h	대칭 Quasi-S Curve(지원 안함)
SYM_S_CURVE_MODE	03h	대칭 S Curve
ASYM_S_CURVE_MODE	04h	비대칭 S Curve

Return AXT_RT_SUCCESS(0000) : Successful execution of API

* See error code Table more information on status error codes

Description

AxmMotSetProfileMode 로 설정한 지정 축의 구동 속도 프로파일을 반환한다.

C Example

```
// 대칭 S-Curve로 0축의 속도 프로파일 설정.
AxmMotSetProfileMode (0, 3);

// 0축의 속도 프로파일 확인.
DWORD uProfile;
AxmMotGetProfileMode (0, &uProfile);
```

VB Example

```
'대칭 S-Curve로 0축의 속도 프로파일 설정.
AxmMotSetProfileMode 0, 3

' 0 축의 속도 프로파일 확인.
Dim uProfile As Long
AxmMotGetProfileMode 0, uProfile
```

Delphi Example

```
{ 0축의 속도 프로파일 확인. }
var
uProfile : DWord;

begin
{ 대칭 S-Curve로 0축의 속도 프로파일 설정. }
AxmMotSetProfileMode (0, 3);
AxmMotGetProfileMode (0, @uProfile);
end;
```

See Also

AxmMotLoadParaAll, AxmMotSaveParaAll, AxmMotGetParaLoad, AxmMotSetParaLoad, AxmMotSetMaxVel, AxmMotGetMaxVel, AxmMotSetAbsRelMode, AxmMotGetAbsRelMode, AxmMotSetProfileMode, AxmMotSetAccelUnit, AxmMotGetAccelUnit, AxmMotSetMinVel, AxmMotGetMinVel, AxmMotSetAccelJerk, AxmMotGetAccelJerk, AxmMotSetDecelJerk, AxmMotGetDecelJerk

AxmMotSetAccelUnit

Purpose

지정 축의 가속도단위를 설정한다.

Format

C

DWORD AxmMotSetAccelUnit (long lAxisNo, DWORD uAccelUnit);

Visual Basic

Function AxmMotSetAccelUnit (ByVal lAxisNo As Long, ByVal uAccelUnit As Long) As Long

Delphi

function AxmMotSetAccelUnit (lAxisNo : LongInt; uAccelUnit : DWord) : DWord; stdcall;

Input/Output

Name	in/out	Format	Init Value	Explanation
AxisNo	in	long	-	축번호(0 ~ (최대축수 - 1))
AccelUnit	in	DWORD	400	가속도 (가속도의 단위는 Unit/pulse를 1/1로 한경우에 PPS[Pulses/sec^2])

AccelUnit

#define	Value	Explanation
UNIT_SEC2	00h	unit/sec^2
SEC	01h	sec

Return AXT_RT_SUCCESS(0000) : Successful execution of API

* See error code Table more information on status error codes

Description

지정 축의 가속도 는 이동 속도의 가속율을 바탕으로 계산하는 unit/sec^2 (0) 단위로 설정할 수 있거나, 가속 시간 sec (1)을 설정할 수 있다.

C Example

```
// 0축의 가속도단위 unit/sec² 설정.
AxmMotSetAccelUnit (0, UNIT_SEC2);
```

```
// 0축의 가속도 단위 확인.
DWORD uAccelUnit;
AxmMotGetAccelUnit (0, &uAccelUnit);
```

VB Example

```
' 0축의 가속도단위 unit/sec² 설정.
AxmMotSetAccelUnit 0, UNIT_SEC2

' 0축의 가속도 단위 확인.
Dim uAccelUnit As Long
AxmMotGetAccelUnit 0, uAccelUnit
```

Delphi Example

```
{ 0축의 가속도 단위 확인. }
var
uAccelUnit: DWord;

begin
{ 0축의 가속도단위 unit/sec² 설정. }
AxmMotSetAccelUnit (0, UNIT_SEC2);

AxmMotGetAccelUnit (0, @uAccelUnit);
end;
```

See Also

AxmMotLoadParaAll, AxmMotSaveParaAll, AxmMotGetParaLoad, AxmMotSetParaLoad, AxmMotSetMaxVel, AxmMotGetMaxVel, AxmMotSetAbsRelMode, AxmMotGetAbsRelMode, AxmMotSetProfileMode, AxmMotGetProfileMode, AxmMotGetAccelUnit, AxmMotSetMinVl, AxmMotGetMinVel, AxmMotSetAccelJerk, AxmMotGetAccelJerk, AxmMotSetDecelJerk, AxmMotGetDecelJerk

AxmMotGetAccelUnit

Purpose

지정 축의 설정한 가속도 단위를 반환한다.

Format

C

DWORD AxmMotGetAccelUnit (long lAxisNo, DWORD *upAccelUnit);

Visual Basic

Function AxmMotGetAccelUnit (ByVal lAxisNo As Long, ByRef upAccelUnit As Long) As Long

Delphi

function AxmMotGetAccelUnit (lAxisNo : LongInt; upAccelUnit : PDWord) : DWord; stdcall;

Input/Output

Name	in/out	Format	Init Value	Explanation
AxisNo	in	long	-	축번호(0 ~ (최대축수 - 1))
AccelUnit	out	DWORD*	400	가속도 (가속도의 단위는 Unit/pulse를 1/1 로 한경우에 PPS[Pulses/sec^2])

AccelUnit

#define	Value	Explanation
UNIT_SEC2	00h	unit/sec^2
SEC	01h	sec

Return AXT_RT_SUCCESS(0000) : Successful execution of API

* See error code Table more information on status error codes

Description

AxmMotSetAccelUnit 로 설정한 지정 축의 가속도 단위를 반환한다.

C Example

```
// 0축의 가속도단위 unit/sec² 설정.
AxmMotSetAccelUnit (0, UNIT_SEC2);

// 0축의 가속도 단위 확인.
```

```
DWORD uAccelUnit;
AxmMotGetAccelUnit (0, &uAccelUnit);
```

VB Example

```
' 0축의 가속도단위 unit/sec² 설정.
AxmMotSetAccelUnit 0, UNIT_SEC2

' 0축의 가속도 단위 확인.
Dim uAccelUnit As Long
AxmMotGetAccelUnit 0, uAccelUnit
```

Delphi Example

```
{ 0축의 가속도 단위 확인. }
var
uAccelUnit: DWord;

begin
{ 0축의 가속도단위 unit/sec² 설정. }
AxmMotSetAccelUnit (0, UNIT_SEC2);

AxmMotGetAccelUnit (0, @uAccelUnit);
end;
```

See Also

AxmMotLoadParaAll, AxmMotSaveParaAll, AxmMotGetParaLoad, AxmMotSetParaLoad, AxmMotSetMaxVel, AxmMotGetMaxVel, AxmMotSetAbsRelMode, AxmMotGetAbsRelMode, AxmMotSetProfileMode, AxmMotGetProfileMode, AxmMotSetAccelUnit, AxmMotSetMinVel, AxmMotGetMinVel, AxmMotSetAccelJerk, AxmMotGetAccelJerk, AxmMotSetDecelJerk, AxmMotGetDecelJerk

AxmMotSetMinVel

Purpose

지정 축에 초기 속도를 설정한다.

Format

C

```
DWORD AxmMotSetMinVel (long lAxisNo, double dMinVelocity);
```

Visual Basic

```
Function AxmMotSetMinVel (ByVal lAxisNo As Long, ByVal dMinVelocity As Doubel) As Long
```

Delphi

```
function AxmMotSetMinVel (lAxisNo : LongInt; dMinVelocity : Double): DWord; stdcall;
```

Input/Output

Name	in/out	Format	Init Value	Explanation
AxisNo	in	long	-	축번호(0 ~ (최대축수 - 1))
MinVelocity	In	double	1	초기 속도 (초기속도의 단위는 Unit/pulse를 1/1로 한경우에 PPS[Pulses/sec])

Return AXT_RT_SUCCESS(0000) : Successful execution of API

* See error code Table more information on status error codes

Description

초기속도는 가속의 시작속도와 감속의 종료속도를 의미한다. 초기속도가 작업속도보다 높게 설정되면 기존에 설정된 초기속도로부터 출발하여 작업속도까지 도달하며 그이후의 이동은 초기 설정 속도로 가속없이 동작한다. 특별한 경우가 아니면 이 함수를 사용할 필요는 없지만 기본적으로 초기속도는 1로 설정된다. 스텝모터를 사용할 경우 기동 초기속도를 설정하여 탈조 현상을 없앨 수 있다.

주의사항: 최소속도를 UNIT/PULSE 보다 작게할 경우 최소단위가 UNIT/PULSE로 맞추어지기 때문에 최소 속도가 UNIT/PULSE 가 된다

C Example

```
// 0축에 초기 속도를 1로 설정
AxmMotSetMinVel (0, 1);
```

```
// 0축에 초기 속도를 확인
double MinVelocity;
AxmMotGetMinVel (0, &MinVelocity);
```

VB Example

```
' 0축에 초기 속도를 1로 설정
AxmMotSetMinVel 0, 1

' 0축에 초기 속도를 확인
Dim MinVelocity As Double
AxmMotGetMinVel 0, MinVelocity
```

Delphi Example

```
{ 0축에 초기 속도를 확인 }
var
MinVelocity: Double;

begin
{ 0축에 초기 속도를 1로 설정 }
AxmMotSetMinVel (0, 1);

AxmMotGetMinVel (0, @MinVelocity)
end;
```

See Also

AxmMotLoadParaAll, AxmMotSaveParaAll, AxmMotGetParaLoad, AxmMotSetParaLoad, AxmMotSetMaxVel, AxmMotGetMaxVel, AxmMotSetAbsRelMode, AxmMotGetAbsRelMode, AxmMotSetProfileMode, AxmMotGetProfileMode, AxmMotSetAccelUnit, AxmMotGetAccelUnit, AxmMotGetMinVel, AxmMotSetAccelJerk, AxmMotGetAccelJerk, AxmMotSetDecelJerk, AxmMotGetDecelJerk

AxmMotGetMinVel

Purpose

지정 축의 초기 속도를 반환한다.

Format

C

DWORD AxmMotGetMinVel (long lAxisNo, double *dpMinVelocity);

Visual Basic

Function AxmMotGetMinVel (ByVal lAxisNo As Long, ByRef dpMinVelocity As Double) As Long

Delphi

function AxmMotGetMinVel (lAxisNo : LongInt; dpMinVelocity : PDouble) : DWord; stdcall;

Input/Output

Name	in/out	Format	Init Value	Explanation
AxisNo	in	long	-	축번호(0 ~ (최대축수 - 1))
MinVelocity	out	double*	1	초기 속도 (초기속도의 단위는 Unit/pulse를 1/1로 한경우에 PPS[Pulses/sec])

Return AXT_RT_SUCCESS(0000) : Successful execution of API

* See error code Table more information on status error codes

Description

AxmMotSetMinVel 설정한 지정 축의 초기 속도를 반환한다.

C Example

```
// 0축에 초기 속도를 1로 설정
AxmMotSetMinVel (0, 1);

// 0축에 초기 속도를 확인
double MinVelocity;
AxmMotGetMinVel (0, &MinVelocity);
```

VB Example

```
' 0축에 초기 속도를 1로 설정
AxmMotSetMinVel 0, 1

' 0축에 초기 속도를 확인
Dim MinVelocity As Double
AxmMotGetMinVel 0, MinVelocity
```

Delphi Example

```
{ 0축에 초기 속도를 확인 }
var
MinVelocity: Double;

begin
{ 0축에 초기 속도를 1로 설정 }
AxmMotSetMinVel (0, 1);

AxmMotGetMinVel (0, @MinVelocity)
end;
```

See Also

AxmMotLoadParaAll, AxmMotSaveParaAll, AxmMotGetParaLoad, AxmMotSetParaLoad, AxmMotSetMaxVel, AxmMotGetMaxVel, AxmMotSetAbsRelMode, AxmMotGetAbsRelMode, AxmMotSetProfileMode, AxmMotGetProfileMode, AxmMotSetAccelUnit, AxmMotGetAccelUnit, AxmMotGetMinVel, AxmMotSetAccelJerk, AxmMotGetAccelJerk, AxmMotSetDecelJerk, AxmMotGetDecelJerk

AxmMotSetAccelJerk

Purpose

지정 축의 가속 저크값을 설정한다.

Format

C

DWORD AxmMotSetAccelJerk (long lAxisNo, double dAccelJerk);

Visual Basic

Function AxmMotSetAccelJerk (ByVal lAxisNo As Long, ByVal dAccelJerk As Double) As Long

Delphi

function AxmMotSetAccelJerk (lAxisNo : LongInt; dAccelJerk : Double) : DWord; stdcall;

Input/Output

Name	in/out	Format	Init Value	Explanation
AxisNo	in	long	-	축번호(0 ~ (최대축수 - 1))
AccelJerk	In	double	100	가속 저크값, 단위는 %

Return AXT_RT_SUCCESS(0000) : Successful execution of API

* See error code Table more information on status error codes

Description

단위는 %이다. S-Curve 속도 프로파일 모드로 구동할 경우 가속 시 최대 속도와 시작 속도 사이에서 S-Curve로 구동할 범위를 %로 설정한다.

C Example

```
// 0축의 저크값 설정
AxmMotSetAccelJerk (0, 2000);
AxmMotSetDecelJerk (0, 2000);

// 0의 저크값 확인
double dAccelJerk, dDecelJerk;
AxmMotGetAccelJerk (0, &dAccelJerk);
AxmMotGetDecelJerk (0, &dDecelJerk);
```

VB Example

```
' 0축의 저크값 설정
AxmMotSetAccelJerk 0, 2000
AxmMotSetDecelJerk 0, 2000

' 0의 저크값 확인
Dim dAccelJerk , dDecelJerk As Double
AxmMotGetAccelJerk 0, dAccelJerk
AxmMotGetDecelJerk 0, dDecelJerk
```

Delphi Example

```
{ 0의 저크값 확인 }
var
dAccelJerk, dDecelJerk: Double;

Begin
{ 0축의 저크값 설정 }
AxmMotSetAccelJerk (0, 2000);
AxmMotSetDecelJerk (0, 2000);

AxmMotGetAccelJerk (0, @dAccelJerk);
AxmMotGetDecelJerk (0, @dDecelJerk);
end;
```

See Also

AxmMotLoadParaAll, AxmMotSaveParaAll, AxmMotGetParaLoad, AxmMotSetParaLoad, AxmMotSetMaxVel, AxmMotGetMaxVel, AxmMotSetAbsRelMode, AxmMotGetAbsRelMode, AxmMotSetProfileMode, AxmMotGetProfileMode, AxmMotSetAccelUnit, AxmMotGetAccelUnit, AxmMotSetMinVel, AxmMotGetMinVel, AxmMotGetAccelJerk, AxmMotSetDecelJerk, AxmMotGetDecelJerk

AxmMotGetAccelJerk

Purpose

지정 축에 설정한 가속 저크값을 반환한다.

Format

C

DWORD AxmMotGetAccelJerk (long lAxisNo, double *dpAccelJerk);

Visual Basic

Function AxmMotGetAccelJerk (ByVal lAxisNo As Long, ByRef dpAccelJerk As Double) As Long

Delphi

function AxmMotGetAccelJerk (lAxisNo : LongInt; dpAccelJerk : PDouble) : DWord; stdcall;

Input/Output

Name	in/out	Format	Init Value	Explanation
AxisNo	in	long	-	축번호(0 ~ (최대축수 - 1))
AccelJerk	out	double*	100	가속 저크값, 단위는 %

Return AXT_RT_SUCCESS(0000) : Successful execution of API

* See error code Table more information on status error codes

Description

단위는 %이다. S-Curve 속도 프로파일 모드로 구동할 경우 가속 시 최대 속도와 시작 속도 사이에서 S-Curve로 구동할 범위를 %로 설정한다.

C Example

```
// 0축의 저크값 설정
AxmMotSetAccelJerk (0, 2000);
AxmMotSetDecelJerk (0, 2000);

// 0의 저크값 확인
double dAccelJerk, dDecelJerk;
AxmMotGetAccelJerk (0, &dAccelJerk);
AxmMotGetDecelJerk (0, &dDecelJerk);
```

VB Example

```
' 0축의 저크값 설정
AxmMotSetAccelJerk 0, 2000
AxmMotSetDecelJerk 0, 2000

' 0의 저크값 확인
Dim dAccelJerk , dDecelJerk As Double
AxmMotGetAccelJerk 0, dAccelJerk
AxmMotGetDecelJerk 0, dDecelJerk
```

Delphi Example

```
{ 0의 저크값 확인 }
var
dAccelJerk, dDecelJerk: Double;

Begin
{ 0축의 저크값 설정 }
AxmMotSetAccelJerk (0, 2000);
AxmMotSetDecelJerk (0, 2000);

AxmMotGetAccelJerk (0, @dAccelJerk);
AxmMotGetDecelJerk (0, @dDecelJerk);
end;
```

See Also

AxmMotLoadParaAll, AxmMotSaveParaAll, AxmMotGetParaLoad, AxmMotSetParaLoad, AxmMotSetMaxVel, AxmMotGetMaxVel, AxmMotSetAbsRelMode, AxmMotGetAbsRelMode, AxmMotSetProfileMode, AxmMotGetProfileMode, AxmMotSetAccelUnit, AxmMotGetAccelUnit, AxmMotSetMinVel, AxmMotGetMinVel, AxmMotSetAccelJerk, AxmMotSetDecelJerk, AxmMotGetDecelJerk

AxmMotSetDecelJerk

Purpose

지정 축의 감속 저크값을 설정한다.

Format

C

DWORD AxmMotSetDecelJerk (long lAxisNo, double dDecelJerk);

Visual Basic

Function AxmMotSetDecelJerk (ByVal lAxisNo As Long, ByVal dDecelJerk As Double) As Long

Delphi

function AxmMotSetDecelJerk (lAxisNo : LongInt; dDecelJerk : Double) : DWord; stdcall;

Input/Output

Name	in/out	Format	Init Value	Explanation
AxisNo	in	long	-	채널(축) 번호(0부터 시작함)
DecelJerk	In	double	100	감속속 저크값, 단위는 %

Return AXT_RT_SUCCESS(0000) : Successful execution of API

* See error code Table more information on status error codes

Description

단위는 %이다. S-Curve 속도 프로파일 모드로 구동할 경우 감속 시 최대 속도와 시작 속도 사이에서 S-Curve로 구동할 범위를 %로 설정한다.

C Example

```
// 0축의 저크값 설정
AxmMotSetAccelJerk(0, 2000);
AxmMotSetDecelJerk(0, 2000);

// 0의 저크값 확인
double dAccelJerk, dDecelJerk;
AxmMotGetAccelJerk (0, &dAccelJerk);
AxmMotGetDecelJerk (0, &dDecelJerk);
```

VB Example

```
' 0축의 저크값 설정
AxmMotSetAccelJerk 0, 2000
AxmMotSetDecelJerk 0, 2000

' 0의 저크값 확인
Dim dAccelJerk , dDecelJerk As Double
AxmMotGetAccelJerk 0, dAccelJerk
AxmMotGetDecelJerk 0, dDecelJerk
```

Delphi Example

```
{ 0의 저크값 확인 }
var
dAccelJerk, dDecelJerk: Double;

begin
{ 0축의 저크값 설정 }
AxmMotSetAccelJerk(0, 2000);
AxmMotSetDecelJerk(0, 2000);

AxmMotGetAccelJerk (0, @dAccelJerk);
AxmMotGetDecelJerk (0, @dDecelJerk);
end;
```

See Also

AxmMotLoadParaAll, AxmMotSaveParaAll, AxmMotGetParaLoad, AxmMotSetParaLoad, AxmMotSetMaxVel, AxmMotGetMaxVel, AxmMotSetAbsRelMode, AxmMotGetAbsRelMode, AxmMotSetProfileMode, AxmMotGetProfileMode, AxmMotSetAccelUnit, AxmMotGetAccelUnit, AxmMotSetMinVel, AxmMotGetMinVel, AxmMotSetAccelJerk, AxmMotGetAccelJerk, AxmMotGetDecelJerk

AxmMotGetDecelJerk

Purpose

지정 축에 설정한 감속 저크값을 반환한다.

Format

C

DWORD AxmMotGetDecelJerk (long lAxisNo, double *dpDecelJerk);

Visual Basic

Function AxmMotGetDecelJerk (ByVal lAxisNo As Long, ByRef dpDecelJerk As Double) As Long

Delphi

function AxmMotGetDecelJerk(lAxisNo : LongInt; dpDecelJerk : PDouble) : DWord; stdcall;

Input/Output

Name	in/out	Format	Init Value	Explanation
AxisNo	in	long	-	축번호(0 ~ (최대축수 - 1))
DecelJerk	out	double*	100	감속속 저크값, 단위는 %

Return AXT_RT_SUCCESS(0000) : Successful execution of API

* See error code Table more information on status error codes

Description

단위는 %이다. S-Curve 속도 프로파일 모드로 구동할 경우 감속 시 최대 속도와 시작 속도 사이에서 S-Curve로 구동할 범위를 %로 설정한다.

C Example

```
// 0축의 저크값 설정
AxmMotSetAccelJerk(0, 2000);
AxmMotSetDecelJerk(0, 2000);

// 0의 저크값 확인
double dAccelJerk, dDecelJerk;
AxmMotGetAccelJerk (0, &dAccelJerk);
AxmMotGetDecelJerk (0, &dDecelJerk);
```

VB Example

```
' 0축의 저크값 설정
AxmMotSetAccelJerk 0, 2000
AxmMotSetDecelJerk 0, 2000

' 0의 저크값 확인
Dim dAccelJerk , dDecelJerk As Double
AxmMotGetAccelJerk 0, dAccelJerk
AxmMotGetDecelJerk 0, dDecelJerk
```

Delphi Example

```
{ 0의 저크값 확인 }
var
dAccelJerk, dDecelJerk: Double;

begin
{ 0축의 저크값 설정 }
AxmMotSetAccelJerk(0, 2000);
AxmMotSetDecelJerk(0, 2000);

AxmMotGetAccelJerk (0, @dAccelJerk);
AxmMotGetDecelJerk (0, @dDecelJerk);
end;
```

See Also

AxmMotLoadParaAll, AxmMotSaveParaAll, AxmMotGetParaLoad, AxmMotSetParaLoad, AxmMotSetMaxVel, AxmMotGetMaxVel, AxmMotSetAbsRelMode, AxmMotGetAbsRelMode, AxmMotSetProfileMode, AxmMotGetProfileMode, AxmMotSetAccelUnit, AxmMotGetAccelUnit, AxmMotSetMinVel, AxmMotGetMinVel, AxmMotSetAccelJerk, AxmMotGetAccelJerk, AxmMotSetDecelJerk,

입출력 레벨 설정 함수

이 단원에서는 외부 센서들에 의한 레벨 및 범용 입출력, 서보드라이브에 관련된 레벨함수들을 소개한다. 레벨을 잘못 맞추어졌을 경우 모터가 동작을 안할 수 있으므로 레벨을 제대로 맞추고 모션 구동을 시작한다.

Function	Description
AxmSignalSetZphaseLevel	지정 축의 Z 상 Level을 설정한다.
AxmSignalGetZphaseLevel	지정 축의 Z 상 Level을 반환한다.
AxmSignalSetServoOnLevel	지정 축의 Amp-On신호의 출력 레벨을 설정한다.
AxmSignalGetServoOnLevel	지정 축의 Amp-On신호의 출력 레벨 설정을 반환한다.
AxmSignalSetServoAlarmResetLevel	지정 축의 Servo-Alarm Reset 신호의 출력 레벨을 설정한다.
AxmSignalGetServoAlarmResetLevel	지정 축의 Servo-Alarm Reset 신호의 출력 레벨을 설정을 반환한다.
AxmSignalSetInpos	지정 축의 InPos 신호 사용 여부 및 신호 입력 레벨을 설정한다
AxmSignalGetInpos	지정 축의 InPos 신호 사용 여부 및 신호 입력 레벨을 반환한다.
AxmSignalReadInpos	지정 축의 Inpositon 신호 입력 상태를 반환한다.
AxmSignalSetServoAlarm	지정 축의 알람 신호 입력 시 비상 정지의 사용 여부 및 신호 입력 레벨을 설정한다.
AxmSignalGetServoAlarm	지정 축의 알람 신호 입력 시 비상 정지의 사용 여부 및 신호 입력 레벨을 반환한다.
AxmSignalReadServoAlarm	지정 축의 알람 신호의 입력 레벨을 반환한다.
AxmSignalSetLimit	지정 축의 end limit sensor의 사용 유무 및 신호의 입력 레벨을 설정한다. end limit sensor 신호 입력 시 감속정지 또는 급정지에 대한 설정도 가능하다.

AxmSignalGetLimit	지정 축의 end limit sensor의 사용 유무 및 신호의 입력 레벨, 신호 입력 시 정지모드를 반환한다
AxmSignalReadLimit	지정축의 end limit sensor의 입력 상태를 반환한다.
AxmSignalSetSoftLimit	지정 축의 Software limit의 사용 유무, 사용할 카운트, 그리고 정지 방법을 설정한다
AxmSignalGetSoftLimit	지정 축의 Software limit의 사용 유무, 사용할 카운트, 그리고 정지 방법을 반환한다
AxmSignalSetStop	비상 정지 신호의 정지 방법 (급정지/감속정지) 또는 사용 유무를 설정한다.
AxmSignalGetStop	비상 정지 신호의 정지 방법 (급정지/감속정지) 또는 사용 유무를 반환한다.
AxmSignalReadStop	비상 정지 신호의 입력 상태를 반환한다.

AxmSignalSetZphaseLevel

Purpose

지정 축의 Z상 입력 신호의 출력 레벨을 설정한다.

Format

C

```
DWORD AxmSignalSetZphaseLevel (long lAxisNo, DWORD uLevel);
```

Visual Basic

```
Function AxmSignalSetZphaseLevel (ByVal lAxisNo As Long, ByVal uLevel As Long) As Long
```

Delphi

```
function AxmSignalSetZphaseLevel (lAxisNo : LongInt; uLevel : DWord) : DWord; stdcall;
```

Input / Output

Name	in/out	Format	Init Value	Explanation
AxisNo	in	long	–	축번호(0 ~ (최대축수 – 1))
Level	In	DWORD	1	Z상 신호 입력의 Active Level

Level

#define	Value	Explanation
LOW	00h	B 접점(Normal Close)
HIGH	01h	A 접점(Normal Open)

Return AXT_RT_SUCCESS(0000) : Successful execution of API

* See error code Table more information on status error codes

Description

사용자가 지정한 축의 Z상 입력 레벨을 설정한다.

Z상 신호는 엔코더의 제로 펄스 신호를 의미 한다. 범용 입력 1를 Encoder의 Z상 신호로 사용한다. 이 신호를 사용하면 ORG신호를 사용했을 때보다 정밀한 원점 모든 작업을 수행할 수 있도록 해준다.

Home Level을 읽기 위해서는 AxmStatusReadMechanical 함수를 사용하여 Bit 8을 읽으면 레벨이 바뀌는지 확인할 수 있다.

C Example

```
// 0 축에 Z상 레벨을 설정한다.
AxmSignalSetZphaseLevel(0, HIGH);

// 0축의 Z상 레벨 값을 확인
DWORD  state;

AxmSignalGetZphaseLevel (0, &state);
```

VB Example

```
' 0 축에 Z상 레벨을 설정한다.
AxmSignalSetZphaseLevel 0, HIGH

'0축의 Z상 레벨 값을 확인
Dim state As Long

AxmSignalGetZphaseLevel 0, state
```

Delphi Example

```
{ 0축의 Z상 레벨 값을 확인 }
var
state: DWord;

Begin
{ 0 축에 Z상 레벨을 설정한다. }
AxmSignalSetZphaseLevel(0, HIGH);
AxmSignalGetZphaseLevel(0, @state);
end;
```

See Also

AxmSignalGetZphaseLevel, AxmSignalSetServoOnLevel, AxmSignalGetServoOnLevel, AxmSignalSetServoAlarmResetLevel, AxmSignalGetServoAlarmResetLevel, AxmSignalSetInpos, AxmSignalGetInpos, AxmSignalReadInpos, AxmSignalSetServoAlarm, AxmSignalGetServoAlarm AxmSignalReadServoAlarm, AxmSignalSetLimit, AxmSignalGetLimit, AxmSignalReadLimit, AxmSignalSetSoftLimit, AxmSignalGetSoftLimit, AxmSignalSetStop, AxmSignalGetStop, AxmSignalReadStop

AxmSignalGetZphaseLevel

Purpose

지정 축의 Z상 신호의 입력 레벨 설정을 반환한다.

Format

C

```
DWORD AxmSignalGetZphaseLevel (long lAxisNo, DWORD *upLevel);
```

Visual Basic

```
Function AxmSignalGetZphaseLevel (ByVal lAxisNo As Long, ByRef upLevel As Long) As Long
```

Delphi

```
function AxmSignalGetZphaseLevel (lAxisNo : LongInt; upLevel : PDWord) : DWord; stdcall;
```

Input / Output

Name	in/out	Format	Init Value	Explanation
AxisNo	in	long	–	축번호(0 ~ (최대축수 – 1))
Level	out	DWORD*	1	Z상 신호 입력의 Active Level

Level

#define	Value	Explanation
LOW	00h	B 접점(Normal Close)
HIGH	01h	A 접점(Normal Open)

Return AXT_RT_SUCCESS(0000) : Successful execution of API

* See error code Table more information on status error codes

Description

'AxmSignalSetZphaseLevel' 로 설정한 Z상 입력 레벨을 반환한다.

C Example

```
// 0 축에 Z상 레벨을 설정한다.
AxmSignalSetZphaseLevel(0, HIGH);

// 0축의 Z상 레벨 값을 확인
DWORD  state;
```

```
AxmSignalGetZphaseLevel (0, &state);
```

VB Example

```
' 0 축에 Z상 레벨을 설정한다.
AxmSignalSetZphaseLevel 0, HIGH

'0축의 Z상 레벨 값을 확인
Dim state As Long

AxmSignalGetZphaseLevel 0, state
```

Delphi Example

```
{ 0축의 Z상 레벨 값을 확인 }
var
state: DWord;

Begin
{ 0 축에 Z상 레벨을 설정한다. }
AxmSignalSetZphaseLevel(0, HIGH);
AxmSignalGetZphaseLevel(0, @state);
end;
```

See Also

AxmSignalSetZphaseLevel, AxmSignalSetServoOnLevel, AxmSignalGetServoOnLevel, AxmSignalSetServoAlarmResetLevel, AxmSignalGetServoAlarmResetLevel, AxmSignalSetInpos, AxmSignalGetInpos, AxmSignalReadInpos, AxmSignalSetServoAlarm, AxmSignalGetServoAlarm, AxmSignalReadServoAlarm, AxmSignalSetLimit, AxmSignalGetLimit, AxmSignalReadLimit, AxmSignalSetSoftLimit, AxmSignalGetSoftLimit, AxmSignalSetStop, AxmSignalGetStop, AxmSignalReadStop

AxmSignalSetServoOnLevel

Purpose

지정 축의 Servo-On 신호의 출력 레벨을 설정한다.

Format

C

```
DWORD AxmSignalSetServoOnLevel (long lAxisNo, DWORD uLevel);
```

Visual Basic

```
Function AxmSignalSetServoOnLevel (ByVal lAxisNo As Long, ByVal uLevel As Long) As Long
```

Delphi

```
function AxmSignalSetServoOnLevel (lAxisNo : LongInt; uLevel : DWord) : DWord; stdcall;
```

Input / Output

Name	in/out	Format	Init Value	Explanation
AxisNo	in	long	-	축번호(0 ~ (최대축수 - 1))
Level	In	DWORD	1	Servo-On 신호의 Active Level

Level

#define	Value	Explanation
LOW	00h	B 접점(Normal Close)
HIGH	01h	A 접점(Normal Open)

Return AXT_RT_SUCCESS(0000) : Successful execution of API

* See error code Table more information on status error codes

Description

사용자가 지정한 축의 Servo-On신호의 출력 레벨을 설정한다. 각 회사의 Servo-Drive 의 입력 레벨을 다를 수 있으므로 레벨을 설정하는데 이용한다.

C Example

```
// 0 축에 Servo-On 레벨을 설정한다.
AxmSignalSetServoOnLevel(0, HIGH);

// 0축의 Servo-On 레벨 값을 확인
```

```
DWORD  state;

AxmSignalGetServoOnLevel (0, &state);
```

VB Example

```
'0 축에 Servo-On 레벨을 설정한다.
AxmSignalSetServoOnLevel 0, HIGH

'0축의 Servo-On 레벨 값을 확인
Dim state As Long

AxmSignalGetServoOnLevel 0, state
```

Delphi Example

```
{ 0축의 Servo-On 레벨 값을 확인 }
var
state: DWord;

Begin
{ 0 축에 Servo-On 레벨을 설정한다. }
AxmSignalSetServoOnLevel (0, HIGH);
AxmSignalGetServoOnLevel (0, @state);
end;
```

See Also

AxmSignalSetZphaseLevel, AxmSignalGetZphaseLevel, AxmSignalGetServoOnLevel, AxmSignalGetServoAlarmResetLevel, AxmSignalSetInpos, AxmSignalGetInpos, AxmSignalReadInpos, AxmSignalSetServoAlarm, AxmSignalGetServoAlarm, AxmSignalReadServoAlarm, AxmSignalSetLimit, AxmSignalGetLimit, AxmSignalReadLimit, AxmSignalSetSoftLimit, AxmSignalGetSoftLimit, AxmSignalSetStop, AxmSignalGetStop, AxmSignalReadStop

AxmSignalGetServoOnLevel

Purpose

지정 축의 Servo-On 신호의 출력 레벨 설정을 반환한다.

Format

C

```
DWORD AxmSignalGetServoOnLevel (long lAxisNo, DWORD *upLevel);
```

Visual Basic

```
Function AxmSignalGetServoOnLevel (ByVal lAxisNo As Long, ByRef upLevel As Long) As Long
```

Delphi

```
function AxmSignalGetServoOnLevel (lAxisNo : LongInt; upLevel : PDWord) : DWord; stdcall;
```

Input / Output

Name	in/out	Format	Init Value	Explanation
AxisNo	In	long	-	축번호(0 ~ (최대축수 - 1))
Level	out	DWORD*	1	Servo-On 신호의 Active Level

Level

#define	Value	Explanation
LOW	00h	B 접점(Normal Close)
HIGH	01h	A 접점(Normal Open)

Return AXT_RT_SUCCESS(0000) : Successful execution of API

* See error code Table more information on status error codes

Description

'AxmSignalSetServoOnLevel' 로 설정한 Servo-On 신호의 출력 레벨 설정을 반환한다.

C Example

```
// 0 축에 Servo-On 레벨을 설정한다.
AxmSignalSetServoOnLevel(0, HIGH);

// 0축의 Servo-On 레벨 값을 확인
DWORD  state;
```

```
AxmSignalGetServoOnLevel (0, &state);
```

VB Example

```
'0 축에 Servo-On 레벨을 설정한다.
AxmSignalSetServoOnLevel 0, HIGH

'0축의 Servo-On 레벨 값을 확인
Dim state As Long

AxmSignalGetServoOnLevel 0, state
```

Delphi Example

```
{ 0축의 Servo-On 레벨 값을 확인 }
var
state: DWord;

Begin
{ 0 축에 Servo-On 레벨을 설정한다. }
AxmSignalSetServoOnLevel (0, HIGH);
AxmSignalGetServoOnLevel (0, @state);
end;
```

See Also

AxmSignalSetZphaseLevel, AxmSignalGetZphaseLevel, AxmSignalSetServoOnLevel, AxmSignalSetServoAlarmResetLevel, AxmSignalGetServoAlarmResetLevel, AxmSignalSetInpos, AxmSignalGetInpos, AxmSignalReadInpos, AxmSignalSetServoAlarm, AxmSignalGetServoAlarm, AxmSignalReadServoAlarm, AxmSignalSetLimit, AxmSignalGetLimit, AxmSignalReadLimit, AxmSignalSetSoftLimit, AxmSignalGetSoftLimit, AxmSignalSetStop, AxmSignalGetStop, AxmSignalReadStop

AxmSignalSetServoAlarmResetLevel

Purpose

지정 축의 Servo-Alarm Reset 신호의 출력 레벨을 설정한다.

Format

C

```
DWORD AxmSignalSetServoAlarmResetLevel (long lAxisNo, DWORD uLevel);
```

Visual Basic

```
Function AxmSignalSetServoAlarmResetLevel (ByVal lAxisNo As Long, ByVal uLevel As Long) As Long
```

Delphi

```
function AxmSignalSetServoAlarmResetLevel (lAxisNo : LongInt; uLevel : DWord) : DWord; stdcall;
```

Input / Output

Name	in/out	Format	Init Value	Explanation
AxisNo	in	long	-	축번호(0 ~ (최대축수 - 1))
Level	In	DWORD	1	Servo-Alarm Reset 신호의 Active Level

Level

#define	Value	Explanation
LOW	00h	B 접점(Normal Close)
HIGH	01h	A 접점(Normal Open)

Return AXT_RT_SUCCESS(0000) : Successful execution of API

* See error code Table more information on status error codes

Description

모션 구동을 하면서 비정상적인 동작으로 인해 알람이 발생하거나, 기타 다른 사유로 인해 알람이 발생했을 때 ServoAlarm Reset을 출력 함으로써 알람을 Clear시킨다. 이때 각 회사마다 Servo-Drive의 입력레벨이 다를 수 있으므로 레벨을 설정할 때 사용한다.

C Example

```
// 0 축에 Servo-Alarm Reset 레벨을 설정한다.
AxmSignalSetServoAlarmResetLevel(0, HIGH);

// 0축의 Servo-Alarm Reset 레벨 값을 확인
DWORD  state;

AxmSignalGetServoAlarmResetLevel(0, &state);
```

VB Example

```
‘ 0 축에 Servo-Alarm Reset 레벨을 설정한다.
AxmSignalSetServoAlarmResetLevel 0, HIGH

‘0축의 Servo-Alarm Reset 레벨 값을 확인
Dim state As Long

AxmSignalGetServoAlarmResetLevel 0, state
```

Delphi Example

```
{ 0축의 Servo-Alarm Reset 레벨 값을 확인 }
var
state: DWord;

Begin
{ 0 축에 Servo-Alarm Reset 레벨을 설정한다. }
AxmSignalSetServoAlarmResetLevel(0, HIGH);
AxmSignalGetServoAlarmResetLevel(0, @state);
end;
```

See Also

AxmSignalSetZphaseLevel, AxmSignalGetZphaseLevel, AxmSignalSetServoOnLevel, AxmSignalGetServoOnLevel, AxmSignalGetServoAlarmResetLevel, AxmSignalSetInpos, AxmSignalGetInpos, AxmSignalReadInpos, AxmSignalSetServoAlarm, AxmSignalGetServoAlarm, AxmSignalReadServoAlarm, AxmSignalSetLimit, AxmSignalGetLimit, AxmSignalReadLimit, AxmSignalSetSoftLimit, AxmSignalGetSoftLimit, AxmSignalSetStop, AxmSignalGetStop, AxmSignalReadStop

AxmSignalGetServoAlarmResetLevel

Purpose

지정 축의 Servo-Alarm Reset 신호의 출력 레벨 설정을 반환한다.

Format

C

```
DWORD AxmSignalGetServoAlarmResetLevel (long lAxisNo, DWORD *upLevel);
```

Visual Basic

```
Function AxmSignalGetServoAlarmResetLevel (ByVal lAxisNo As Long, ByRef upLevel As Long) As Long
```

Delphi

```
function AxmSignalGetServoAlarmResetLevel (lAxisNo : LongInt; upLevel : PDWord) : DWord; stdcall;
```

Input / Output

Name	in/out	Format	Init Value	Explanation
AxisNo	in	long	-	축번호(0 ~ (최대축수 - 1))
Level	out	DWORD*	1	Servo-Alarm Reset 신호의 Active Level

Level

#define	Value	Explanation
LOW	00h	B 접점(Normal Close)
HIGH	01h	A 접점(Normal Open)

Return AXT_RT_SUCCESS(0000) : Successful execution of API

* See error code Table more information on status error codes

Description

'AxmSignalSetServoOnLevel' 로 설정한 Servo-Alarm Reset 신호의 출력 레벨 설정을 반환한다.

C Example

```
// 0 축에 Servo-Alarm Reset 레벨을 설정한다.
AxmSignalSetServoAlarmResetLevel(0, HIGH);
```

```
// 0축의 Servo-Alarm Reset 레벨 값을 확인
DWORD  state;

AxmSignalGetServoAlarmResetLevel(0, &state);
```

VB Example

```
' 0 축에 Servo-Alarm Reset 레벨을 설정한다.
AxmSignalSetServoAlarmResetLevel 0, HIGH

'0축의 Servo-Alarm Reset 레벨 값을 확인
Dim state As Long

AxmSignalGetServoAlarmResetLevel 0, state
```

Delphi Example

```
{ 0축의 Servo-Alarm Reset 레벨 값을 확인 }
var
state: DWord;

Begin
{ 0 축에 Servo-Alarm Reset 레벨을 설정한다. }
AxmSignalSetServoAlarmResetLevel(0, HIGH);
AxmSignalGetServoAlarmResetLevel(0, @state);
end;
```

See Also

AxmSignalSetZphaseLevel, AxmSignalGetZphaseLevel, AxmSignalSetServoOnLevel, AxmSignalGetServoOnLevel, AxmSignalSetServoAlarmResetLevel, AxmSignalSetInpos, AxmSignalGetInpos, AxmSignalReadInpos, AxmSignalSetServoAlarm, AxmSignalGetServoAlarm, AxmSignalReadServoAlarm, AxmSignalSetLimit, AxmSignalGetLimit, AxmSignalReadLimit, AxmSignalSetSoftLimit, AxmSignalGetSoftLimit, AxmSignalSetStop, AxmSignalGetStop, AxmSignalReadStop

AxmSignalSetInpos

Purpose

지정 축의 InPos 신호 사용 여부를 설정한다. Inpositon 신호는 서보드라이버를 사용하는 경우에만 유효한 신호로써 서보드라이브가 위치결정 이동을 완료했음을 알리는 신호이다. InPos의 범위는 서보드라이버에서 설정할 수 있으므로 InPos 기능을 활용하려면 서보드라이버에서 InPos범위를 적절한 값으로 설정주어야 한다.

Format

C

```
DWORD AxmSignalSetInpos (long lAxisNo, DWORD uUse);
```

Visual Basic

```
Function AxmSignalSetInpos (ByVal lAxisNo As Long, ByVal uUse As Long) As Long
```

Delphi

```
function AxmSignalSetInpos (lAxisNo : LongInt; uUse : Dword) : Dword; stdcall;
```

Input / Output

Name	in/out	Format	Init Value	Explanation
AxisNo	in	long	-	축번호(0 ~ (최대축수 - 1))
Use	In	DWORD	2	InPos 신호 사용 여부

Use

#define	Value	Explanation
LOW	00h	B 접점(NORMAL CLOSE)
HIGH	01h	A 접점(NORMAL OPEN)
UNUSED	02h	사용안함
USED	03h	현상태유지

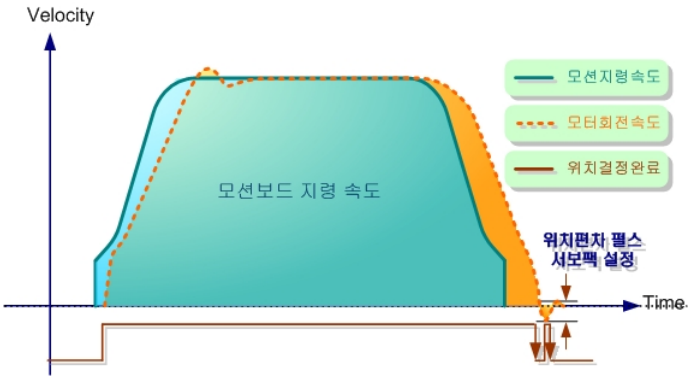
Return AXT_RT_SUCCESS(0000) : Successful execution of API

* See error code Table more information on status error codes

Description

InPos 신호를 사용할 때는 'ENABLE'로 설정하고 사용하지 않을 때에는 'DISABLE'로 설정한다. 'ENABLE'로 설정할 경우 InPos 신호의 Active Level은 이전에 설정된 레벨이 적용된다. InPos 신호의

Active Level을 설정하고자 할 경우는 'LOW' 또는 'HIGH'로 설정한다.
이 함수를 호출하면 모션 종료 시 inPos 신호(서보팩의 출력)가 들어왔을 때 구동이 끝난 것으로 인식된다. 즉, 함수 내에서 루프를 돌고 있는 move함수들은 inPos이 active될 때까지 그 함수 내에 머무르게 된다. InPos 신호란 서보 모터 드라이버 내부에 있는 명령 펄스 입력과 모터로부터 받은 피드백 펄스 입력의 차가 서보 드라이버에 설정한 오차 범위 내에 들어왔을 때 발생하는 신호이다. 따라서 이 신호를 위치 결정 완료 신호로 사용할 수 있다.
InPos 신호의 입력 확인은 AxmSignalReadInpos 함수로 반환한다.



참고로 스텝모터를 사용하는 경우나 InPos 신호를 연결하지 않은 경우에는 반드시 사용하지 않는 게 좋다.

C Example

```
// 0축에 InPos 신호 사용 여부를 설정
AxmSignalSetInpos (0, HIGH);

// 0 축의 InPos 신호의 사용 여부를 확인
DWORD    use;

AxmSignalGetInpos (0, &use);
```

VB Example

```
' 0축에 InPos 신호 사용 여부를 설정
AxmSignalSetInpos 0, HIGH

' 0 축의 InPos 신호의 사용 여부를 확인
Dim use As Long

AxmSignalGetInpos 0, use
```

Delphi Example

```
{ 0 축의 InPos 신호의 사용 여부를 확인 }
var
use: DWord;
```

```
Begin
{ 0축에 InPos 신호 사용 여부를 설정 }
AxmSignalSetInpos (0, HIGH);
AxmSignalGetInpos (0, @use);
end;
```

See Also

AxmSignalSetZphaseLevel, AxmSignalGetZphaseLevel, AxmSignalSetServoOnLevel, AxmSignalGetServoOnLevel, AxmSignalSetServoAlarmResetLevel, AxmSignalGetServoAlarmResetLevel, AxmSignalGetInpos, AxmSignalReadInpos, AxmSignalSetServoAlarm, AxmSignalGetServoAlarm, AxmSignalReadServoAlarm, AxmSignalSetLimit, AxmSignalGetLimit, AxmSignalReadLimit, AxmSignalSetSoftLimit, AxmSignalGetSoftLimit, AxmSignalSetStop, AxmSignalGetStop, AxmSignalReadStop

AxmSignalGetInpos

Purpose

지정 축의 InPos 신호의 사용 여부를 반환한다.

Format

C

```
DWORD AxmSignalGetInpos (long lAxisNo, DWORD *upUse);
```

Visual Basic

```
Function AxmSignalGetInpos (ByVal lAxisNo As Long, ByRef upUse As Long) As Long
```

Delphi

```
function AxmSignalGetInpos (lAxisNo : LongInt; upUse : PDword) : Dword; stdcall;
```

Input / Output

Name	in/out	Format	Init Value	Explanation
AxisNo	in	long	–	축번호(0 ~ (최대축수 – 1))
Use	out	DWORD*	2	InPos 신호 사용 여부

Use

#define	Value	Explanation
LOW	00h	B 접점(NORMAL CLOSE)
HIGH	01h	A 접점(NORMAL OPEN)
UNUSED	02h	사용안함
USED	03h	현상태유지

Return AXT_RT_SUCCESS(0000) : Successful execution of API

* See error code Table more information on status error codes

Description

' AxmSignalSetInpos ' 로 설정한 InPos 신호의 Level상태를 반환한다.
사용자가 지정한 축의 InPos 신호가 'ENABLE' 또는 'DISABLE'인지 사용 여부를 반환한다.
InPos 신호를 사용할 경우 InPos 신호의 Active Level을 확인할 수 있다.

C Example

```
// 0축에 InPos 신호 사용 여부를 설정
AxmSignalSetInpos (0, HIGH);

// 0 축의 InPos 신호의 사용 여부를 확인
DWORD    use;

AxmSignalGetInpos (0, &use);
```

VB Example

```
' 0축에 InPos 신호 사용 여부를 설정
AxmSignalSetInpos 0, HIGH

' 0 축의 InPos 신호의 사용 여부를 확인
Dim use As Long

AxmSignalGetInpos 0, use
```

Delphi Example

```
{ 0 축의 InPos 신호의 사용 여부를 확인 }
var
use: DWord;

Begin
{ 0축에 InPos 신호 사용 여부를 설정 }
AxmSignalSetInpos (0, HIGH);
AxmSignalGetInpos (0, @use);
end;
```

See Also

AxmSignalSetZphaseLevel, AxmSignalGetZphaseLevel, AxmSignalSetServoOnLevel, AxmSignalGetServoOnLevel, AxmSignalSetServoAlarmResetLevel, AxmSignalGetServoAlarmResetLevel, AxmSignalSetInpos, AxmSignalReadInpos, AxmSignalSetServoAlarm, AxmSignalGetServoAlarm, AxmSignalReadServoAlarm, AxmSignalSetLimit, AxmSignalGetLimit, AxmSignalReadLimit, AxmSignalSetSoftLimit, AxmSignalGetSoftLimit, AxmSignalSetStop, AxmSignalGetStop, AxmSignalReadStop

AxmSignalReadInpos

Purpose

지정 축의 Inpositon 신호의 입력 상태를 반환한다.

Format

C

```
DWORD AxmSignalReadInpos (long lAxisNo, DWORD *upStatus);
```

Visual Basic

```
Function AxmSignalReadInpos (ByVal lAxisNo As Long, ByRef upStatus As Long) As Long
```

Delphi

```
function AxmSignalReadInpos (lAxisNo : LongInt; upStatus : PDword) : Dword; stdcall;
```

Input / Output

Name	in/out	Format	Init Value	Explanation
AxisNo	in	long	-	축번호(0 ~ (최대축수 - 1))
Status	out	DWORD*	-	InPos 신호의 입력 상태

Status

#define	Value	Explanation
INACTIVE	00h	비활성화
ACTIVE	01h	활성화

Return AXT_RT_SUCCESS(0000) : Successful execution of API

* See error code Table more information on status error codes

Description

' AxmSignalSetInpos ' 로 설정한 현재 InPos 신호의 입력상태를 반환한다.

C Example

```
// 0축에 Inpositon 신호의 입력 확인
DWORD upStatus;
AxmSignalReadInpos (0, &upStatus);
```

VB Example

```
' 0축에 Inpositon 신호의 입력 레벨을 설정
Dim upStatus As Long
AxmSignalReadInpos 0, upStatus
```

Delphi Example

```
{ 0축에 Inpositon 신호의 입력 레벨을 설정 }
var
upStatus: DWord;

begin
AxmSignalReadInpos (0, @upStatus);
end;
```

See Also

AxmSignalSetZphaseLevel, AxmSignalGetZphaseLevel, AxmSignalSetServoOnLevel, AxmSignalGetServoOnLevel, AxmSignalSetServoAlarmResetLevel, AxmSignalGetServoAlarmResetLevel, AxmSignalSetInpos, AxmSignalGetInpos, AxmSignalSetServoAlarm, AxmSignalGetServoAlarm, AxmSignalReadServoAlarm, AxmSignalSetLimit, AxmSignalGetLimit, AxmSignalReadLimit, AxmSignalSetSoftLimit, AxmSignalGetSoftLimit, AxmSignalSetStop, AxmSignalGetStop, AxmSignalReadStop

AxmSignalSetServoAlarm

Purpose

지정 축의 알람 신호 입력 시 비상 정지의 사용 여부 및 알람 신호 Active Level을 설정한다.

Format

C

```
DWORD AxmSignalSetServoAlarm (long lAxisNo, DWORD uUse);
```

Visual Basic

```
Function AxmSignalSetServoAlarm (ByVal lAxisNo As Long, ByVal uUse As Long) As Long
```

Delphi

```
function AxmSignalSetServoAlarm (lAxisNo : LongInt; uUse : Dword) : Dword; stdcall;
```

Input / Output

Name	in/out	Format	Init Value	Explanation
AxisNo	in	long	-	축번호(0 ~ (최대축수 - 1))
Use	In	DWORD	1	알람 신호 사용 여부 및 Active Level

Use

#define	Value	Explanation
LOW	00h	B 접점(NORMAL CLOSE)
HIGH	01h	A 접점(NORMAL OPEN)
UNUSED	02h	사용안함
USED	03h	현상태유지

Return AXT_RT_SUCCESS(0000) : Successful execution of API

* See error code Table more information on status error codes

Description

Alarm 입력이 On 되면 해당 축의 모션작업은 급정지 하게 된다.
Low(B접점) : 평상시 Close -> 감지되면 Open
High(A접점): 평상시 Open -> 감지되면 Close

C Example

```
// 0축에 알람 신호 입력 시 비상 정지의 사용 여부를 설정
AxmSignalSetServoAlarm (0, HIGH);

// 0축의 알람 신호 입력 시 비상 정지 여부의 설정을 확인
DWORD    use;

AxmSignalGetServoAlarm(0, &use);
```

VB Example

```
' 0축에 알람 신호 입력 시 비상 정지의 사용 여부를 설정
AxmSignalSetServoAlarm 0, HIGH

' 0축의 알람 신호 입력 시 비상 정지 여부의 설정을 확인
Dim use As Long

AxmSignalGetServoAlarm 0, use
```

Delphi Example

```
{ 0축의 알람 신호 입력 시 비상 정지 여부의 설정을 확인 }
var
use : DWord;

Begin
{ 0축에 알람 신호 입력 시 비상 정지의 사용 여부를 설정 }
AxmSignalSetServoAlarm (0, HIGH);
AxmSignalGetServoAlarm(0, @use);
end;
```

See Also

AxmSignalSetZphaseLevel, AxmSignalGetZphaseLevel, AxmSignalSetServoOnLevel, AxmSignalGetServoOnLevel, AxmSignalSetServoAlarmResetLevel, AxmSignalGetServoAlarmResetLevel, AxmSignalSetInpos, AxmSignalGetInpos, AxmSignalReadInpos, AxmSignalGetServoAlarm, AxmSignalReadServoAlarm, AxmSignalSetLimit, AxmSignalGetLimit, AxmSignalReadLimit, AxmSignalSetSoftLimit, AxmSignalGetSoftLimit, AxmSignalSetStop, AxmSignalGetStop, AxmSignalReadStop

AxmSignalGetServoAlarm

Purpose

지정 축의 알람 신호 입력 시 비상 정지의 사용 여부 및 알람 신호 Active Level 설정을 반환한다.

Format

C

```
DWORD AxmSignalGetServoAlarm (long lAxisNo, DWORD *upUse);
```

Visual Basic

```
Function AxmSignalGetServoAlarm (ByVal lAxisNo As Long, ByRef upUse As Long) As Long
```

Delphi

```
function AxmSignalGetServoAlarm (lAxisNo : LongInt; upUse : PDword) : Dword; stdcall;
```

Input / Output

Name	in/out	Format	Init Value	Explanation
AxisNo	in	long	-	축번호(0 ~ (최대축수 - 1))
Use	out	DWORD*	1	알람 신호 사용 여부 및 Active Level

Use

#define	Value	Explanation
LOW	00h	B 접점(NORMAL CLOSE)
HIGH	01h	A 접점(NORMAL OPEN)
UNUSED	02h	사용안함
USED	03h	현상태유지

Return AXT_RT_SUCCESS(0000) : Successful execution of API

* See error code Table more information on status error codes

Description

'AxmSignalSetServoAlarm' 로 설정한 Servo-Alarm신호의 출력 레벨 설정을 반환한다.
Alarm 입력이 On 되면 해당 축의 모션작업은 급정지 하게 된다.
Low(B접점) : 평상시 Close -> 감지되면 Open
High(A접점): 평상시 Open -> 감지되면 Close

C Example

```
// 0축에 알람 신호 입력 시 비상 정지의 사용 여부를 설정
AxmSignalSetServoAlarm (0, HIGH);

// 0축의 알람 신호 입력 시 비상 정지 사용 여부의 설정을 확인
DWORD    use;

AxmSignalGetServoAlarm(0, &use);
```

VB Example

```
' 0축에 알람 신호 입력 시 비상 정지의 사용 여부를 설정
AxmSignalSetServoAlarm 0, HIGH

' 0축의 알람 신호 입력 시 비상 정지 사용 여부의 설정을 확인
Dim use As Long

AxmSignalGetServoAlarm 0, use
```

Delphi Example

```
{ 0축의 알람 신호 입력 시 비상 정지 사용 여부의 설정을 확인 }
var
use : DWord;

Begin
{ 0축에 알람 신호 입력 시 비상 정지의 사용 여부를 설정 }
AxmSignalSetServoAlarm (0, HIGH);
AxmSignalGetServoAlarm(0, @use);
end;
```

See Also

AxmSignalSetZphaseLevel, AxmSignalGetZphaseLevel, AxmSignalSetServoOnLevel, AxmSignalGetServoOnLevel, AxmSignalSetServoAlarmResetLevel, AxmSignalGetServoAlarmResetLevel, AxmSignalSetInpos, AxmSignalGetInpos, AxmSignalReadInpos, AxmSignalSetServoAlarm, AxmSignalReadServoAlarm, AxmSignalSetLimit, AxmSignalGetLimit, AxmSignalReadLimit, AxmSignalSetSoftLimit, AxmSignalGetSoftLimit, AxmSignalSetStop, AxmSignalGetStop, AxmSignalReadStop

AxmSignalReadServoAlarm

Purpose

지정 축의 알람 신호의 입력 상태를 반환한다.

Format

C

```
DWORD AxmSignalReadServoAlarm (long lAxisNo, DWORD *upStatus);
```

Visual Basic

```
Function AxmSignalReadServoAlarm (ByVal lAxisNo As Long, ByVal upStatus As Long) As Long
```

Delphi

```
function AxmSignalReadServoAlarm (lAxisNo : LongInt; upStatus : PDword) : Dword; stdcall;
```

Input / Output

Name	in/out	Format	Init Value	Explanation
AxisNo	in	long	-	축번호(0 ~ (최대축수 - 1))
Status	out	DWORD*	-	알람 신호의 입력 상태

Status

#define	Value	Explanation
INACTIVE	00h	비활성화
ACTIVE	01h	활성화

Return AXT_RT_SUCCESS(0000) : Successful execution of API

* See error code Table more information on status error codes

Description

'AxmSignalSetServoAlarm' 로 설정한 Servo-Alarm신호의 현재 입력상태를 반환한다
Alarm 입력이 On 되면 해당 축의 모션작업은 급정지 하게 된다.
입력신호가 On되면 upStatus가 1, Off되면 0을 반환한다
Low(B접점) : 평상시 Close -> 감지되면 Open
High(A점점): 평상시 Open -> 감지되면 Close

C Example

```
// 0축의 알람 신호 확인
DWORD upStatus;
AxmSignalReadServoAlarm (0, &upStatus);
```

VB Example

```
' 0축의 알람 신호 확인
Dim upStatus As Long
AxmSignalReadServoAlarm 0, upStatus
```

Delphi Example

```
{ 0축의 알람 신호 확인 }
var
upStatus: DWord;

begin
AxmSignalReadServoAlarm (0, @upStatus);
end;
```

See Also

AxmSignalSetZphaseLevel, AxmSignalGetZphaseLevel, AxmSignalSetServoOnLevel, AxmSignalGetServoOnLevel, AxmSignalSetServoAlarmResetLevel, AxmSignalGetServoAlarmResetLevel, AxmSignalSetInpos, AxmSignalGetInpos, AxmSignalReadInpos, AxmSignalSetServoAlarm, AxmSignalGetServoAlarm, AxmSignalSetLimit, AxmSignalGetLimit, AxmSignalReadLimit, AxmSignalSetSoftLimit, AxmSignalGetSoftLimit, AxmSignalSetStop, AxmSignalGetStop, AxmSignalReadStop

AxmSignalSetLimit

Purpose

지정 축의 리미트 센서 사용 유무 및 신호 Active Level을 설정한다.

Format

C

```
DWORD AxmSignalSetLimit (long lAxisNo, DWORD uStopMode, DWORD uPositiveLevel, DWORD uNegativeLevel);
```

Visual Basic

```
Function AxmSignalSetLimit (ByVal lAxisNo As Long, ByVal uStopMode As Long, ByVal uPositiveLevel As Long, ByVal uNegativeLevel As Long) As Long
```

Delphi

```
function AxmSignalSetLimit (lAxisNo : LongInt; uStopMode : DWord; uPositiveLevel : DWord; uNegativeLevel : DWord) : DWord; stdcall;
```

Input / Output

Name	in/out	Format	Init Value	Explanation
AxisNo	in	long	–	축번호(0 ~ (최대축수 – 1))
StopMode	In	DWORD	0	리미트 센서 검출 후 정지 모드 설정값
PositiveLevel	In	DWORD	1	+ 리미트 센서 Active Level
NegativeLevel	In	DWORD	1	– 리미트 센서 Active Level

StopMode

#define	Value	Explanation
EMERGENCY_STOP	00h	급정지
SLOWDOWN_STOP	01h	감속정지

PositiveLevel

#define	Value	Explanation
LOW	00h	B 접점(NORMAL CLOSE)
HIGH	01h	A 접점(NORMAL OPEN)
UNUSED	02h	사용안함
USED	03h	현상태유지

NegativeLevel

#define	Value	Explanation
LOW	00h	B 접점(NORMAL CLOSE)
HIGH	01h	A 접점(NORMAL OPEN)
UNUSED	02h	사용안함
USED	03h	현상태유지

Return AXT_RT_SUCCESS(0000) : Successful execution of API

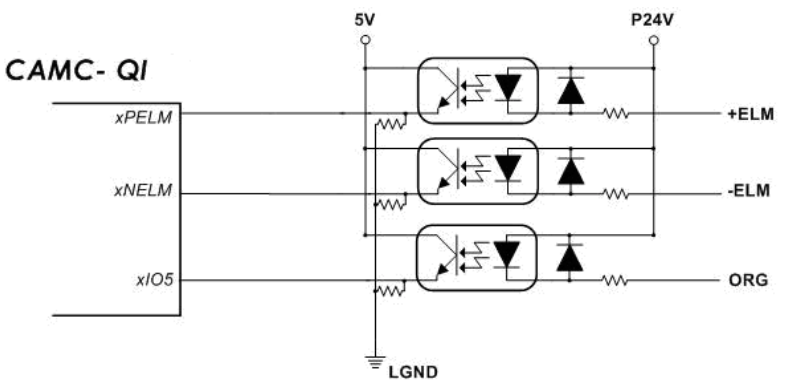
* See error code Table more information on status error codes

Description

-End Limit가 입력이 On되면 해당 축의 (-)방향 모션작업은 정지하게 되고, +End Limit 입력이On되면 (+)방향 모션작업은 정지하게 된다.

Low(B접점) : 평상시 Close -> 감지되면 Open

High(A접점): 평상시 Open -> 감지되면 Close



C Example

```
// 0축에 end limit sensor의 사용 가능하게 설정함
AxmSignalSetLimit (0, EMERGENCY_STOP, HIGH, HIGH);

// 0축의 end limit sensor의 사용 유무 확인
DWORD StopMode, PositiveLevel, NegativeLevel;
AxmSignalGetLimit (0, &StopMode, &PositiveLevel, &NegativeLevel);
```

VB Example

```
' 0축에 end limit sensor의 사용 가능하게 설정함
AxmSignalSetLimit 0, EMERGENCY_STOP, HIGH, HIGH

' 0축의 end limit sensor의 사용 유무 확인
Dim StopMode , PositiveLevel, NegativeLevel As Long
AxmSignalGetLimit 0, StopMode, PositiveLevel, NegativeLevel
```

Delphi Example

```
{ 0축의 end limit sensor의 사용 유무 확인 }
var
StopMode, PositiveLevel, NegativeLevel: DWord;

Begin
{ 0축에 end limit sensor의 사용 가능하게 설정함 }
AxmSignalSetLimit (0, EMERGENCY_STOP, HIGH, HIGH);
AxmSignalGetLimit (0, @StopMode, @PositiveLevel, @NegativeLevel);
end;
```

See Also

AxmSignalSetZphaseLevel, AxmSignalGetZphaseLevel, AxmSignalSetServoOnLevel, AxmSignalGetServoOnLevel, AxmSignalSetServoAlarmResetLevel, AxmSignalGetServoAlarmResetLevel, AxmSignalSetInpos, AxmSignalGetInpos, AxmSignalReadInpos, AxmSignalSetServoAlarm, AxmSignalGetServoAlarm, AxmSignalReadServoAlarm, AxmSignalGetLimit, AxmSignalReadLimit, AxmSignalSetSoftLimit, AxmSignalGetSoftLimit, AxmSignalSetStop, AxmSignalGetStop, AxmSignalReadStop

AxmSignalGetLimit

Purpose

지정 축의 리미트 센서 사용 유무 및 신호 Active Level설정 값을 반환한다.

Format

C

```
DWORD AxmSignalGetLimit (long lAxisNo, DWORD *upStopMode, DWORD *upPositiveLevel,
DWORD *upNegativeLevel);
```

Visual Basic

```
Function AxmSignalGetLimit (ByVal lAxisNo As Long, ByRef upStopMode As Long, ByRef
upPositiveLevel As Long, ByRef upNegativeLevel As Long) As Long
```

Delphi

```
function AxmSignalGetLimit (lAxisNo : LongInt; upStopMode : PDword; upPositiveLevel : PDword;
upNegativeLevel : PDword) : Dword; stdcall;
```

Input / Output

Name	in/out	Format	Init Value	Explanation
AxisNo	in	long	-	축번호(0 ~ (최대축수 - 1))
StopMode	out	DWORD*	0	리미트 센서 검출 후 정지 모드 설정값
PositiveLevel	out	DWORD*	1	+ 리미트 센서 Active Level
NegativeLevel	out	DWORD*	1	- 리미트 센서 Active Level

StopMode

#define	Value	Explanation
EMERGENCY_STOP	00h	급정지
SLOWDOWN_STOP	01h	감속정지

PositiveLevel

#define	Value	Explanation
LOW	00h	B 접점(NORMAL CLOSE)
HIGH	01h	A 접점(NORMAL OPEN)
UNUSED	02h	사용안함
USED	03h	현상태유지

NegativeLevel

#define	Value	Explanation
LOW	00h	B 접점(NORMAL CLOSE)
HIGH	01h	A 접점(NORMAL OPEN)
UNUSED	02h	사용안함
USED	03h	현상태유지

Return AXT_RT_SUCCESS(0000) : Successful execution of API
* See error code Table more information on status error codes

Description

참고 : 필요한 데이터만 가지고 올 경우, 필요하지 않는 변수에 NULL을 넣어 사용 가능하다.
' AxmSignalSetLimit ' 로 설정한 Pos,Neg End Limt 신호의 출력 레벨 설정을 반환과 StopMode를 반환한다. -End Limit가 입력이 On되면 해당 축의 (-)방향 모션작업은 정지하게 되고, +End Limit 입력이On되면 (+)방향 모션작업은 정지하게 된다.
Low(B접점) : 평상시 Close -> 감지되면 Open
High(A점점): 평상시 Open -> 감지되면 Close

C Example

```
// 0축에 end limit sensor의 사용 가능하게 설정함
AxmSignalSetLimit (0, EMERGENCY_STOP, HIGH, HIGH);

// 0축의 end limit sensor의 사용 유무 확인
DWORD StopMode, PositiveLevel, NegativeLevel;
AxmSignalGetLimit (0, &StopMode, &PositiveLevel, &NegativeLevel);
```

VB Example

```
' 0축에 end limit sensor의 사용 가능하게 설정함
AxmSignalSetLimit 0, EMERGENCY_STOP, HIGH, HIGH

' 0축의 end limit sensor의 사용 유무 확인
Dim StopMode , PositiveLevel, NegativeLevel As Long
AxmSignalGetLimit 0, StopMode, PositiveLevel, NegativeLevel
```

Delphi Example

```
{ 0축의 end limit sensor의 사용 유무 확인 }
var
StopMode, PositiveLevel, NegativeLevel: DWord;

Begin
```

```
{ 0축에 end limit sensor의 사용 가능하게 설정함 }
AxmSignalSetLimit (0, EMERGENCY_STOP, HIGH, HIGH);
AxmSignalGetLimit (0, @StopMode, @PositiveLevel, @NegativeLevel);
end;
```

See Also

AxmSignalSetZphaseLevel, AxmSignalGetZphaseLevel, AxmSignalSetServoOnLevel, AxmSignalGetServoOnLevel, AxmSignalSetServoAlarmResetLevel, AxmSignalGetServoAlarmResetLevel, AxmSignalSetInpos, AxmSignalGetInpos, AxmSignalReadInpos, AxmSignalSetServoAlarm, AxmSignalGetServoAlarm, AxmSignalReadServoAlarm, AxmSignalSetLimit, AxmSignalReadLimit, AxmSignalSetSoftLimit, AxmSignalGetSoftLimit, AxmSignalSetStop, AxmSignalGetStop, AxmSignalReadStop

AxmSignalReadLimit

Purpose

지정 축의 리미트 센서 신호의 입력 상태를 반환한다.

Format

C

```
DWORD AxmSignalReadLimit   (long lAxisNo, DWORD *upPositiveStatus, DWORD
*upNegativeStatus);
```

Visual Basic

```
Function AxmSignalReadLimit (ByVal lAxisNo As Long, ByRef upPositiveStatus As Long, ByRef
upNegativeStatus As Long) As Long
```

Delphi

```
function AxmSignalReadLimit (lAxisNo : LongInt; upPositiveStatus : PDword; upNegativeStatus :
PDword) : Dword; stdcall;
```

Input / Output

Name	in/out	Format	Init Value	Explanation
AxisNo	in	long	-	축번호(0 ~ (최대축수 - 1))
PositiveStatus	out	DWORD*	0	+ 리미트 센서 신호 입력 상태
NegativeStatus	out	DWORD*	0	- 리미트 센서 신호 입력 상태

PositiveStatus

#define	Value	Explanation
INACTIVE	00h	비활성화 + 리미트 센서 신호 Off 상태
ACTIVE	01h	활성화 + 리미트 센서 신호 On 상태

NegativeStatus

#define	Value	Explanation
INACTIVE	00h	비활성화 - 리미트 센서 신호 Off 상태
ACTIVE	01h	활성화 - 리미트 센서 신호 On 상태

Return AXT_RT_SUCCESS(0000) : Successful execution of API

* See error code Table more information on status error codes

Description

참고 : 필요한 데이터만 가지고 오고 싶을 때는 필요하지 않는 변수에 NULL 넣어 사용 가능하다.
' AxmSignalSetLimit ' 로 설정한 Pos,Neg End Limt 신호의 현재 입력상태를 반환한다.
-End Limit가 입력이 On되면 해당 축의 (-)방향 모션작업은 정지하게 되고, +End Limit 입력이On되면 (+)방향 모션작업은 정지하게 된다.

Low(B접점) : 평상시 Close -> 감지되면 Open
High(A점점): 평상시 Open -> 감지되면 Close

C Example

```
// 0축 리미트 센서 입력 상태 확인
DWORD upPositiveLevel, upNegativeLevel;
AxmSignalReadLimit (0, &upPositiveLevel, &upNegativeLevel);
```

VB Example

```
' 0 축 리미트 센서 입력 상태 확인
Dim upPositiveLevel , upNegativeLevel As Long
AxmSignalReadLimit 0, upPositiveLevel, upNegativeLevel
```

Delphi Example

```
{ 0 축 리미트 센서 입력 상태 확인 }
var
upPositiveLevel, upNegativeLevel: DWord;

begin
AxmSignalReadLimit (0, @upPositiveLevel, @upNegativeLevel);
end;
```

See Also

AxmSignalSetZphaseLevel, AxmSignalGetZphaseLevel, AxmSignalSetServoOnLevel, AxmSignalGetServoOnLevel, AxmSignalSetServoAlarmResetLevel, AxmSignalGetServoAlarmResetLevel, AxmSignalSetInpos, AxmSignalGetInpos, AxmSignalReadInpos, AxmSignalSetServoAlarm, AxmSignalGetServoAlarm, AxmSignalReadServoAlarm, AxmSignalSetLimit, AxmSignalGetLimit, AxmSignalSetSoftLimit, AxmSignalGetSoftLimit, AxmSignalSetStop, AxmSignalGetStop, AxmSignalReadStop

AxmSignalSetSoftLimit

Purpose

지정축의 소프트 리미트 기능을 설정한다.

Format

C

```
DWORD AxmSignalSetSoftLimit (long lAxisNo, DWORD uUse, DWORD uStopMode, DWORD uSelection, double dPositivePos, double dNegativePos);
```

Visual Basic

```
Function AxmSignalSetSoftLimit (ByVal lAxisNo As Long, ByVal uUse As Long, ByVal uStopMode As Long, ByVal uSelection As Long, ByVal dPositivePos As Double, ByVal dNegativePos As Double) As Long
```

Delphi

```
function AxmSignalSetSoftLimit (lAxisNo : LongInt; uUse : Dword; uStopMode : Dword; uSelection : Dword; dPositivePos : Double; dNegativePos : Double) : Dword; stdcall;
```

Input / Output

Name	in/out	Format	Init Value	Explanation
AxisNo	in	long	-	축번호(0 ~ (최대축수 - 1))
Use	In	DWORD	1	소프트 리미트 기능 사용 유무
StopMode	In	DWORD	0	소프트 리미트 On 시 모션 정지 모드
Selection	In	DWORD	0	현재 위치 입력 소스 선택
PositivePos	In	double	0	+ 방향 소프트 리미트 위치값
NegativePos	In	double	0	- 방향 소프트 리미트 위치값

Use

#define	Value	Explanation
DISABLE	00h	소프트 리미트 사용하지 않음
ENABLE	01h	소프트 리미트 사용

StopMode

#define	Value	Explanation
EMERGENCY_STOP	00h	급정지

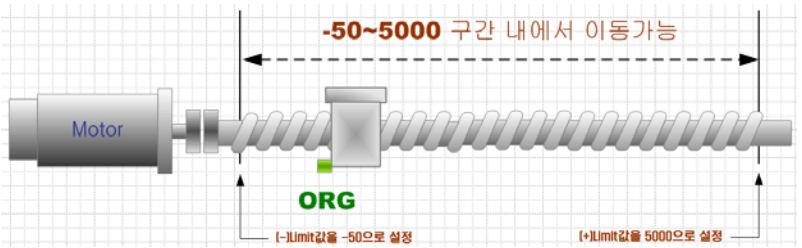
SLOWDOWN_STOP	01h	감속정지

Selection

#define	Value	Explanation
COMMAND	00h	목표 위치 Cmd Pos을 현재 위치 비교값으로 사용
ACTUAL	01h	실제 위치 Act Pos을 현재 위치 비교값으로 사용

Return AXT_RT_SUCCESS(0000) : Successful execution of API

* See error code Table more information on status error codes



Description

소프트 리미트 기능을 사용하면 dPositivePositiin과 dNegativePos 범위 내에서 모션 동작이 가능하며, 이 범위를 벗어나게 되면 소프트 리미트에 의하여 모션이 정지하게 된다.

주의사항:

1. 소프트웨어 리밋을 미리 설정해놓고 원점검색을 수행할 경우 도중에 사용자 또는 외부의요인에 의해서 원점 검색이 중지 되어도 소프트웨어 리밋은 설정을 유지 한다.
2. 소프트웨어 리밋을 미리 설정해놓고 원점검색을 수행할 경우 원점 검색 시작시 소프트웨어 리밋이 DISABLE되었다가 원점검색을 완료하면 소프트웨어 리밋은 다시 ENABLE 된다.

N Limit : (-)방향으로의 스트로크 한계값을 지정한다.
P Limit: (+)방향으로의 스트로크 한계값을 지정한다.

C Example

```
// 0축의 Soft Limit를 설정한다.
AxmSignalSetSoftLimit(0, ENABLE, EMERGENCY_STOP, CMD, 50000, -50);
```

```
// 0축의 Soft Limit 설정 사항 확인
DWORD upUse, upStopMode, upSelection;
double dpPositivePos, dpNegativePos;

AxmSignalGetSoftLimit (0, &upUse, &upStopMode, &upSelection,
 &dpPositivePos, &dpNegativePos);
```

VB Example

```
' 0축의 Soft Limit를 설정한다.
AxmSignalSetSoftLimit 0, ENABLE, EMERGENCY_STOP, CMD, 50000, -50

' 0축의 Soft Limit 설정 사항 확인
Dim upUse , upStopMode , upSelection As Long
Dim dpPositivePos , dpNegativePos As Double

AxmSignalGetSoftLimit 0, upUse, upStopMode, upSelection, dpPositivePos,
 dpNegativePos
```

Delphi Example

```
{ 0축의 Soft Limit 설정 사항 확인 }
var
upUse, upStopMode, upSelection: DWord;
dpPositivePos, dpNegativePos: Double;

Begin
{ 0축의 Soft Limit를 설정한다. }
AxmSignalSetSoftLimit(0, ENABLE, EMERGENCY_STOP, CMD, 50000, -50);
AxmSignalGetSoftLimit (0, @upUse, @upStopMode, @upSelection,
 @dpPositivePos, @dpNegativePos);
end;
```

See Also

AxmSignalSetZphaseLevel, AxmSignalGetZphaseLevel, AxmSignalSetServoOnLevel, AxmSignalGetServoOnLevel, AxmSignalSetServoAlarmResetLevel, AxmSignalGetServoAlarmResetLevel, AxmSignalSetInpos, AxmSignalGetInpos, AxmSignalReadInpos, AxmSignalSetServoAlarm, AxmSignalGetServoAlarm, AxmSignalReadServoAlarm, AxmSignalSetLimit, AxmSignalGetLimit, AxmSignalReadLimit, AxmSignalGetSoftLimit, AxmSignalSetStop, AxmSignalGetStop, AxmSignalReadStop

AxmSignalGetSoftLimit

Purpose

지정축의 소프트 리미트 기능을 사용유무, 위치, 정지모드 등을 반환한다.

Format

C

```
DWORD AxmSignalGetSoftLimit (long lAxisNo, DWORD *upUse, DWORD *upStopMode, DWORD
*upSelection, double *dpPositivePos, double *dpNegativePos);
```

Visual Basic

```
Function AxmSignalGetSoftLimit (ByVal lAxisNo As Long, ByRef upUse As Long, ByRef
upStopMode As Long, ByRef upSelection As Long, ByRef dpPositivePos As Double, ByRef
dpNegativePos As Double) As Long
```

Delphi

```
function AxmSignalGetSoftLimit (lAxisNo : LongInt; upUse : PDWord; upStopMode : PDWord;
upSelection : PDWord; dpPositivePos : PDouble; dpNegativePos : PDouble) : DWord; stdcall;
```

Input / Output

Name	in/out	Format	Init Value	Explanation
AxisNo	in	long	-	축번호(0 ~ (최대축수 - 1))
Use	out	DWORD	1	소프트 리미트 기능 사용 유무
StopMode	out	DWORD	0	소프트 리미트 On 시 모션 정지 모드
Selection	out	DWORD	0	현재 위치 입력 소스 선택
PositivePos	out	double	0	+ 방향 소프트 리미트 위치값
NegativePos	out	double	0	- 방향 소프트 리미트 위치값

Use

#define	Value	Explanation
DISABLE	00h	소프트 리미트 사용하지 않음
ENABLE	01h	소프트 리미트 사용

StopMode

#define	Value	Explanation
EMERGENCY_STOP	00h	급정지

SLOWDOWN_STOP	01h	감속정지

Selection

#define	Value	Explanation
COMMAND	00h	목표 위치 Cmd Pos을 현재 위치 비교값으로 사용
ACTUAL	01h	실제 위치 Act Pos을 현재 위치 비교값으로 사용

Return AXT_RT_SUCCESS(0000) : Successful execution of API

* See error code Table more information on status error codes

Description

참고 : 필요한 데이터만 가지고 오고 싶을 때는 필요하지 않는 변수에 NULL 넣어 사용 가능하다.
' AxmSignalSetSoftLimit ' 로 설정한 Pos, Neg 한계값을 반환한다.

C Example

```
// 0축의 Soft Limit를 설정한다.
AxmSignalSetSoftLimit(0, ENABLE, EMERGENCY_STOP, CMD, 50000, -50);

// 0축의 Soft Limit 설정 사항 확인
DWORD upUse, upStopMode, upSelection;
double dpPositivePos, dpNegativePos;

AxmSignalGetSoftLimit (0, &upUse, &upStopMode, &upSelection,
       &dpPositivePos, &dpNegativePos);
```

VB Example

```
' 0축의 Soft Limit를 설정한다.
AxmSignalSetSoftLimit 0, ENABLE, EMERGENCY_STOP, CMD, 50000, -50

' 0축의 Soft Limit 설정 사항 확인
Dim upUse , upStopMode , upSelection As Long
Dim dpPositivePos , dpNegativePos As Double

AxmSignalGetSoftLimit 0, upUse, upStopMode, upSelection, dpPositivePos,
       dpNegativePos
```

Delphi Example

```
{ 0축의 Soft Limit 설정 사항 확인 }
var
upUse, upStopMode, upSelection: DWord;
dpPositivePos, dpNegativePos: Double;
```

```
Begin
{ 0축의 Soft Limit를 설정한다. }
AxmSignalSetSoftLimit(0, ENABLE, EMERGENCY_STOP, CMD, 50000, -50);
AxmSignalGetSoftLimit (0, @upUse, @upStopMode, @upSelection,
       @dpPositivePos, @dpNegativePos);
end;
```

See Also

AxmSignalSetZphaseLevel, AxmSignalGetZphaseLevel, AxmSignalSetServoOnLevel, AxmSignalGetServoOnLevel, AxmSignalSetServoAlarmResetLevel, AxmSignalGetServoAlarmResetLevel, AxmSignalSetInpos, AxmSignalGetInpos, AxmSignalReadInpos, AxmSignalSetServoAlarm, AxmSignalGetServoAlarm, AxmSignalReadServoAlarm, AxmSignalSetLimit, AxmSignalGetLimit, AxmSignalReadLimit, AxmSignalSetSoftLimit, AxmSignalSetStop, AxmSignalGetStop, AxmSignalReadStop

AxmSignalSetStop

Purpose

지정 축의 비상 정지 신호(SSTOP, ESTOP) 정지모드와 레벨를 설정한다.

Format

C

```
DWORD AxmSignalSetStop (long lAxisNo, DWORD uStopMode, DWORD uLevel);
```

Visual Basic

```
Function AxmSignalSetStop (ByVal lAxisNo As Long, ByVal uStopMode As Long, ByVal uLevel As Long) As Long
```

Delphi

```
function AxmSignalSetStop (lAxisNo : LongInt; uStopMode : Dword; uLevel : Dword) : Dword; stdcall;
```

Input / Output

Name	in/out	Format	Init Value	Explanation
AxisNo	in	long	-	축번호(0 ~ (최대축수 - 1))
StopMode	In	DWORD	0	정지 모드(급정지만 지원함)
Level	In	DWORD	0	정지 신호 사용 유무 및 Active Level

StopMode

#define	Value	Explanation
EMERGENCY_STOP	00h	급정지
SLOWDOWN_STOP	01h	감속정지

Level

#define	Value	Explanation
LOW	00h	B 접점(NORMAL CLOSE)
HIGH	01h	A 접점(NORMAL OPEN)
UNUSED	02h	사용안함
USED	03h	현상태유지

Return AXT_RT_SUCCESS(0000) : Successful execution of API

* See error code Table more information on status error codes

Description

사용자가 지정한 축의 비상 정지 신호의 레벨과 정지모드를 설정하는데 잘못 설정되면 펄스출력이 안나가는 현상이 발생할 수 있다. 이 신호 레벨은 AxmStatusReadMechanical로 확인할 수 있다.

Low(B접점) : 평상시 Close -> 감지되면 Open

High(A점점): 평상시 Open -> 감지되면 Close

C Example

```
// 0축의 정지 신호 사용 유무를 설정
AxmSignalSetStop (0, EMERGENCY_STTOP, ENABLE);

// 0축의 비상 정지 신호 사용 유무 설정 값을 확인
DWORD upStopMode, upLevel;
AxmSignalGetStop (0, &upStopMode, &upLevel);
```

VB Example

```
‘ 0축의 정지 신호 사용 유무를 설정
AxmSignalSetStop 0, EMERGENCY_STTOP, ENABLE

‘ 0축의 비상 정지 신호 사용 유무 설정 값을 확인
Dim upStopMode , upLevel As Long
AxmSignalGetStop 0, upStopMode, upLevel
```

Delphi Example

```
{ 0축의 비상 정지 신호 사용 유무 설정 값을 확인 }
var
upStopMode, upLevel: DWord;

Begin
{ 0축의 정지 신호 사용 유무를 설정 }
AxmSignalSetStop (0, EMERGENCY_STTOP, ENABLE);
AxmSignalGetStop (0, @upStopMode, @upLevel);
end;
```

See Also

AxmSignalSetZphaseLevel, AxmSignalGetZphaseLevel, AxmSignalSetServoOnLevel, AxmSignalGetServoOnLevel, AxmSignalSetServoAlarmResetLevel, AxmSignalGetServoAlarmResetLevel, AxmSignalSetInpos, AxmSignalGetInpos, AxmSignalReadInpos, AxmSignalSetServoAlarm, AxmSignalGetServoAlarm, AxmSignalReadServoAlarm, AxmSignalSetLimit, AxmSignalGetLimit, AxmSignalReadLimit, AxmSignalSetSoftLimit, AxmSignalGetSoftLimit, AxmSignalGetStop, AxmSignalReadStop

AxmSignalGetStop

Purpose

지정 축의 비상 정지 신호(SSTOP, ESTOP)의 정지모드와 레벨를 반환한다.

Format

C

```
DWORD AxmSignalGetStop (long lAxisNo, DWORD *upStopMode, DWORD *upLevel);
```

Visual Basic

```
Function AxmSignalGetStop (ByVal lAxisNo As Long, ByRef upStopMode As Long, ByRef upLevel
As Long) As Long
```

Delphi

```
function AxmSignalGetStop (lAxisNo : LongInt; upStopMode : PDword; upLevel : PDword) :
Dword; stdcall;
```

Input / Output

Name	in/out	Format	Init Value	Explanation
AxisNo	in	long	-	축번호(0 ~ (최대축수 - 1))
StopMode	out	DWORD*	0	정지 모드(급정지만 지원함)
Level	out	DWORD*	0	정지 신호 사용 유무 및 Active Level

StopMode

#define	Value	Explanation
EMERGENCY_STOP	00h	급정지
SLOWDOWN_STOP	01h	감속정지

Level

#define	Value	Explanation
LOW	00h	B 접점(NORMAL CLOSE)
HIGH	01h	A 접점(NORMAL OPEN)
UNUSED	02h	사용안함
USED	03h	현상태유지

Return AXT_RT_SUCCESS(0000) : Successful execution of API

* See error code Table more information on status error codes

Description

참고 : 필요한 데이터만 가지고 올 경우, 필요하지 않는 변수에 NULL을 넣어 사용 가능하다.
' AxmSignalSetStop' 로 설정한 비상 정지 신호(SSTOP, ESTOP)의 정지모드와 레벨를 반환한다.
비상 정지 신호의 레벨과 정지모드를 설정 할 때, 잘못 설정되면 펄스출력이 안나가는 현상이 발생할 수 있다. 이 신호 레벨은 AxmStatusReadMechanical로 확인할 수 있다.
Low(B접점) : 평상시 Close -> 감지되면 Open
High(A점점): 평상시 Open -> 감지되면 Close

C Example

```
// 0축의 정지 신호 사용 유무를 설정
AxmSignalSetStop (0, EMERGENCY_STTOP, ENABLE);

// 0축의 비상 정지 신호 사용 유무 설정 값을 확인
DWORD upStopMode, upLevel;
AxmSignalGetStop (0, &upStopMode, &upLevel);
```

VB Example

```
‘ 0축의 정지 신호 사용 유무를 설정
AxmSignalSetStop 0, EMERGENCY_STTOP, ENABLE

‘ 0축의 비상 정지 신호 사용 유무 설정 값을 확인
Dim upStopMode , upLevel As Long
AxmSignalGetStop 0, upStopMode, upLevel
```

Delphi Example

```
{ 0축의 비상 정지 신호 사용 유무 설정 값을 확인 }
var
upStopMode, upLevel: DWord;
Begin
{ 0축의 정지 신호 사용 유무를 설정 }
AxmSignalSetStop (0, EMERGENCY_STTOP, ENABLE);
AxmSignalGetStop (0, @upStopMode, @upLevel);
end;
```

See Also

AxmSignalSetZphaseLevel, AxmSignalGetZphaseLevel, AxmSignalSetServoOnLevel, AxmSignalGetServoOnLevel, AxmSignalSetServoAlarmResetLevel, AxmSignalGetServoAlarmResetLevel, AxmSignalSetInpos, AxmSignalGetInpos, AxmSignalReadInpos, AxmSignalSetServoAlarm, AxmSignalGetServoAlarm, AxmSignalReadServoAlarm, AxmSignalSetLimit, AxmSignalGetLimit, AxmSignalReadLimit, AxmSignalSetSoftLimit, AxmSignalGetSoftLimit, AxmSignalSetStop, AxmSignalReadStop

AxmSignalReadStop

Purpose

지정 축의 비상 정지 신호의 입력 상태를 반환한다.

Format

C

DWORD AxmSignalReadStop (long lAxisNo, DWORD *upStatus);

Visual Basic

Function AxmSignalReadStop (ByVal lAxisNo As Long, ByRef upStatus As Long) As Long

Delphi

function AxmSignalReadStop (lAxisNo : LongInt; upStatus : PDword) : Dword; stdcall;

Input / Output

Name	in/out	Format	Init Value	Explanation
AxisNo	in	long	-	축번호(0 ~ (최대축수 - 1))
Status	out	DWORD*	-	펄스 출력 방식

Status

#define	Value	Explanation
INACTIVE	00h	비활성화 (정지 신호 Off)
ACTIVE	01h	활성화 (정지 신호 On)

Return AXT_RT_SUCCESS(0000) : Successful execution of API

* See error code Table more information on status error codes

Description

' AxmSignalSetStop ' 로 설정한 비상 정지 신호(SSTOP, ESTOP) 의 현재 입력상태를 반환한다.

C Example

```
// 0축의 STOP신호의 상태 확인
DWORD uStatus;
AxmSignalReadStop (0, &uStatus);
```

VB Example

```
' 0축의 STOP신호의 상태 확인
Dim uStatus As Long
AxmSignalReadStop 0, uStatus
```

Delphi Example

```
{ 0축의 STOP신호의 상태 확인 }
var
uStatus: DWord;

begin
AxmSignalReadStop (0, @uStatus);
end;
```

See Also

AxmSignalSetZphaseLevel, AxmSignalGetZphaseLevel, AxmSignalSetServoOnLevel, AxmSignalGetServoOnLevel, AxmSignalSetServoAlarmResetLevel, AxmSignalGetServoAlarmResetLevel AxmSignalSetInpos, AxmSignalGetInpos, AxmSignalReadInpos, AxmSignalSetServoAlarm, AxmSignalGetServoAlarm, AxmSignalReadServoAlarm, AxmSignalSetLimit, AxmSignalGetLimit, AxmSignalReadLimit, AxmSignalSetSoftLimit, AxmSignalGetSoftLimit, AxmSignalSetStop, AxmSignalGetStop

AxmSignalServoOn

Purpose

지정 축의 Servo-On신호를 출력을 설정한다.

Format

C

```
DWORD AxmSignalServoOn (long lAxisNo, DWORD uOnOff);
```

Visual Basic

```
Function AxmSignalServoOn (ByVal lAxisNo As Long, ByVal uOnOff As Long) As Long
```

Delphi

```
function AxmSignalServoOn (lAxisNo : LongInt; uOnOff : DWord) : DWord; stdcall;
```

Input / Output

Name	in/out	Format	Init Value	Explanation
AxisNo	in	long	-	축번호(0 ~ (최대축수 - 1))
OnOff	in	DWORD	1	Serve-On 신호의 출력 설정

OnOff

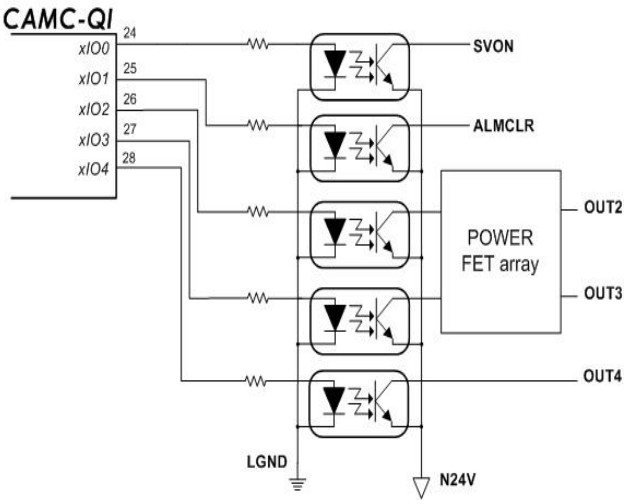
#define	Value	Explanation
FALSE	00h	Servo-On 출력 Off
TRUE	01h	Servo-On 출력 On

Return AXT_RT_SUCCESS(0000) : Successful execution of API

* See error code Table more information on status error codes

Description

서보드라이버를 사용할 때는 외부에서 스위치를 이용하여 서보드라이버의 ON/OFF를 제어할 수 있도록 하는데, 이를 SERVO-ON 신호라고 하고, 이 함수는 SERVO-ON 신호의 ON/OFF를 제어하는 함수다. AxmSignalIsServoOn 함수는 현재의 SERVO-ON 신호의 출력 상태를 반환하고 AxmSignalSetServoOnLevel 함수를 이용해서 Active Level을 설정할 수 있다.



C Example

```
// 0축에 Servo On
AxmSignalServoOn (0, ENABLE);

// 0축의 Servo On 확인
DWORD  level;

AxmSignalIsServoOn (0, &level);
```

VB Example

```
‘ 0축에 Servo On
AxmSignalServoOn 0, ENABLE

‘ 0축의 Servo On 확인
Dim level As Long

AxmSignalIsServoOn 0, level
```

Delphi Example

```
{ 0축의 Servo On 확인 }
var
level: DWord;

Begin

{ 0축에 Servo On }
AxmSignalServoOn (0, ENABLE);
AxmSignalIsServoOn (0, @level);
```

```
end;
```

See Also

AxmSignalIsServoOn, AxmSignalServoAlarmReset, AxmSignalWriteOutput, AxmSignalReadOutput, AxmSignalWriteOutputBit, AxmSignalReadOutputBit, AxmSignalReadInput, AxmSignalReadInputBit

AxmSignalIsServoOn

Purpose

지정 축의 Servo-On 출력 신호의 상태를 반환한다.

Format

C

```
DWORD AxmSignalIsServoOn(long lAxisNo, DWORD *upOnOff);
```

Visual Basic

```
Function AxmSignalIsServoOn(ByVal lAxisNo As Long, ByRef upOnOff As Long) As Long
```

Delphi

```
function AxmSignalIsServoOn(lAxisNo : LongInt; upOnOff : PDWord) : DWord; stdcall;
```

Input / Output

Name	in/out	Format	Init Value	Explanation
AxisNo	in	long	-	축번호(0 ~ (최대축수 - 1))
OnOff	in	DWORD*	1	Serve-On 신호의 출력 설정

OnOff

#define	Value	Explanation
FALSE	00h	Servo-On 출력 Off
TRUE	01h	Servo-On 출력 On

Return AXT_RT_SUCCESS(0000) : Successful execution of API

* See error code Table more information on status error codes

Description

' AxmSignalServoOn' 로 설정한 Servo-On 출력 신호의 상태를 반환한다. 서보드라이버를 사용할 때는 외부에서 스위치를 이용하여 서보드라이버의 ON/OFF를 제어할 수 있도록 하는데, 이를 SERVO-ON 신호라고 하고, 이 함수는 SERVO-ON 신호의 ON/OFF를 제어하는 함수다. AxmSignalIsServoOn 함수는 현재의 SERO-OM 신호의 출력 상태를 반환하고 AxmSignalSetServoOnLevel 함수를 이용해서 Active Level을 설정할 수 있다. 노드의 범용 출력 0은 서보온으로 사용하고 1은 서보 알람 리셋으로 사용한다.

C Example

```
// 0축에 Servo On
AxmSignalServoOn (0, ENABLE);

// 0축의 Servo On 확인
DWORD  level;

AxmSignalIsServoOn (0, &level);
```

VB Example

```
‘ 0축에 Servo On
AxmSignalServoOn 0, ENABLE

‘ 0축의 Servo On 확인
Dim level As Long

AxmSignalIsServoOn 0, level
```

Delphi Example

```
{ 0축의 Servo On 확인 }
var
level: DWord;

Begin

{ 0축에 Servo On }
AxmSignalServoOn (0, ENABLE);
AxmSignalIsServoOn (0, @level);
end;
```

See Also

AxmSignalServoOn, AxmSignalServoAlarmReset, AxmSignalWriteOutput, AxmSignalReadOutput, AxmSignalWriteOutputBit, AxmSignalReadOutputBit, AxmSignalReadInput, AxmSignalReadInputBit

AxmSignalServoAlarmReset

Purpose

지정 축의 Servo-Alarm Reset신호를 출력을 설정한다.

Format

C

```
DWORD AxmSignalServoAlarmReset (long lAxisNo, DWORD uOnOff);
```

Visual Basic

```
Function AxmSignalServoAlarmReset (ByVal lAxisNo As Long, ByVal uOnOff As Long) As Long
```

Delphi

```
function AxmSignalServoAlarmReset (lAxisNo : LongInt; uOnOff : DWord) : DWord; stdcall;
```

Input / Output

Name	in/out	Format	Init Value	Explanation
AxisNo	in	long	-	축번호(0 ~ (최대축수 - 1))
OnOff	in	DWORD	1	Serve-Alarm Reset 신호의 출력 설정

OnOff

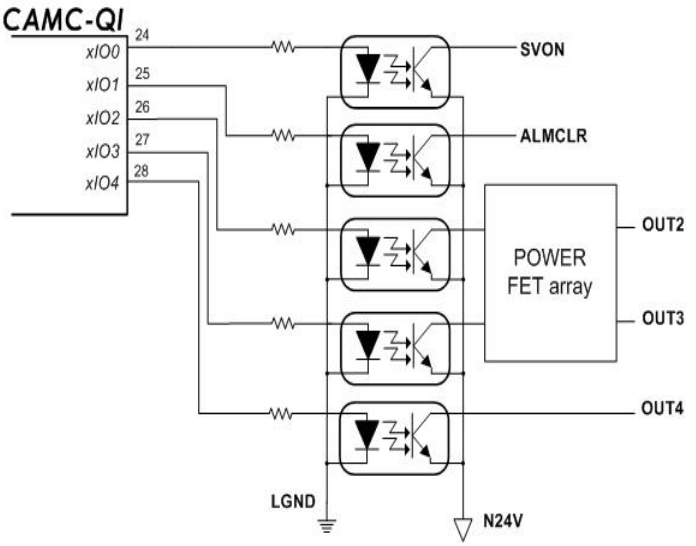
#define	Value	Explanation
FALSE	00h	Servo-Alarm Reset 출력 Off
TRUE	01h	Servo-Alarm Reset 출력 On

Return AXT_RT_SUCCESS(0000) : Successful execution of API

* See error code Table more information on status error codes

Description

지정 축의 과부하 또는 다른 기타 사유로 인해서 Servo Drive에 알람이 발생했을 경우 Clear 시킬 때 사용한다. 노드의 범용 출력 0은 서보온으로 사용하고 1은 서보 알람 리셋으로 사용한다.



C Example

```
// 0축에 Servo Alarm Reset
AxmSignalServoAlarmReset (0, ENABLE);
```

VB Example

```
‘ 0축에 Servo Alarm Reset
AxmSignalServoAlarmReset 0, ENABLE
```

Delphi Example

```
begin
{ 0축에 Servo Alarm Reset }
AxmSignalServoAlarmReset (0, ENABLE);
End;
```

See Also

AxmSignalServoOn, AxmSignalIsServoOn, AxmSignalWriteOutput, AxmSignalReadOutput, AxmSignalWriteOutputBit, AxmSignalReadOutputBit, AxmSignalReadInput, AxmSignalReadInputBit

AxmSignalWriteOutput

Purpose

지정축의 범용 출력값을 설정한다. 여기서 Value값은 Hex값으로 입력한다.

Format

C

```
DWORD AxmSignalWriteOutput (long lAxisNo, DWORD uValue);
```

Visual Basic

```
Function AxmSignalWriteOutput (ByVal lAxisNo As Long, ByVal uValue As Long) As Long
```

Delphi

```
function AxmSignalWriteOutput (lAxisNo : LongInt; uValue : Dword) : Dword ; stdcall;
```

Input / Output

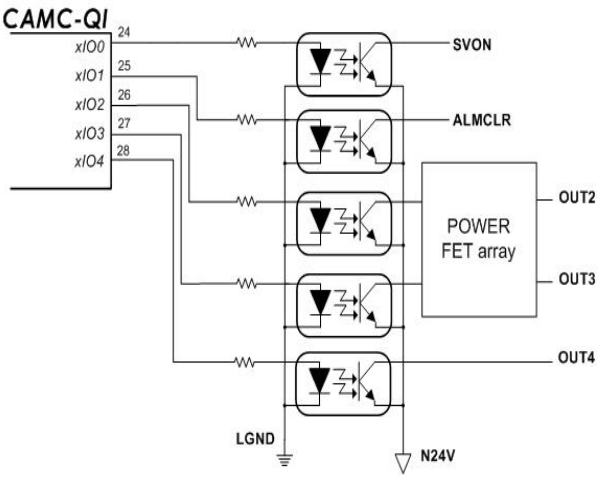
Name	in/out	Format	Init Value	Explanation
AxisNo	in	long	-	축번호(0 ~ (최대축수 - 1))
Value	In	DWORD	-	범용 출력 값(Hex값)

Return AXT_RT_SUCCESS(0000) : Successful execution of API

* See error code Table more information on status error codes

Description

모듈의 범용 출력으로는 0 - 4까지 있으며 기본으로 범용(0)를 서보온으로 쓰며, 범용(1)은 알람 클리어 출력신호로 사용한다. AxmSignalWriteOutput는 동시에 여러 출력을 사용할 때 이 함수를 이용하고 한 개의 출력을 이용할 때는 AxmSignalWriteOutputBit 함수를 이용한다.



C Example

```
// 0축의 범용 출력 신호 설정.
AxmSignalWriteOutput(0, 1);

// 0축의 범용 출력 신호 확인.
DWORD uValue;
AxmSignalReadOutput (0, &uValue);
```

VB Example

```
‘ 0축의 범용 출력 신호 설정
AxmSignalWriteOutput 0, 1

‘0축의 범용 출력 신호 확인.
Dim uValue As Long
AxmSignalReadOutput 0, uValue
```

Delphi Example

```
{ 0축의 범용 출력 신호 확인. }
var
uValue: DWord;

Begin
{ 0축의 범용 출력 신호 설정 }
AxmSignalWriteOutput(0, 1);

AxmSignalReadOutput (0, @uValue);
end;
```

See Also

AxmSignalServoOn, AxmSignalIsServoOn, AxmSignalServoAlarmReset, AxmSignalReadOutput, AxmSignalWriteOutputBit, AxmSignalReadOutputBit, AxmSignalReadInput, AxmSignalReadInputBit

AxmSignalReadOutput

Purpose

지정축의 범용 출력값을 Hex값으로 반환한다.

Format

C

DWORD AxmSignalReadOutput (long lAxisNo, DWORD *upValue);

Visual Basic

Function AxmSignalReadOutput (ByVal lAxisNo As Long, ByRef upValue As Long) As Long

Delphi

function AxmSignalReadOutput (lAxisNo : LongInt; upValue : PDword) : Dword ; stdcall;

Input / Output

Name	in/out	Format	Init Value	Explanation
AxisNo	in	long	-	축번호(0 ~ (최대축수 - 1))
Value	out	DWORD*	-	범용 출력 값

Return AXT_RT_SUCCESS(0000) : Successful execution of API

* See error code Table more information on status error codes

Description

'AxmSignalWriteOutput' 로 설정한 범용 출력값을 반환한다.

C Example

```
// 0축의 범용 출력 신호 설정.
AxmSignalWriteOutput(0, 1);

// 0축의 범용 출력 신호 확인.
DWORD uValue;
AxmSignalReadOutput (0, &uValue);
```

VB Example

```
' 0축의 범용 출력 신호 설정
AxmSignalWriteOutput 0, 1
```

```
'0축의 범용 출력 신호 확인.
Dim uValue As Long
AxmSignalReadOutput 0, uValue
```

Delphi Example

```
{ 0축의 범용 출력 신호 확인. }
var
uValue: DWord;

Begin
{ 0축의 범용 출력 신호 설정 }
AxmSignalWriteOutput(0, 1);

AxmSignalReadOutput (0, @uValue);
end;
```

See Also

AxmSignalServoOn, AxmSignalIsServoOn, AxmSignalServoAlarmReset, AxmSignalWriteOutput, AxmSignalWriteOutputBit, AxmSignalReadOutputBit, AxmSignalReadInput, AxmSignalReadInputBit

AxmSignalWriteOutputBit

Purpose

지정축의 범용 출력의 특정 비트의 On/Off 상태를 설정한다.

Format

C

```
DWORD AxmSignalWriteOutputBit(long lAxisNo, long lBitNo, DWORD uOnOff);
```

Visual Basic

```
Function AxmSignalWriteOutputBit (ByVal lAxisNo As Long, ByVal lBitNo As Long, ByVal uOnOff As Long) As Long
```

Delphi

```
function AxmSignalWriteOutputBit (lAxisNo : LongInt; lBitNo : LongInt; uOnOff : DWord) : DWord; stdcall;
```

Input / Output

Name	in/out	Format	Init Value	Explanation
AxisNo	in	long	-	축번호(0 ~ (최대축수 - 1))
BitNo	in	long	-	범용 출력 비트 번호
OnOff	In	DWORD	-	출력 On/Off

BitNo

#define	Value	Explanation
UIO_OUT0	00h	서보온 출력 (SVON)
UIO_OUT1	01h	알람 클리어 출력 (ALARMCLEAR)
UIO_OUT2	02h	범용 출력 2
UIO_OUT3	03h	범용 출력 3
UIO_OUT4	04h	범용 출력 4

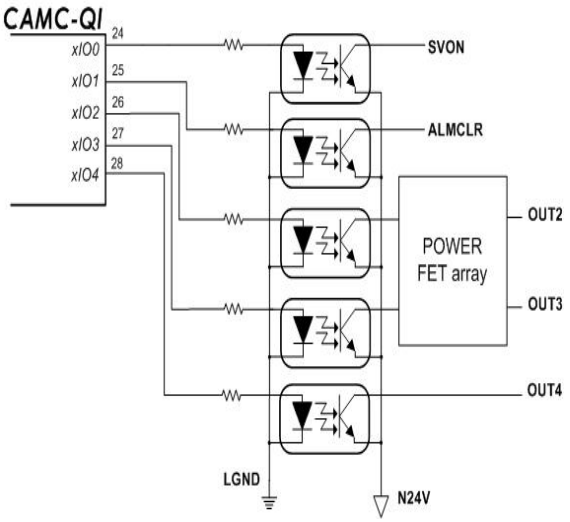
OnOff

#define	Value	Explanation
FALSE	00h	출력 Off (0)
TRUE	01h	출력 On (1)

Return AXT_RT_SUCCESS(0000) : Successful execution of API
* See error code Table more information on status error codes

Description

모듈의 범용 출력으로는 0 – 4까지 있으며 기본으로 범용(0)를 서보온으로 쓰며, 범용(1)은 알람 클리어 출력신호로 사용한다. AxmSignalWriteOutput는 동시에 여러 출력을 사용할 때 이 함수를 이용하고 한 개의 출력을 이용할 때는 AxmSignalWriteOutputBit 함수를 이용한다.



C Example

```
// 0축의 범용 출력 신호 4 ON.
AxmSignalWriteOutputBit(0, 4, 1);

// 0축의 범용 출력 신호 4 확인.
DWORD uOn;
AxmSignalReadOutputBit(0, 4, &uOn);
```

VB Example

```
‘ 0축의 범용 출력 신호 4 ON.
AxmSignalWriteOutputBit 0, 4, 1

‘ 0축의 범용 출력 신호 4 확인.
Dim uOn As Long
AxmSignalReadOutputBit 0, 4, uOn
```

Delphi Example

```
{ 0축의 범용 출력 신호 4 확인. }
var
uOn: DWord;

Begin
{ 0축의 범용 출력 신호 4 ON. }
AxmSignalWriteOutputBit(0, 4, 1);

AxmSignalReadOutputBit (0, 4, @uOn);
end;
```

See Also

AxmSignalServoOn, AxmSignalIsServoOn, AxmSignalServoAlarmReset, AxmSignalWriteOutput, AxmSignalReadOutput, AxmSignalReadOutputBit, AxmSignalReadInput, AxmSignalReadInputBit

AxmSignalReadOutputBit

Purpose

지정축의 범용 출력의 특정 비트의 On/Off 상태를 반환한다.

Format

C

```
DWORD AxmSignalReadOutputBit(long lAxisNo, long lBitNo, DWORD *upOnOff);
```

Visual Basic

```
Function AxmSignalReadOutputBit(ByVal lAxisNo As Long, ByVal lBitNo As Long, ByRef upOnOff
As Long) As Long
```

Delphi

```
function AxmSignalReadOutputBit(lAxisNo : LongInt; lBitNo : LongInt; upOnOff : PDWord) : DWord;
stdcall;
```

Input / Output

Name	in/out	Format	Init Value	Explanation
AxisNo	in	long	-	축번호(0 ~ (최대축수 - 1))
BitNo	in	long	-	범용 출력 비트 번호
OnOff	out	DWORD*	-	출력 On/Off

BitNo

#define	Value	Explanation
UIO_OUT0	00h	서보온 출력 (SVON)
UIO_OUT1	01h	알람 클리어 출력 (ALARMCLEAR)
UIO_OUT2	02h	범용 출력 2
UIO_OUT3	03h	범용 출력 3
UIO_OUT4	04h	범용 출력 4

OnOff

#define	Value	Explanation
FALSE	00h	출력 Off (0)
TRUE	01h	출력 On (1)

Return AXT_RT_SUCCESS(0000) : Successful execution of API
* See error code Table more information on status error codes

Description

'AxmSignalWriteOutputBit' 로 설정한 범용 출력값을 반환한다.

C Example

```
// 0축의 범용 출력 신호 4 ON.
AxmSignalWriteOutputBit(0, 4, 1);

// 0축의 범용 출력 신호 4 확인.
DWORD uOn;
AxmSignalReadOutputBit(0, 4, &uOn);
```

VB Example

```
' 0축의 범용 출력 신호 4 ON.
AxmSignalWriteOutputBit 0, 4, 1

' 0축의 범용 출력 신호 4 확인.
Dim uOn As Long
AxmSignalReadOutputBit 0, 4, uOn
```

Delphi Example

```
{ 0축의 범용 출력 신호 4 확인. }
var
uOn: DWord;

Begin
{ 0축의 범용 출력 신호 4 ON. }
AxmSignalWriteOutputBit(0, 4, 1);

AxmSignalReadOutputBit (0, 4, @uOn);
end;
```

See Also

AxmSignalServoOn, AxmSignalIsServoOn, AxmSignalServoAlarmReset, AxmSignalWriteOutput, AxmSignalReadOutput, AxmSignalWriteOutputBit, AxmSignalReadInput, AxmSignalReadInputBit

AxmSignalReadInput

Purpose

지정축의 범용 입력값을 Hex값으로 반환한다.

Format

C

```
DWORD AxmSignalReadInput (long lAxisNo, DWORD *upValue);
```

Visual Basic

```
Function AxmSignalReadInput (lAxisNo : LongInt; upValue : PDWord) : DWord; As Long
```

Delphi

```
function AxmSignalReadInput (lAxisNo: Long, upValue: PLong) : Dword ; stdcall;
```

Input / Output

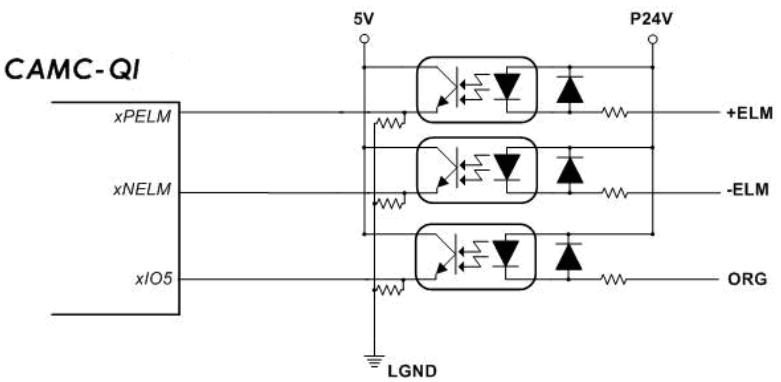
Name	in/out	Format	Init Value	Explanation
AxisNo	in	long	-	축번호(0 ~ (최대축수 - 1))
Value	Out	DWORD	-	범용 입력값

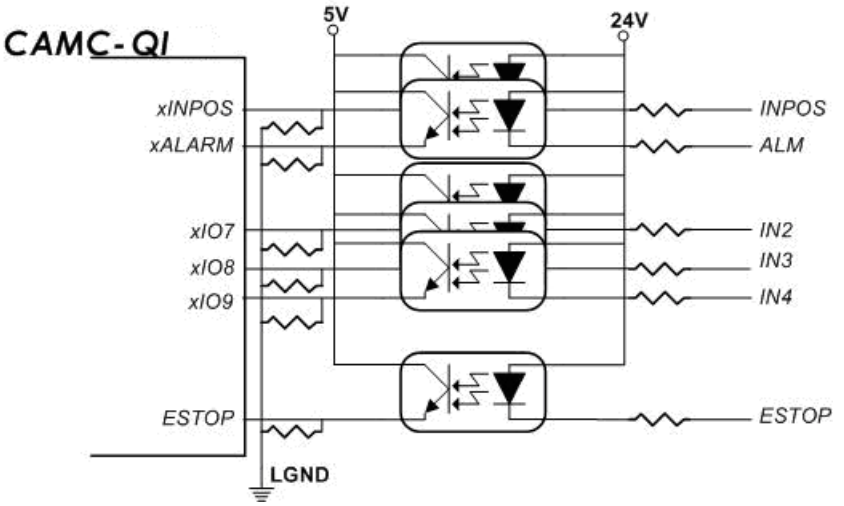
Return AXT_RT_SUCCESS(0000) : Successful execution of API

* See error code Table more information on status error codes

Description

지정축의 범용 입력0,1,2,3을 Hex값으로 반환한다.





C Example

```
// 0축의 범용 입력 신호 확인.
DWORD uValue;
AxmSignalReadInput (0, &uValue);
```

VB Example

```
' 0축의 범용 입력 신호 확인.
Dim uValue As Long
AxmSignalReadInput 0, uValue
```

Delphi Example

```
{ 0축의 범용 입력 신호 확인. }
var
uValue: DWord;

begin
AxmSignalReadInput (0, @uValue);
end;
```

See Also

AxmSignalServoOn, AxmSignalIsServoOn, AxmSignalServoAlarmReset, AxmSignalWriteOutput, AxmSignalReadOutput, AxmSignalWriteOutputBit, AxmSignalReadOutputBit, AxmSignalReadInputBit

AxmSignalReadInputBit

Purpose

지정축의 범용 입력의 해당 비트 On/Off 상태를 반환한다.

Format

C

```
DWORD AxmSignalReadInputBit (long lAxisNo, long lBitNo, DWORD *upOn);
```

Visual Basic

```
Function AxmSignalReadInputBit (ByVal lAxisNo As Long, ByVal lBitNo As Long ,ByRef upOn As Long) As Long
```

Delphi

```
function AxmSignalReadInputBit (lAxisNo : LongInt; lBitNo : LongInt; upOn : PDWord) : Dword ; stdcall;
```

Input / Output

Name	in/out	Format	Init Value	Explanation
AxisNo	in	long	-	축번호(0 ~ (최대축수 - 1))
BitNo	in	long	-	범용 출력 비트 번호
On	in	DWORD*	-	입력 비트의 On/Off 상태

BitNo

#define	Value	Explanation
UIO_INP0	00h	원점 센서 입력
UIO_INP1	01h	Z상 입력
UIO_INP2	02h	범용 입력 2
UIO_INP3	03h	범용 입력 3
UIO_INP4	04h	범용 입력 4

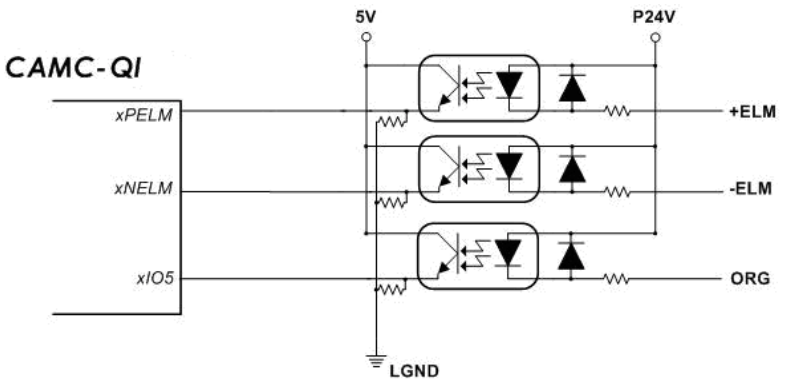
On

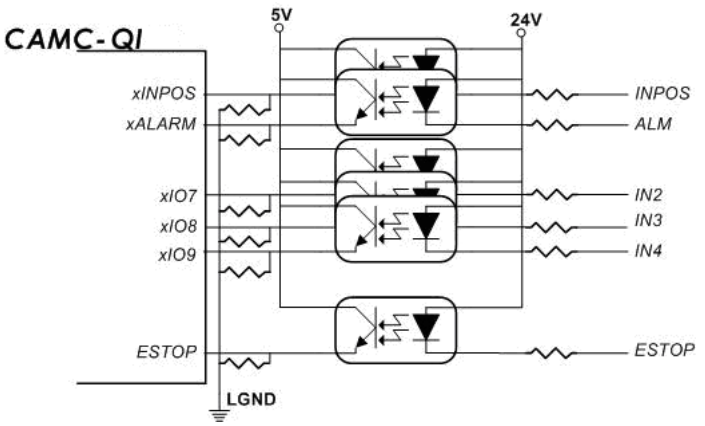
#define	Value	Explanation
FALSE	00h	출력 Off (0)
TRUE	01h	출력 On (1)

Return AXT_RT_SUCCESS(0000) : Successful execution of API
* See error code Table more information on status error codes

Description

지정축의 범용 입력의 해당 비트 Bit 값으로 반환한다.





C Example

```
// 0축의 범용 입력 4의 값을 확인.
DWORD uOn;
AxmSignalReadInputBit(0, 4, &uOn);
```

VB Example

```
' 0축의 범용 입력 4의 값을 확인.
Dim uOn As Long
AxmSignalReadInputBit 0, 4, uOn
```

Delphi Example

```
{ 0축의 범용 입력 4의 값을 확인. }
var
uOn: DWord;

begin
AxmSignalReadInputBit (0, 4, @uOn);
end;
```

See Also

AxmSignalServoOn, AxmSignalIsServoOn, AxmSignalServoAlarmReset, AxmSignalWriteOutput, AxmSignalReadOutput, AxmSignalWriteOutputBit, AxmSignalReadOutputBit

모션 구동 후에 상태 확인 함수

이 단원에서는 모션제어의 상태에 관련된 함수들에 대하여 설명한다. 상태 함수들은 모션의 상태를 감시하는데 필요한 함수들을 그룹화한 것이다. 모션의 상태에는 속도,가속도,감속도,위치, 모션 중인지 등을 체크하는 것을 포함하며 현재 진행되고 있는 Drive 진행상태, 모션의 I/O상태 , 정지되어졌을 때 왜 정지됐는지에 대한 상태, Command와 Actual값의 차이도 상태함수를 이용하여 확인할 수 있다.

Function	Description
AxmStatusReadInMotion	지정 축의 펄스 출력 상태를 반환한다.
AxmStatusReadDrivePulseCount	지정 축의 구동 펄스 카운터 값을 반환한다.
AxmStatusReadMotion	지정 축의 DriveStatus 레지스터를 반환한다.
AxmStatusReadStop	지정 축의 EndStatus 레지스터를 반환한다.
AxmStatusReadMechanical	지정 축의 Mechanical Signal Data를 반환한다.
AxmStatusReadVel	지정 축의 현재 구동 속도를 읽어온다.
AxmStatusReadPosError	지정 축의 Comand Position과 Actual Position의 차를 반환한다.
AxmStatusReadDriveDistance	최종 드라이브로 이동하는(이동한) 거리를 확인 한다
AxmStatusSetActPos	지정 축의 Act 위치를 설정한다.
AxmStatusGetActPos	지정 축의 Act 위치를 반환한다.
AxmStatusSetCmdPos	지정 축의 Cmd 위치를 설정한다.
AxmStatusGetCmdPos	지정 축의 Cmd 위치를 반환한다.
AxmStatusSetCmdComparatorPos	내부 위치 비교기에 값을 설정한다.(Pos = Unit)
AxmStatusGetCmdComparatorPos	내부 위치 비교기에 값을 반환한다.(Pos = Unit)

AxmStatusReadInMotion

Purpose

지정 축에 대하여 모션이 완료됐는지를 반환한다.

Format

C

```
DWORD AxmStatusReadInMotion (long lAxisNo, DWORD *upStatus);
```

Visual Basic

```
Function AxmStatusReadInMotion (ByVal lAxisNo As Long, ByRef upStatus As Long) As Long
```

Delphi

```
function AxmStatusReadInMotion (lAxisNo : LongInt; upStatus : PDWord) : DWord; stdcall;
```

Input / Output

Name	in/out	Format	Init Value	Explanation
AxisNo	in	long	-	축번호(0 ~ (최대축수 - 1))
Status	Out	DWORD*	-	모션 구동 상태

Status

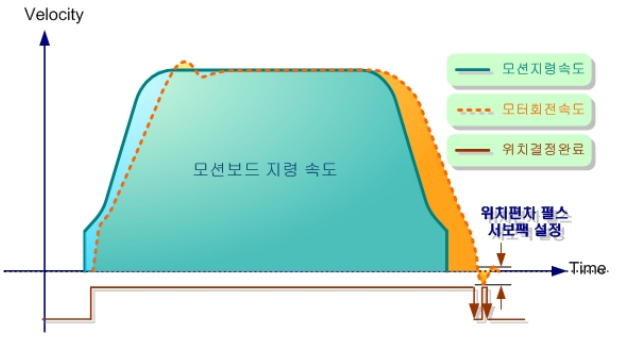
#define	Value	Explanation
FALSE	00h	모션 구동 하지 않음
TRUE	01h	모션 구동 중

Return AXT_RT_SUCCESS(0000) : Successful execution of API

* See error code Table more information on status error codes

Description

해당 축이 모션 구동 중이면 *upStatus = 1이 출력되고, 모션 구동 중이 아니면 *upStatus = 0이 출력된다.



▶ AxmSignalSetInpos함수를 사용하는 경우에는 Inpos 입력신호가 Enable로 설정되어 있다면 Command 펄스 출력이 완료해도 INP 입력이 ON 되기 전까지는 모션이 완료되지 않은 것으로 간주되어 반환 되지 않는다. 이 내용은 AxmSignalSetInpos함수를 참고하기 바란다.

C Example

```
// 0 축의 구동 상태를 확인
DWORD Status;
AxmStatusReadInMotion(0, &Status);
if (Status == FALSE)
 AfxMessageBox("모터가 정지하였습니다.");
```

VB Example

```
' 0 축의 구동 상태를 확인
Dim Status As Long
AxmStatusReadInMotion 0, Status
If Status = FALSE Then
 MsgBox "모터가 정지하였습니다.", vbOKCancel
End If
```

Delphi Example

```
{ 0 축의 구동 상태를 확인 }
var
Status: DWord;

begin
AxmStatusReadInMotion(0, @Status);
if (Status = 0) then
 Application.MessageBox('모터가 정지하였습니다.', 'Ajinextek', MB_OK);
end;
```

See Also

AxmStatusReadDrivePulseCount, AxmStatusReadMotion, AxmStatusReadStop,

AxmStatusReadMechanical, AxmStatusReadVel, AxmStatusReadPosError, AxmStatusReadDriveDistance, AxmStatusSetActPos, AxmStatusGetActPos, AxmStatusSetCmdPos, AxmStatusGetCmdPos,

AxmStatusReadDrivePulseCount

Purpose

지정 한 축의 현재 구동 시작부터 종료 시점까지의 이동량을 펄스 단위 값으로 반환한다.

Format

C

DWORD AxmStatusReadDrivePulseCount (long lAxisNo, long *lpPulse);

Visual Basic

Function AxmStatusReadDrivePulseCount (ByVal lAxisNo As Long, ByRef lpPulse As Long) As Long

Delphi

function AxmStatusReadDrivePulseCount (lAxisNo : LongInt; lpPulse ; PLongInt) : DWord; stdcall;

Input / Output

Name	in/out	Format	Init Value	Explanation
AxisNo	in	long	-	축번호(0 ~ (최대축수 - 1))
Pulse	Out	long*	-	구동한 펄스 카운터 값

Return AXT_RT_SUCCESS(0000) : Successful execution of API

* See error code Table more information on status error codes

Description

지정 축의 현재 구동 중인 모션의 시작부터 종료시점까지의 펄스 카운터 값을 반환한다. 모션 구동 종료 후 이전에 출력한 펄스카운터 값을 유지하다가, 새로운 모션 구동이 시작되면 펄스카운터 값이 갱신된다.

주의사항: 구동중에만 카운터값을 표시하고 구동종료후에는 카운터값이 CLEAR된다.

C Example

```
// 0축의 펄스 카운트를 가져온다.
long   Pulse;
AxmStatusReadDrivePulseCount (0, &Pulse);
```

VB Example

```
' 0축의 펄스 카운트를 가져온다.
Dim Pulse As Long
AxmStatusReadDrivePulseCount 0, Pulse
```

Delphi Example

```
{ 0축의 펄스 카운트를 가져온다. }
var
Pulse: LongInt;

begin
AxmStatusReadDrivePulseCount (0, @Pulse);
end;
```

See Also

AxmStatusReadInMotion, AxmStatusReadMotion, AxmStatusReadStop, AxmStatusReadMechanical, AxmStatusReadVel, AxmStatusReadPosError, AxmStatusReadDriveDistance, AxmStatusSetActPos, AxmStatusGetActPos, AxmStatusSetCmdPos, AxmStatusGetCmdPos

AxmStatusReadMotion

Purpose

지정 축의 모션 구동 상태 값을 반환한다.

Format

C

```
DWORD AxmStatusReadMotion (long lAxisNo, DWORD *upStatus);
```

Visual Basic

```
Function AxmStatusReadMotion (ByVal lAxisNo As Long, ByRef upStatus As Long) As Long
```

Delphi

```
function AxmStatusReadMotion (lAxisNo : LongInt; upStatus : PDWord) : DWord; stdcall;
```

Input / Output

Name	in/out	Format	Init Value	Explanation
AxisNo	in	long	-	축번호(0 ~ (최대축수 - 1))
Status	out	DWORD*	-	지정 축의 모션 구동 상태

Return AXT_RT_SUCCESS(0000) : Successful execution of API

* See error code Table more information on status error codes

Description.

사용자가 지정한 축의 모션 구동 상태를 반환한다. 구동 상태의 자세한 값은 아래의 Table과 같다.

주의사항 : QIDRIVE_STATUS_7 은 현재 라이브러리 에서 구현된 HOME함수와 상관없이 동작되며 이 비트는 칩에 홈레지스터를 이용한 홈구현시 사용되므로 사용자는 무시한다.

QIDRIVE_STATUS_8 이 현재 라이브러리 에서 구현된 HOME함수와 같이 동작된다.

홈시작함수(AxmHomeSetStart)가 신호검색 함수(AxmMoveSignalSearch)를 여러번사용하여 구현이 되어있으며 홈시작함수 사용시 QIDRIVE_STATUS_8 BIT: 1 된다.)

Status

축의 구동 상태

#define	Return Value	Explanation
QIDRIVE_STATUS_0	00000001h	Bit 0, BUSY(드라이브 구동 중)

QIDRIVE_STATUS_1	00000002h	Bit 1, DOWN(감속 중)
QIDRIVE_STATUS_2	00000004h	Bit 2, CONST(등속 중)
QIDRIVE_STATUS_3	00000008h	Bit 3, UP(가속 중)
QIDRIVE_STATUS_4	00000010h	Bit 4, 연속 드라이브 구동 중
QIDRIVE_STATUS_5	00000020h	Bit 5, 지정 거리 드라이브 구동 중
QIDRIVE_STATUS_6	00000040h	Bit 6, MPG 드라이브 구동 중
QIDRIVE_STATUS_7	00000080h	Bit 7, 원점검색 드라이브 구동중
QIDRIVE_STATUS_8	00000100h	Bit 8, 신호 검색 드라이브 구동 중
QIDRIVE_STATUS_9	00000200h	Bit 9, 보간 드라이브 구동 중
QIDRIVE_STATUS_10	00000400h	Bit 10, Slave 드라이브 구동중
QIDRIVE_STATUS_11	00000800h	Bit 11, 현재 구동 드라이브 방향(보간 드라이브에서는 표시 정보 다름)
QIDRIVE_STATUS_12	00001000h	Bit 12, 펄스 출력후 서보위치 완료 신호 대기중.
QIDRIVE_STATUS_13	00002000h	Bit 13, 직선 보간 드라이브 구동중.
QIDRIVE_STATUS_14	00004000h	Bit 14, 원호 보간 드라이브 구동중.
QIDRIVE_STATUS_15	00008000h	Bit 15, 펄스 출력 중.
QIDRIVE_STATUS_16	00010000h	Bit 16, 구동 예약 데이터 개수(처음)(0-7)
QIDRIVE_STATUS_17	00020000h	Bit 17, 구동 예약 데이터 개수(중간)(0-7)
QIDRIVE_STATUS_18	00040000h	Bit 18, 구동 예약 데이터 갯수(끝)(0-7)
QIDRIVE_STATUS_19	00080000h	Bit 19, 구동 예약 Queue 비어 있음.
QIDRIVE_STATUS_20	00100000h	Bit 20, 구동 예약 Queue 가득 찮
QIDRIVE_STATUS_21	00200000h	Bit 21, 현재 구동 드라이브의 속도 모드(처음)
QIDRIVE_STATUS_22	00400000h	Bit 22, 현재 구동 드라이브의 속도 모드(끝)
QIDRIVE_STATUS_23	00800000h	Bit 23, MPG 버퍼 #1 Full

QIDRIVE_STATUS_24	01000000h	Bit 24, MPG 버퍼 #2 Full
QIDRIVE_STATUS_25	02000000h	Bit 25, MPG 버퍼 #3 Full
QIDRIVE_STATUS_26	04000000h	Bit 26, MPG 버퍼 데이터 OverFlow

C Example

```
DWORD drivestatus;

// 0축의 구동 상태를 읽어온다.
AxmStatusReadMotion (0, &drivestatus);

switch(drivestatus)
{
case IPDRIVE_STATUS_BUSY:
 AfxMessageBox("0축이 구동 중입니다.");
 break;

case IPDRIVE_STATUS_DOWN:
 AfxMessageBox("0축이 감속 중입니다.");
 break;

case IPDRIVE_STATUS_CONST:
 AfxMessageBox("0축이 정속 중입니다.");
 break;

case IPDRIVE_STATUS_UP:
 AfxMessageBox("0축이 가속 중입니다.");
 break;

case IPDRIVE_STATUS_ICL:
 AfxMessageBox("0축이 ICL 상태입니다.");
 break;

case IPDRIVE_STATUS_ICG:
 AfxMessageBox("0축이 ICG 상태입니다.");
 break;

case IPDRIVE_STATUS_ECL:
 AfxMessageBox("0축이 ECL 상태입니다.");
 break;

case IPDRIVE_STATUS_ECG:
 AfxMessageBox("0축이 ECC 상태입니다.");
 break;

case IPDRIVE_STATUS_DRIVE_DIRECTION:
 AfxMessageBox("드라이브 방향 신호(0=CW/1=CCW)");
 break;
```

```
case IPDRIVE_STATUS_COMMAND_BUSY:
 AfxMessageBox("명령어 디코딩 중");
 break;

case IPDRIVE_STATUS_PRESET_DRIVING:
 AfxMessageBox("지정 펄스 수 드라이브 구동 중");
 break;

case IPDRIVE_STATUS_CONTINUOUS_DRIVING:
 AfxMessageBox("연속 구동 드라이브 중");
 break;

case IPDRIVE_STATUS_SIGNAL_SEARCH_DRIVING:
 AfxMessageBox("신호 검출 드라이브 구동 중");
 break;

case IPDRIVE_STATUS_ORG_SEARCH_DRIVING:
 AfxMessageBox("원점 검색 드라이브 구동 중");
 break;

case IPDRIVE_STATUS_MPG_DRIVING:
 AfxMessageBox("MPG 드라이브 구동 중");
 break;

case IPDRIVE_STATUS_SENSOR_DRIVING:
 AfxMessageBox("센서 위치 드라이브 구동 중");
 break;

case IPDRIVE_STATUS_L_C_INTERPOLATION:
 AfxMessageBox("직선/원호 보간 드라이브 중");
 break;

case IPDRIVE_STATUS_INTERRUPT_BANK1:
 AfxMessageBox("BANK1 에서 인터럽트 발생");
 break;

case IPDRIVE_STATUS_INTERRUPT_BANK2:
 AfxMessageBox("BANK2 에서 인터럽트 발생");
 break;

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

VB Example

```
Dim drivestatus As Long

' 0축의 구동 상태를 읽어온다.
AxmStatusReadMotion 0, drivestatus

Select Case drivestatus
```

```
Case IPDRIVE_STATUS_BUSY
 MsgBox “0축이 구동 중입니다.”

Case IPDRIVE_STATUS_DOWN
 MsgBox “0축이 감속 중입니다.”

Case IPDRIVE_STATUS_CONST
 MsgBox “0축이 정속 중입니다.”

Case IPDRIVE_STATUS_UP
 MsgBox “0축이 가속 중입니다.”

Case IPDRIVE_STATUS_ICL
 MsgBox “0축이 ICL 상태입니다.”

Case IPDRIVE_STATUS_ICG
 MsgBox “0축이 ICG 상태입니다.”

Case IPDRIVE_STATUS_ECL
 MsgBox “0축이 ECL 상태입니다.”

Case IPDRIVE_STATUS_ECG
 MsgBox “0축이 ECC 상태입니다.”

Case IPDRIVE_STATUS_DRIVE_DIRECTION
 MsgBox “드라이브 방향 신호(0=CW/1=CCW)”

Case IPDRIVE_STATUS_COMMAND BUSY
 MsgBox “명령어 디코딩 중”

Case IPDRIVE_STATUS_PRESET_DRIVING
 MsgBox “지정 펄스 수 드라이브 구동 중”

Case IPDRIVE_STATUS_CONTINUOUS_DRIVING
 MsgBox “연속 구동 드라이브 중”

Case IPDRIVE_STATUS_SIGNAL_SEARCH_DRIVING
 MsgBox “신호 검출 드라이브 구동 중”

Case IPDRIVE_STATUS_ORG_SEARCH_DRIVING
 MsgBox “원점 검색 드라이브 구동 중”

Case IPDRIVE_STATUS_MPG_DRIVING
 MsgBox “MPG 드라이브 구동 중”

Case IPDRIVE_STATUS_SENSOR_DRIVING
 MsgBox “센서 위치 드라이브 구동 중”

Case IPDRIVE_STATUS_L_C_INTERPOLATION
 MsgBox “직선/원호 보간 드라이브 중”

Case IPDRIVE_STATUS_INTERRUPT_BANK1
 MsgBox “BANK1 에서 인터럽트 발생”
```

```
Case IPDRIVE_STATUS_INTERRUPT_BANK2
 MsgBox "BANK2 에서 인터럽트 발생"

Case Else
 MsgBox "알 수 없는 데이터 입니다."
End Select
```

Delphi Example

```
var
drivestatus: DWord;

begin
{ 0축의 구동 상태를 읽어온다. }
AxmStatusReadMotion (0, @drivestatus);

begin
case drivestatus of
IPDRIVE_STATUS_BUSY:
Application.MessageBox ('0축이 구동 중입니다.', 'Ajinextek', MB_OK);

IPDRIVE_STATUS_DOWN:
Application.MessageBox('0축이 감속 중입니다.', 'Ajinextek', MB_OK);

IPDRIVE_STATUS_CONST:
Application.MessageBox('0축이 정속 중입니다.', 'Ajinextek', MB_OK);

IPDRIVE_STATUS_UP:
Application.MessageBox ('0축이 가속 중입니다.', 'Ajinextek', MB_OK);

IPDRIVE_STATUS_ICL:
Application.MessageBox ('0축이 ICL 상태입니다.', 'Ajinextek', MB_OK);

IPDRIVE_STATUS_ICG:
Application.MessageBox ('0축이 ICG 상태입니다.', 'Ajinextek', MB_OK);

IPDRIVE_STATUS_ECL:
Application.MessageBox ('0축이 ECL 상태입니다.', 'Ajinextek', MB_OK);

IPDRIVE_STATUS_ECG:
Application.MessageBox ('0축이 ECC 상태입니다.', 'Ajinextek', MB_OK);

IPDRIVE_STATUS_DRIVE_DIRECTION:
Application.MessageBox ('드라이브 방향 신호(0=CW/1=CCW)', 'Ajinextek', MB_OK);

IPDRIVE_STATUS_COMMAND_BUSY:
Application.MessageBox ('명령어 디코딩 중', 'Ajinextek', MB_OK);

IPDRIVE_STATUS_PRESET_DRIVING:
Application.MessageBox ('지정 펄스 수 드라이브 구동 중', 'Ajinextek', MB_OK);

IPDRIVE_STATUS_CONTINUOUS_DRIVING:
Application.MessageBox ('연속 구동 드라이브 중', 'Ajinextek', MB_OK);
```

```
IPDRIVE_STATUS_SIGNAL_SEARCH_DRIVING:
Application.MessageBox (‘신호 검출 드라이브 구동 중’, ‘Ajinextek’, MB_OK);

IPDRIVE_STATUS_ORG_SEARCH_DRIVING:
Application.MessageBox (‘원점 검색 드라이브 구동 중’, ‘Ajinextek’, MB_OK);

IPDRIVE_STATUS_MPG_DRIVING:
Application.MessageBox (‘MPG 드라이브 구동 중’, ‘Ajinextek’, MB_OK);

IPDRIVE_STATUS_SENSOR_DRIVING:
Application.MessageBox (‘센서 위치 드라이브 구동 중’, ‘Ajinextek’, MB_OK);

IPDRIVE_STATUS_L_C_INTERPOLATION:
Application.MessageBox (‘직선/원호 보간 드라이브 중’, ‘Ajinextek’, MB_OK);

IPDRIVE_STATUS_INTERRUPT_BANK1:
Application.MessageBox (‘BANK1 에서 인터럽트 발생’, ‘Ajinextek’, MB_OK);

IPDRIVE_STATUS_INTERRUPT_BANK2:
Application.MessageBox (‘BANK2 에서 인터럽트 발생’, ‘Ajinextek’, MB_OK);

else Application.MessageBox(‘알 수 없는 데이터 입니다.’, ‘Ajinextek’, MB_OK);
end;
end;
```

See Also

AxmStatusReadInMotion, AxmStatusReadDrivePulseCount, AxmStatusReadStop, AxmStatusReadMechanical, AxmStatusReadVel, AxmStatusReadPosError, AxmStatusReadDriveDistance, AxmStatusSetActPos, AxmStatusGetActPos, AxmStatusSetCmdPos, AxmStatusGetCmdPos,

AxmStatusReadStop

Purpose

지정 축의 모션 종료에 대한 상태 값을 반환한다.

Format

C

```
DWORD AxmStatusReadStop (long lAxisNo, DWORD *upStatus);
```

Visual Basic

```
Function AxmStatusReadStop (ByVal lAxisNo As Long, ByRef upStatus As Long) As Long
```

Delphi

```
function AxmStatusReadStop (lAxisNo : LongInt; upStatus : PDWord) : Dword ; stdcall;
```

Input / Output

Name	in/out	Format	Init Value	Explanation
AxisNo	in	long	-	축번호(0 ~ (최대축수 - 1))
Status	out	DWORD*	-	축의 모션 종료 상태값

Return AXT_RT_SUCCESS(0000) : Successful execution of API

* See error code Table more information on status error codes

Description

지정 축의 모션 종료에 대한 상태 값을 반환한다. 이 함수를 이용하면 모션이 정상종료 하였는지, 외부 센서나 정지 명령 또는 모션 에러와 같은 이유로 정지하였는지에 대한 정보를 확인할 수 있다. 종료 상태 값은 아래의 Table과 같다.

Status

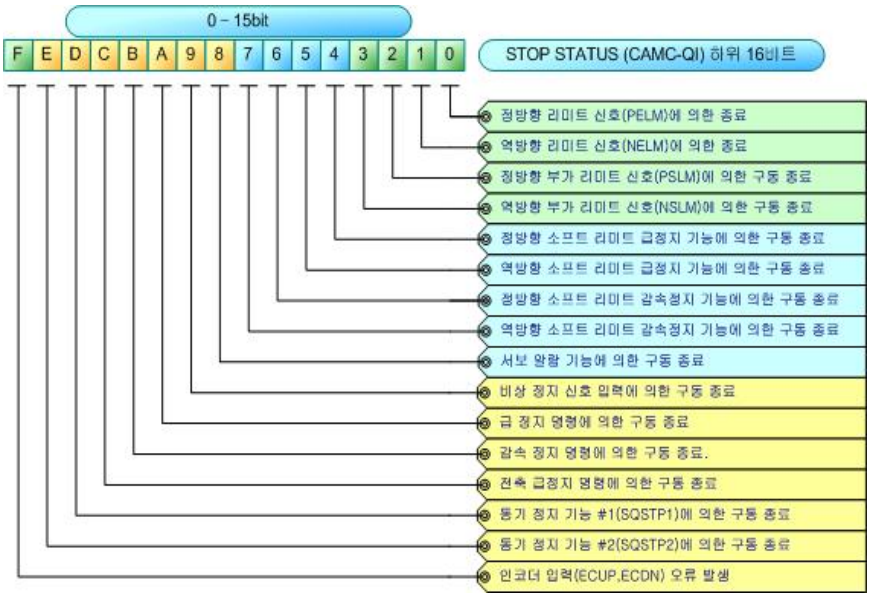
축의 종료 상태

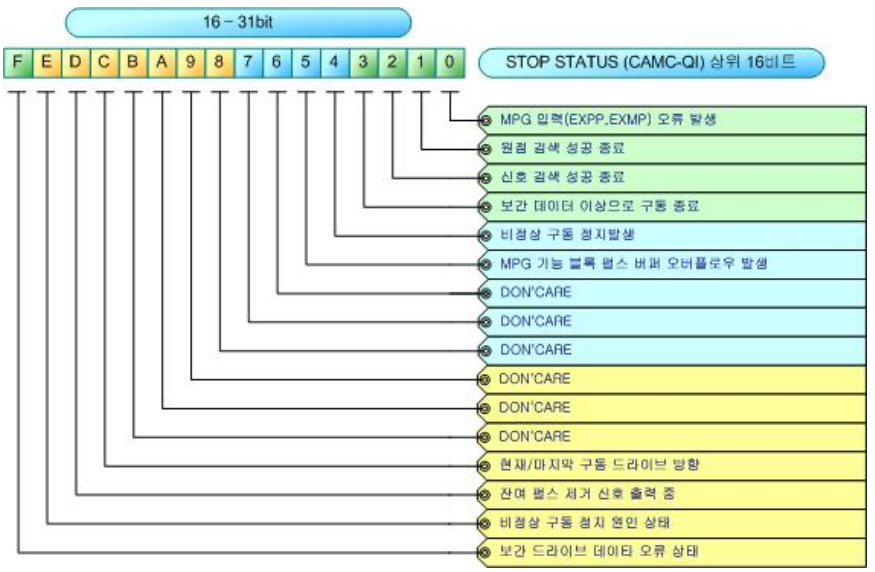
#define	Value	Explanation
	00000000h	구동하지 않음 또는 초기 상태
QIEND_STATUS_0	00000001h	Bit 0, 정방향 리미트 신호(PELM)에 의한 종료
QIEND_STATUS_1	00000002h	Bit 1, 역방향 리미트 신호(NELM)에 의한 종료

QIEND_STATUS_2	00000004h	Bit 2, 정방향 부가 리미트 신호(PSLM)에 의한 구동 종료
QIEND_STATUS_3	00000008h	Bit 3, 역방향 부가 리미트 신호(NSLM)에 의한 구동 종료
QIEND_STATUS_4	00000010h	Bit 4, 정방향 소프트 리미트 급정지 기능에 의한 구동 종료
QIEND_STATUS_5	00000020h	Bit 5, 역방향 소프트 리미트 급정지 기능에 의한 구동 종료
QIEND_STATUS_6	00000040h	Bit 6, 정방향 소프트 리미트 감속정지 기능에 의한 구동 종료
QIEND_STATUS_7	00000080h	Bit 7, 역방향 소프트 리미트 감속정지 기능에 의한 구동 종료
QIEND_STATUS_8	00000100h	Bit 8, 서보 알람 기능에 의한 구동 종료.
QIEND_STATUS_9	00000200h	Bit 9, 비상 정지 신호 입력에 의한 구동 종료.
QIEND_STATUS_10	00000400h	Bit 10, 급 정지 명령에 의한 구동 종료
QIEND_STATUS_11	00000800h	Bit 11, 감속 정지 명령에 의한 구동 종료.
QIEND_STATUS_12	00001000h	Bit 12, 전축 급정지 명령에 의한 구동 종료
QIEND_STATUS_13	00002000h	Bit 13, 동기 정지 기능 #1(SQSTP1)에 의한 구동 종료
QIEND_STATUS_14	00004000h	Bit 14, 동기 정지 기능 #2(SQSTP2)에 의한 구동 종료.
QIEND_STATUS_15	00008000h	Bit 15, 인코더 입력(ECUP,ECDN) 오류 발생
QIEND_STATUS_16	00010000h	Bit 16, MPG 입력(EXPP,EXMP) 오류 발생
QIEND_STATUS_17	00020000h	Bit 17, 원점 검색 성공 종료

QIEND_STATUS_18	00040000h	Bit 18, 신호 검색 성공 종료.
QIEND_STATUS_19	00080000h	Bit 19, 보간 데이터 이상으로 구동 종료.
QIEND_STATUS_20	00100000h	Bit 20, 비정상 구동 정지발생.
QIEND_STATUS_21	00200000h	Bit 21, MPG 기능 블록 펄스 버퍼 오버플로우 발생
QIEND_STATUS_22 - 27		Bit 22,- Bit27 DON'CARE
QIEND_STATUS_28	10000000h	Bit 28, 현재/마지막 구동 드라이브 방향
QIEND_STATUS_29	20000000h	Bit 29, 잔여 펄스 제거 신호 출력 중
QIEND_STATUS_30	40000000h	Bit 30, 비정상 구동 정지 원인 상태
QIEND_STATUS_31	80000000h	Bit 31, 보간 드라이브 데이타 오류 상태.

Stop Status 각 Bit별 의미





C Example

```
DWORD    endstatus;

// 0축의 종료 상태를 읽어온다.
AxmStatusReadStop (0, &endstatus);

switch(endstatus)
{
case 0:
 AfxMessageBox("0축이 정상 종료 되었습니다.");
 break;

case IPEND_STATUS_SLM:
 AfxMessageBox("0축이 limit 감속 정지 신호 입력에 의하여 종료 되었습니다.");
 break;

case IPEND_STATUS_ELM:
 AfxMessageBox("0축이 limit 급 정지 신호 입력에 의하여 종료 되었습니다.");
 break;

case IPEND_STATUS_SSTOP_SIGNAL:
 AfxMessageBox("0축이 감속 정지 신호 입력에 의하여 종료 되었습니다.");
 break;

case IPEND_STATUS_ESTOP_SIGANL:
 AfxMessageBox("0축이 급 정지 신호 입력에 의하여 종료 되었습니다.");
 break;

case IPEND_STATUS_SSTOP_COMMAND:
 AfxMessageBox("0축이 감속 정지 명령에 의하여 종료 되었습니다.");
 break;

case IPEND_STATUS_ESTOP_COMMAND:
 AfxMessageBox("0축이 급 정지 정지 명령에 의하여 종료 되었습니다.");
 break;

case IPEND_STATUS_ALARM_SIGNAL:
 AfxMessageBox("0축이 Alarm 신호 입력에 의하여 종료 되었습니다.");
 break;

case IPEND_STATUS_DATA_ERROR:
AfxMessageBox("0축이 데이터 설정 에러에 의하여 종료 되었습니다. ");
 break;

case IPEND_STATUS_DEVIATION_ERROR:
 AfxMessageBox("0축이 탈조 에러에 의하여 종료 되었습니다. ");
 break;

case IPEND_STATUS_ORIGIN_DETECT:
 AfxMessageBox("0축이 원점 검출에 의하여 종료 되었습니다. ");
```

```
 break;

 case IPEND_STATUS_SIGNAL_DETECT:
 AfxMessageBox("0축이 신호 검출에 의한 의하여 종료 되었습니다. ");
 break;

 case IPEND_STATUS_PRESET_PULSE_DRIVE:
 AfxMessageBox("0축이 Preset pulse drive에 의하여 종료 되었습니다. ");
 break;

 case IPEND_STATUS_SENSOR_PULSE_DRIVE:
 AfxMessageBox("0축이Sensor pulse drive에 의하여 종료 되었습니다. ");
 break;

 case IPEND_STATUS_LIMIT:
 AfxMessageBox("0축이 Limit 완전 정지에 의하여 종료 되었습니다. ");
 break;

 case IPEND_STATUS_SOFTLIMIT:
 AfxMessageBox("0축이 Soft limit에 의하여 종료 되었습니다. ");
 break;

 case IPEND_STATUS_INTERPOLATION_DRIVE:
 AfxMessageBox("0축이 보간 드라이브를 종료 하였습니다.");
 break;

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

VB Example

```
Dim endstatus As Long

‘ 0축의 종료 상태를 읽어온다.
AxmStatusReadStop 0, endstatus

Select Case endstatus
Case 0
 MsgBox "0축이 정상 종료 되었습니다."

Case IPEND_STATUS_SLM
 MsgBox "0축이 limit 감속 정지 신호 입력에 의하여 종료 되었습니다."

Case IPEND_STATUS_ELM
 MsgBox "0축이 limit 급 정지 신호 입력에 의하여 종료 되었습니다."

Case IPEND_STATUS_SSTOP_SIGNAL
 MsgBox "0축이 감속 정지 신호 입력에 의하여 종료 되었습니다."

Case IPEND_STATUS_ESTOP_SIGANL
 MsgBox "0축이 급 정지 신호 입력에 의하여 종료 되었습니다."
```

```
Case IPEND_STATUS_SSTOP_COMMAND
 MsgBox "0축이 감속 정지 명령에 의하여 종료 되었습니다."

Case IPEND_STATUS_ESTOP_COMMAND
 MsgBox "0축이 급 정지 정지 명령에 의하여 종료 되었습니다."

Case IPEND_STATUS_ALARM_SIGNAL
 MsgBox "0축이 Alarm 신호 입력에 의하여 종료 되었습니다."

Case IPEND_STATUS_DATA_ERROR
 MsgBox "0축이 데이터 설정 에러에 의하여 종료 되었습니다."

Case IPEND_STATUS_DEVIATION_ERROR
 MsgBox "0축이 탈조 에러에 의하여 종료 되었습니다."

Case IPEND_STATUS_ORIGIN_DETECT
 MsgBox "0축이 원점 검출에 의하여 종료 되었습니다."

Case IPEND_STATUS_SIGNAL_DETECT
 MsgBox "0축이 신호 검출에 의한 의하여 종료 되었습니다."

Case IPEND_STATUS_PRESET_PULSE_DRIVE
 MsgBox "0축이 Preset pulse drive에 의하여 종료 되었습니다."

Case IPEND_STATUS_SENSOR_PULSE_DRIVE
 MsgBox "0축이Sensor pulse drive에 의하여 종료 되었습니다."

Case IPEND_STATUS_LIMIT
 MsgBox "0축이 Limit 완전 정지에 의하여 종료 되었습니다."

Case IPEND_STATUS_SOFTLIMIT
 MsgBox "0축이 Soft limit에 의하여 종료 되었습니다."

Case IPEND_STATUS_INTERPOLATION_DRIVE
 MsgBox "0축이 보간 드라이브를 종료 하였습니다."

Case Else
 MsgBox "알 수 없는 데이터 입니다."
End Select
```

Delphi Example

```
var
endstatus : Word;

begin
{ 0축의 종료 상태를 읽어온다. }
AxmStatusReadStop (0, @endstatus);

begin
case endstatus of
0:Application.MessageBox('0축이 정상 종료 되었습니다.', 'Ajinextek', MB_OK);
```

```
IPEND_STATUS_SLM:
Application.MessageBox('0축이 limit 감속 정지 신호 입력에 의하여 종료 되었습니다.',
       'Ajinextek', MB_OK);

IPEND_STATUS_ELM:
Application.MessageBox('0축이 limit 급 정지 신호 입력에 의하여 종료 되었습니다.',
       'Ajinextek', MB_OK);

IPEND_STATUS_SSTOP_SIGNAL:
Application.MessageBox('0축이 감속 정지 신호 입력에 의하여 종료 되었습니다.',
       'Ajinextek', MB_OK);

IPEND_STATUS_ESTOP_SIGANL:
Application.MessageBox('0축이 급 정지 신호 입력에 의하여 종료 되었습니다.',
       'Ajinextek', MB_OK);

IPEND_STATUS_SSTOP_COMMAND:
Application.MessageBox('0축이 감속 정지 명령에 의하여 종료 되었습니다.',
       'Ajinextek', MB_OK);

IPEND_STATUS_ESTOP_COMMAND:
Application.MessageBox('0축이 급 정지 정지 명령에 의하여 종료 되었습니다.',
       'Ajinextek', MB_OK);

IPEND_STATUS_ALARM_SIGNAL:
Application.MessageBox('0축이 Alarm 신호 입력에 의하여 종료 되었습니다.',
       'Ajinextek', MB_OK);

IPEND_STATUS_DATA_ERROR:
Application.MessageBox('0축이 데이터 설정 에러에 의하여 종료 되었습니다.',
       'Ajinextek', MB_OK);

IPEND_STATUS_DEVIATION_ERROR:
Application.MessageBox('0축이 탈조 에러에 의하여 종료 되었습니다.', 'Ajinextek',
       MB_OK);

IPEND_STATUS_ORIGIN_DETECT:
Application.MessageBox('0축이 원점 검출에 의하여 종료 되었습니다.', 'Ajinextek',
       MB_OK);

IPEND_STATUS_SIGNAL_DETECT:
Application.MessageBox('0축이 신호 검출에 의한 의하여 종료 되었습니다.',
       'Ajinextek', MB_OK);

IPEND_STATUS_PRESET_PULSE_DRIVE:
Application.MessageBox('0축이 Preset pulse drive에 의하여 종료 되었습니다.',
       'Ajinextek', MB_OK);

IPEND_STATUS_SENSOR_PULSE_DRIVE:
Application.MessageBox('0축이Sensor pulse drive에 의하여 종료 되었습니다.',
       'Ajinextek', MB_OK);

IPEND_STATUS_LIMIT:
Application.MessageBox('0축이 Limit 완전 정지에 의하여 종료 되었습니다.',
```

```
        'Ajinextek', MB_OK);

IPEND_STATUS_SOFTLIMIT:
Application.MessageBox('0축이 Soft limit에 의하여 종료 되었습니다.',
        'Ajinextek', MB_OK);

IPEND_STATUS_INTERPOLATION_DRIVE:
Application.MessageBox('0축이 보간 드라이브를 종료 하였습니다.', 'Ajinextek',
        MB_OK);

else Application.MessageBox('알 수 없는 데이터 입니다.', 'Ajinextek', MB_OK);
end;
end;
end;
```

See Also

AxmStatusReadInMotion, AxmStatusReadDrivePulseCount, AxmStatusReadMotion, AxmStatusReadMechanical, AxmStatusReadVel, AxmStatusReadPosError, AxmStatusReadDriveDistance, AxmStatusSetActPos, AxmStatusGetActPos, AxmStatusSetCmdPos, AxmStatusGetCmdPos,

AxmStatusReadMechanical

Purpose

지정 축의 외부 센서 및 모터 관련 신호들의 상태를 반환한다.

Format

C

```
DWORD AxmStatusReadMechanical (long lAxisNo, DWORD *upStatus);
```

Visual Basic

```
Function AxmStatusReadMechanical (ByVal lAxisNo As Long, ByRef upStatus As Long) As Long
```

Delphi

```
function AxmStatusReadMechanical (lAxisNo : LongInt; upStatus : PDWord) : Dword ; stdcall;
```

Input / Output

Name	in/out	Format	Init Value	Explanation
AxisNo	in	long	-	축번호(0 ~ (최대축수 - 1))
Status	out	DWORD*	-	외부 센서 및 모터 관련 신호 상태값

Return AXT_RT_SUCCESS(0000) : Successful execution of API

* See error code Table more information on status error codes

Description

지정 축의 외부 센서 및 모터 관련 신호들(Mechanical Signal)의 상태를 반환한다. 각각의 상태 신호들의 Active Level을 확인 할 수 있다.

사용자가 지정한 축의 Mechanical Signal의 상태 값은 아래의 Table과 같다

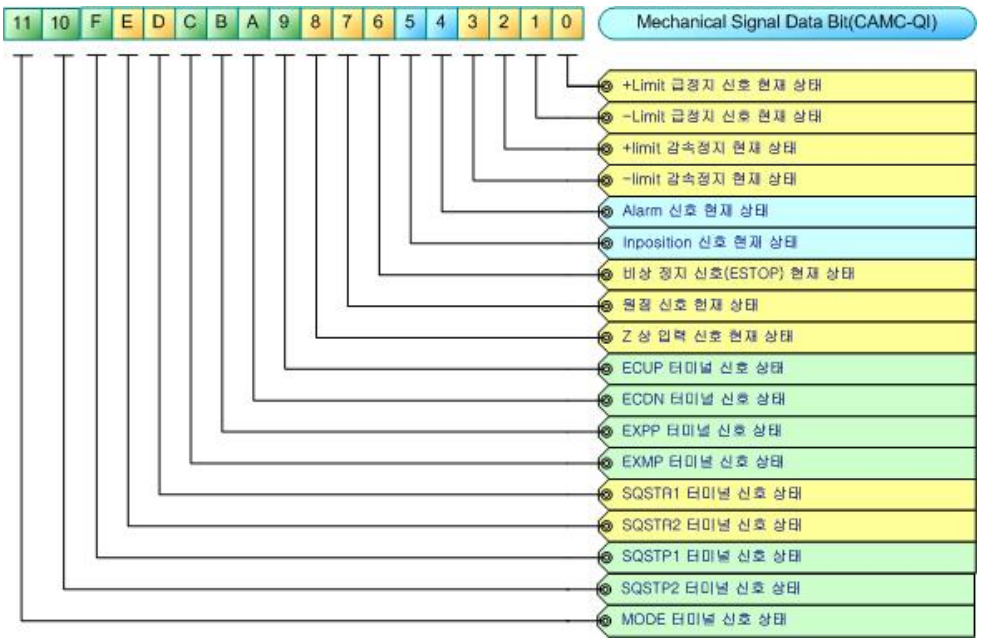
Status

Mechanical Signal 레지스터

#define	Value	Explanation
QIMECHANICAL_PELM_LEVEL	0001h	Bit 0, +Limit 급정지 신호 현재 상태
QIMECHANICAL_NELM_LEVEL	0002h	Bit 1, -Limit 급정지 신호 현재 상태
QIMECHANICAL_PSLM_LEVEL	0004h	Bit 2, +limit 감속정지 현재 상태.
QIMECHANICAL_NSLM_LEVEL	0008h	Bit 3, -limit 감속정지 현재 상태

QIMECHANICAL_ALARM_LEVEL	0010h	Bit 4, Alarm 신호 신호 현재 상태
QIMECHANICAL_INP_LEVEL	0020h	Bit 5, InPos 신호 현재 상태
QIMECHANICAL_ESTOP_LEVEL	0040h	Bit 6, 비상 정지 신호(ESTOP) 현재 상태.
QIMECHANICAL_ORG_LEVEL	0080h	Bit 7, 원점 신호 헌재 상태
QIMECHANICAL_ZPHASE_LEVEL	0100h	Bit 8, Z 상 입력 신호 현재 상태
QIMECHANICAL_ECUP_LEVEL	0200h	Bit 9, ECUP 터미널 신호 상태.
QIMECHANICAL_ECDN_LEVEL	0400h	Bit 10, ECDN 터미널 신호 상태.
QIMECHANICAL_EXPP_LEVEL	0800h	Bit 11, EXPP 터미널 신호 상태
QIMECHANICAL_EXMP_LEVEL	1000h	Bit 12, EXMP 터미널 신호 상태
QIMECHANICAL_SQSTR1_LEVEL	2000h	Bit 13, SQSTR1 터미널 신호 상태
QIMECHANICAL_SQSTR2_LEVEL	4000h	Bit 14, SQSTR2 터미널 신호 상태
QIMECHANICAL_SQSTP1_LEVEL	8000h	Bit 15, SQSTP1 터미널 신호 상태
QIMECHANICAL_SQSTP2_LEVEL	10000h	Bit 16, SQSTP2 터미널 신호 상태
QIMECHANICAL_MODE_LEVEL	20000h	Bit 17, MODE 터미널 신호 상태

Mechanical Signal 각 Bit별 의미



C Example

```
// 0축의 각각의 센서의 감지 상태를 읽어온다.
DWORD Status;
AxmStatusReadMechanical (0, &Status);
```

VB Example

```
‘ 0축의 각각의 센서의 감지 상태를 읽어온다.
Dim Status As Long
AxmStatusReadMechanical 0, Status
```

Delphi Example

```
{ 0축의 각각의 센서의 감지 상태를 읽어온다. }
var
Status: DWord;

begin
AxmStatusReadMechanical (0, @Status);
end;
```

See Also

AxmStatusReadInMotion, AxmStatusReadDrivePulseCount, AxmStatusReadMotion, AxmStatusReadStop, AxmStatusReadVel, AxmStatusReadPosError, AxmStatusReadDriveDistance, AxmStatusSetActPos, AxmStatusGetActPos, AxmStatusSetCmdPos, AxmStatusGetCmdPos

AxmStatusReadVel

Purpose

지정 축의 현재 구동 속도를 읽어온다.

Format

C

```
DWORD AxmStatusReadVel (long lAxisNo, double *dpVelocity);
```

Visual Basic

```
Function AxmStatusReadVel (ByVal lAxisNo As Long, ByRef dpVelocity As Double) As Long
```

Delphi

```
function AxmStatusReadVel (lAxisNo : LongInt; dpVelocity : PDouble) : DWord; stdcall;
```

Input / Output

Name	in/out	Format	Init Value	Explanation
AxisNo	in	long	-	축번호(0 ~ (최대축수 - 1))
Velocity	out	double*	-	현재 속도값

Return AXT_RT_SUCCESS(0000) : Successful execution of API

* See error code Table more information on status error codes

Description

사용자가 지정한 축의 현재 구동 속도를 읽어온다. 속도 값의 단위는 unit/sec이다.

C Example

```
double   velocity;

AxmMoveStartPos(0, 1000, 100, 200, 200);

// 지정축의 현재 구동 속도를 반환한다.
AxmStatusReadVel (0, &Velocity);
```

VB Example

```
Dim velocity As Double

AxmMoveStartPos 0, 1000, 100, 200, 200

‘ 지정축의 현재 구동 속도를 반환한다.
AxmStatusReadVel 0, Velocity
```

Delphi Example

```
var
Velocity : Double;

begin
AxmMoveStartPos(0, 1000, 100, 200, 200);

{ 지정축의 현재 구동 속도를 반환한다. }
AxmStatusReadVel (0, @Velocity);
end;
```

See Also

AxmStatusReadInMotion, AxmStatusReadDrivePulseCount, AxmStatusReadMotion, AxmStatusReadStop, AxmStatusReadMechanical, AxmStatusReadPosError, AxmStatusReadDriveDistance, AxmStatusSetActPos, AxmStatusGetActPos, AxmStatusSetCmdPos, AxmStatusGetCmdPos,

AxmStatusReadPosError

Purpose

지정 축의 Command Position과 Actual Position의 차를 반환한다.

Format

C

DWORD AxmStatusReadPosError (long lAxisNo, double *dpError);

Visual Basic

Function AxmStatusReadPosError (ByVal lAxisNo As Long, ByRef dpError As Double) As Long

Delphi

function AxmStatusReadPosError (lAxisNo : LongInt; dpError : PDouble) : DWord; stdcall;

Input / Output

Name	in/out	Format	Init Value	Explanation
AxisNo	in	long	-	축번호(0 ~ (최대축수 - 1))
Error	out	double*	-	Command Position과 Actual Position 차이 값

Return AXT_RT_SUCCESS(0000) : Successful execution of API

* See error code Table more information on status error codes

Description

사용자가 지정한 축의 제어기에서 나가는 출력 펄스가 제대로 나가는지 내부Command Position과 Actual Position(Encoder) 차이를 반환하여 알 수 있다. 스텝모터도 Encoder가 부가적으로 붙어있으면 알 수 있으며 Encoder가 없을 경우 이 함수를 사용할 수 없다.

C Example

```
// 0축의 Cmd Pos과 Act Pos의 차를 확인
double Error;
AxmStatusReadPosError (0, &Error);
```

VB Example

```
' 0축의 Cmd Pos과 Act Pos의 차를 확인
Dim Error As Double
AxmStatusReadPosError 0, Error
```

Delphi Example

```
{ 0축의 Cmd Pos과 Act Pos의 차를 확인 }
var
Error: Double;

begin
AxmStatusReadPosError (0, @Error);
end;
```

See Also

AxmStatusReadInMotion, AxmStatusReadDrivePulseCount, AxmStatusReadMotion, AxmStatusReadStop, AxmStatusReadMechanical, AxmStatusReadVel, AxmStatusReadDriveDistance, AxmStatusSetActPos, AxmStatusGetActPos, AxmStatusSetCmdPos, AxmStatusGetCmdPos,

AxmStatusReadDriveDistance

Purpose

지정 축의 현재 구동 중인 모션의 시작부터 종료시점까지의 위치 이동거리를 반환한다.

Format

C

```
DWORD AxmStatusReadDriveDistance (long lAxisNo, double *dpUnit);
```

Visual Basic

```
Function AxmStatusReadDriveDistance (ByVal lAxisNo As Long, ByRef dpUnit As Double ) As Long
```

Delphi

```
function AxmStatusReadDriveDistance (lAxisNo : LongInt; dpUnit : PDouble ) : DWord; stdcall;
```

Input / Output

Name	in/out	Format	Init Value	Explanation
AxisNo	in	long	-	축번호(0 ~ (최대축수 - 1))
Unit	out	double*	-	이동거리

Return AXT_RT_SUCCESS(0000) : Successful execution of API

* See error code Table more information on status error codes

Description

모션 구동 종료 후 이전에 이동한 거리 값을 유지하다가, 새로운 모션 구동이 시작되면 이동 거리 값이 갱신된다.

C Example

```
// 0 축에 현재 위치를 '0'으로 설정하고 출력되는 펄스를 확인
double Unit;
AxmStatusReadDriveDistance (0, &Unit);
```

VB Example

```
' 0 축에 현재 위치를 '0'으로 설정하고 출력되는 펄스를 확인
Dim Unit As Double
AxmStatusReadDriveDistance 0, Unit
```

Delphi Example

```
{ 0 축에 현재 위치를 '0'으로 설정하고 출력되는 펄스를 확인 }
var
Unit: Double;

begin
AxmStatusReadDriveDistance (0, @Unit);
end;
```

See Also

AxmStatusReadInMotion, AxmStatusReadDrivePulseCount, AxmStatusReadMotion, AxmStatusReadStop, AxmStatusReadMechanical, AxmStatusReadVel, AxmStatusReadPosError, AxmStatusSetActPos, AxmStatusGetActPos, AxmStatusSetCmdPos, AxmStatusGetCmdPos

AxmStatusSetActPos

Purpose

지정 축에 Actual 위치(Encoder값) 를 설정한다.

Format

C

DWORD AxmStatusSetActPos (long lAxisNo, double dPos);

Visual Basic

Function AxmStatusSetActPos (ByVal lAxisNo As Long, ByVal dPos As Double) As Long

Delphi

Function AxmStatusSetActPos (lAxisNo : LongInt; dPos : Double) : DWord; stdcall;

Input / Output

Name	in/out	Format	Init Value	Explanation
AxisNo	in	long	-	축번호(0 ~ (최대축수 - 1))
Pos	In	double	-	Act 위치

Return AXT_RT_SUCCESS(0000) : Successful execution of API

* See error code Table more information on status error codes

Description

사용자가 지정한 축에 강제로 Actual 위치를 설정할 수 있다. 주로 지금 위치를 0 위치로 만들 때 사용한다.

C Example

```
// 0축에 현재 위치를 엔코더 위치 '0'으로 설정
AxmStatusSetActPos (0, 0);

// 0축에 엔코더 위치 확인
double Pos;
AxmStatusGetActPos (0, &Pos);
```

VB Example

```
' 0축에 현재 위치를 엔코더 위치 '0'으로 설정
AxmStatusSetActPos 0, 0

' 0축에 엔코더 위치 확인
Dim Pos As Double
AxmStatusGetActPos 0, Pos
```

Delphi Example

```
{ 0축에 엔코더 위치 확인 }
var
Pos: Double;

Begin
{ 0축에 현재 위치를 엔코더 위치 '0'으로 설정 }
AxmStatusSetActPos (0, 0);

AxmStatusGetActPos (0, @Pos);
end;
```

See Also

AxmStatusReadInMotion, AxmStatusReadDrivePulseCount, AxmStatusReadMotion, AxmStatusReadStop, AxmStatusReadMechanical, AxmStatusReadVel, AxmStatusReadPosError, AxmStatusReadDriveDistance, AxmStatusGetActPos, AxmStatusSetCmdPos, AxmStatusGetCmdPos

AxmStatusGetActPos

Purpose

지정 축의 Actual 위치(Encoder값) 를 반환한다.

Format

C

```
DWORD AxmStatusGetActPos (long lAxisNo, double *dpPos);
```

Visual Basic

```
Function AxmStatusGetActPos (ByVal lAxisNo As Long, ByRef dpPos As Double) As Long
```

Delphi

```
function AxmStatusGetActPos (lAxisNo : LongInt; dpPos : PDouble) : DWord; stdcall;
```

Input / Output

Name	in/out	Format	Init Value	Explanation
AxisNo	in	long	-	축번호(0 ~ (최대축수 - 1))
Pos	out	double*	-	Act 위치

Return AXT_RT_SUCCESS(0000) : Successful execution of API

* See error code Table more information on status error codes

Description

사용자가 지정한 축의 엔코더로 읽어 들인 Actual 위치를 반환한다.

C Example

```
// 0축에 현재 위치를 엔코더 위치 '0'으로 설정
AxmStatusSetActPos (0, 0);

// 0축에 엔코더 위치 확인
double Pos;
AxmStatusGetActPos (0, &Pos);
```

VB Example

```
' 0축에 현재 위치를 엔코더 위치 '0'으로 설정
AxmStatusSetActPos 0, 0
```

```
' 0축에 엔코더 위치 확인
Dim Pos As Double
AxmStatusGetActPos 0, Pos
```

Delphi Example

```
{ 0축에 엔코더 위치 확인 }
var
Pos: Double;

Begin
{ 0축에 현재 위치를 엔코더 위치 '0'으로 설정 }
AxmStatusSetActPos (0, 0);

AxmStatusGetActPos (0, @Pos);
end;
```

See Also

AxmStatusReadInMotion, AxmStatusReadDrivePulseCount, AxmStatusReadMotion, AxmStatusReadStop, AxmStatusReadMechanical, AxmStatusReadVel, AxmStatusReadPosError, AxmStatusReadDriveDistance, AxmStatusSetActPos, AxmStatusSetCmdPos, AxmStatusGetCmdPos

AxmStatusSetCmdPos

Purpose

지정 축의 Command 위치를 설정한다.

Format

C

DWORD AxmStatusSetCmdPos (long lAxisNo, double dPos);

Visual Basic

Function AxmStatusSetCmdPos (ByVal lAxisNo As Long, ByVal dPos As Double) As Long

Delphi

function AxmStatusSetCmdPos (lAxisNo : LongInt; dPos : Double); DWord; stdcall;

Input / Output

Name	in/out	Format	Init Value	Explanation
AxisNo	in	long	-	축번호(0 ~ (최대축수 - 1))
Pos	In	double	-	Cmd 위치

Return AXT_RT_SUCCESS(0000) : Successful execution of API

* See error code Table more information on status error codes

Description

사용자가 지정한 축에 강제로 Command 위치를 설정할 수 있다. 주로 지금 위치를 0 위치로 만들 때 사용한다.

C Example

```
// 0축에 현재 위치를 명령 위치 '0'으로 설정
AxmStatusSetCmdPos (0, 0);

// 0축에 명령 위치 확인
double Pos;
AxmStatusGetCmdPos (0, &Pos);
```

VB Example

```
' 0축에 현재 위치를 명령 위치 '0'으로 설정
AxmStatusSetCmdPos 0, 0

' 0축에 명령 위치 확인
Dim Pos As Double
AxmStatusGetCmdPos 0, Pos
```

Delphi Example

```
{ 0축에 명령 위치 확인 }
var
Pos: Double;

begin
{ 0축에 현재 위치를 명령 위치 '0'으로 설정 }
AxmStatusSetCmdPos (0, 0);
AxmStatusGetCmdPos (0, @Pos);
end;
```

See Also

AxmStatusReadInMotion, AxmStatusReadDrivePulseCount, AxmStatusReadMotion, AxmStatusReadStop, AxmStatusReadMechanical, AxmStatusReadVel, AxmStatusReadPosError, AxmStatusReadDriveDistance, AxmStatusSetActPos, AxmStatusGetActPos, AxmStatusGetCmdPos

AxmStatusGetCmdPos

Purpose

지정 축의 Command 위치를 반환한다.

Format

C

DWORD AxmStatusGetCmdPos (long lAxisNo, double *dpPos);

Visual Basic

Function AxmStatusGetCmdPos (ByVal lAxisNo As Long, ByRef dpPos As Double) As Long

Delphi

function AxmStatusGetCmdPos (lAxisNo : LongInt; dpPos : PDouble) : DWord; stdcall;

Input / Output

Name	in/out	Format	Init Value	Explanation
AxisNo	in	long	-	축번호(0 ~ (최대축수 - 1))
Pos	out	double*	-	Cmd 위치

Return AXT_RT_SUCCESS(0000) : Successful execution of API

* See error code Table more information on status error codes

Description

사용자가 지정한 축의 내부 Command 위치를 반환한다.

C Example

```
// 0축에 현재 위치를 명령 위치 '0'으로 설정
AxmStatusSetCmdPos (0, 0);

// 0축에 명령 위치 확인
double Pos;
AxmStatusGetCmdPos (0, &Pos);
```

VB Example

```
‘ 0축에 현재 위치를 명령 위치 '0'으로 설정
AxmStatusSetCmdPos 0, 0
```

```
' 0축에 명령 위치 확인
Dim Pos As Double
AxmStatusGetCmdPos 0, Pos
```

Delphi Example

```
{ 0축에 명령 위치 확인 }
var
Pos: Double;

begin
{ 0축에 현재 위치를 명령 위치 '0'으로 설정 }
AxmStatusSetCmdPos (0, 0);
AxmStatusGetCmdPos (0, @Pos);
end;
```

See Also

AxmStatusReadInMotion, AxmStatusReadDrivePulseCount, AxmStatusReadMotion, AxmStatusReadStop, AxmStatusReadMechanical, AxmStatusReadVel, AxmStatusReadPosError, AxmStatusReadDriveDistance, AxmStatusSetActPos, AxmStatusGetActPos, AxmStatusSetCmdPos,

원점 검색 함수

Function	Description
AxmHomeSetSignalLevel	지정 축의 Home 센서 Level 을 설정한다.
AxmHomeGetSignalLevel	지정 축의 Home 센서 Level 을 반환한다.
AxmHomeReadSignal	지정 축의 Home 신호의 입력 상태를 반환한다.
AxmHomeSetMethod	해당 축의 원점검색을 수행하기 위해서는 반드시 원점 검색관련 파라메타들이 설정되어 있어야 된다. 만약 AxmMotLoadParaAll함수 설정 파일을 이용해 초기화가 정상적으로 수행됐다면 별도의 설정은 필요하지 않는다. 원점검색 방법 설정에는 검색 진행방향, 원점으로 사용할 신호, 원점 센서 Active Level, 엔코더 Z상 검출 여부 등을 설정 한다. (자세한 내용은 AxmMotLoadParaAll설명 부분 참조)
AxmHomeGetMethod	해당 축의 원점검색을 수행하기 위해서는 반드시 원점 검색관련 파라메타들이 설정되어 있어야 된다. 설정되어있는 홈 관련 파라메타들을 반환한다.
AxmHomeSetVel	원점을 빠르고 정밀하게 검색하기 위해 여러 단계의 스탭으로 검출한다. 이때 각 스탭에 사용 될 속도를 설정한다. 이 속도들의 설정값에 따라 원점검색 시간과, 원점검색 정밀도가 결정된다. 각 스탭별 속도들을 적절히 바꿔가면서 각 축의 원점검색 속도를 설정하면 된다. (자세한 내용은 AxmMotLoadParaAll설명 부분 참조)
AxmHomeGetVel	원점을 빠르고 정밀하게 검색하기 위해 여러 단계의 스탭으로 검출한다. 이때 각 스탭에 사용 될 속도를 설정한다. 이 속도들의 설정값에 따라 원점검색 시간과, 원점검색 정밀도가 결정된다. 설정된 각각 속도를 가져옵니다.

AxmHomeSetStart	원점검색 시작함수를 실행하면 라이브러리 내부에서 해당축의 원점검색을 수행 할 쓰레드가 자동 생성되어 원점검색을 순차적으로 수행한 후 자동 종료된다.
AxmHomeSetResult	원점검색 함수를 이용해 성공적으로 원점검색이 수행되고나면 검색 결과가 HOME_SUCCESS로 설정된다. 이 함수는 사용자가 원점검색을 수행하지않고 결과를 임의로 설정할 수 있다. 실제 원점검색 완료 후 정상 구동중 예외상황이 발생하면원점검색결과를 HOME_ERR_USER_BREAK 등으로 설정해 원점검색을 다시 수행하도록 할 수 있다.
AxmHomeGetResult	원점검색 함수의 검색 결과를 확인한다. 원점검색이 시작되면 HOME_SEARCHING으로 설정되며 원점검색에 실패하면 실패원인이 설정된다. 실패 원인을 제거한 후 다시 원점검색을 진행하면 된다.
AxmHomeGetRate	원점검색 시작되면 진행율을 확인할 수 있다. 원점검색이 완료되면 성공여부와 관계없이 100을 반환하게 된다. 원점검색 성공여부는 GetHome Result함수를 이용해 확인할 수 있다.

장비 구동을 하기 위해서는 원점검색을 수행해야 된다. 원점검색에 관련된 파라메타 중 중요한 것은 원점검색에 사용되는 속도로 dVelFirst 는 초기속도, dVelSecond 는 원점 센서가 1차 검출된 후 반대 방향으로 빠져 나오는 속도입니다. dVelThird 의 경우 2차 원점검출에 사용되는 속도입니다. dVelLast 의 경우 최종 원점 검출 시에 사용되는 속도로 너무 큰 값을 설정하면 원점 검색 완료 시 위치 에러가 발생 할 수 있다. 그렇다고 너무 작게 설정하면 원점 검색 시 시간이 많이 걸리게 되므로 적당한 값을 설정해야 된다.

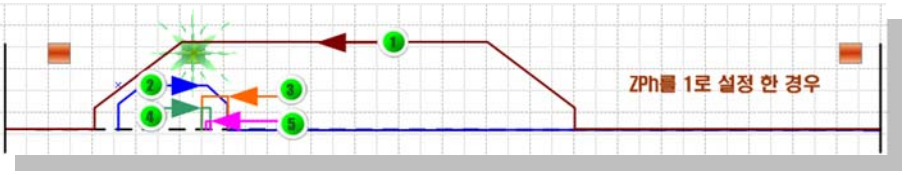
원점검색관련 설정변수	변수 설명
HmSig	원점센서로 사용할 신호를 설정 예) 원점센서(4), +Limit센서(0) 등
HmLev	원점센서의 Active Level을 설정 예) 0: A접점, 1: B접점
HmDir	원점센서 검출시 초기 검색진행 방향을 설정 예) 1: (+)방향, 0: (-)방향
Zphas	원점센서 검출 후 엔코더 Z상을 검출 여부 설정
VelFirst	원점검색 시 초기 고속검출 속도 (초기 원점센서가 감지되지 않았을 경우)
VelSecond	원점검색 시 1차 센서 검출 후 반대 방향으로 빠져나오는 속도

VelThird	원점검색 시 1차 센서 검출 후 재 검색 진행하는 속도
VelLast	원점검색 시 최종 검출 속도 설정 [원점검색의 정밀도 결정]
AccFirst	원점검색 시 초기 고속검출 가속도(가속도의 단위는 Unit/pulse를 1/1로 한경우에 PPS[Pulses/sec])
AccSecond	원점검색 시 1차 센서 검출 후 반대 방향으로 빠져나오는 가속도(가속도의 단위는 Unit/pulse를 1/1로 한경우에 PPS[Pulses/sec])
HClrTim	원점검색 완료 후 Cmd, Act(Encoder)위치를 Clear전 대기시간
HOffset	원점검색 완료 후 기구원점 재 설정 시 이동 Offset 값
NSWLimit	원점검색 완료 후 적용할 (-) Software Limit 값 설정
PSWLimit	원점검색 완료 후 적용할 (+) Software Limit 값 설정 (NSWLimit값이 PSWLimit값보다 같거나 클 경우 S/W Limit 설정되지 않음)

Zph 가 1 인 경우

4번 Step에서 Z상의 Down Edge를 검출하고 5번 Step에서 Z상의 Up Edge를 재 검출함으로 원점검색 속도와 정밀도를 높임

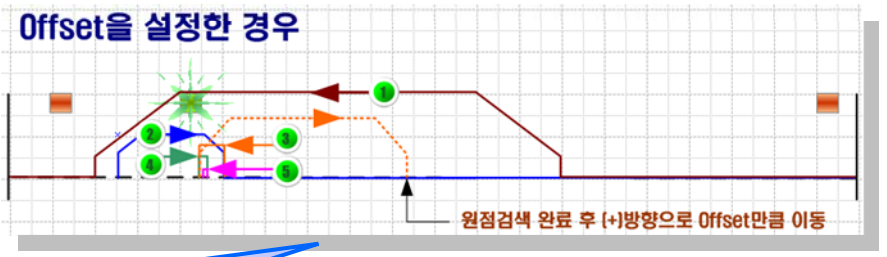
Step 별 원점검출 속도설정의 이해



1. VelFirst, AccFirst사용
   HmDir 방향으로 원점센서를 고속 검출하고 HAccF 감속도로 감속정지
2. VelSecond, AccSecond사용
   HmDir 반대 방향으로 원점센서의 Down Edge를 검출하고 HAccF 감속도로 감속정지
3. VelThird사용
   HmDir 방향으로 원점센서의 Up Edge를 검출하고 급정지
4. VelLast 사용
   HmDir 반대 방향으로 Z상의 Down Edge를 검출하고 급정지
5. VelLast 사용
   HmDir 방향으로 Z상의 Up Edge를 검출하고 급정지

기구적인 원점센서 검출이 완료된 후 특정 위치로 옮겨 원점을 설정해야 될 경우 HOffset에 값을 지정

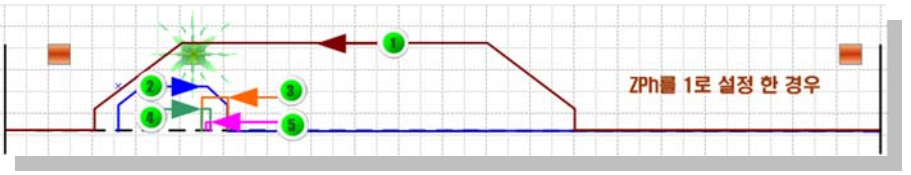
하면 원점 센서 검출이 완료된 후 자동으로 해당 Offset만큼 이동 완료 후 원점을 재 설정하게 된다. 이때 주의할 것은 (+)나 (-)Limit 을 벗어나지 않는 범위에서 Offset을 지정해야 된다. 그렇지 않을 경우 원점검색을 정상적으로 수행할 수 없습니다.



Offset 이동 완료 후 자동으로 0점으로 설정됩니다.

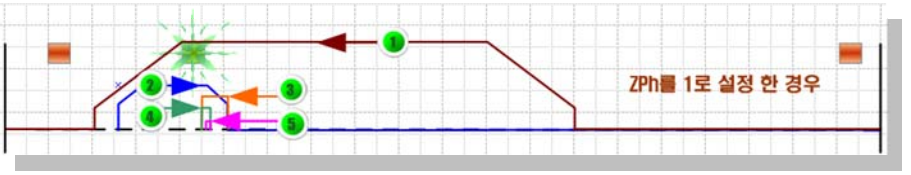
원점검색이 정상적으로 완료된 후 Soft Limit을 설정할 수 있다. Soft Limit은 기구적으로 Limit센서가 장착되어 있지 않거나 상황에 따라서 Limit을 변경해야 될 경우 사용하면 된다. 이때 한가지 주의하실 것은 NLimit(-)보다 PLimit(+)값이 커야 정상적으로 Soft Limit이 설정된다. 만약 NSwLimit값이 PSwLimit 값보다 같거나 클 경우 Soft Limit는 설정되지 않는다.

Zph 가 1 인 경우



4번 Step에서 Z상의 Down Edge를 검출하고 5번 Step에서 Z상의 Up Edge를 재 검출함으로 원점검색 속도와 정밀도를 높임

Step 별 원점검출 속도설정의 이해



VelFirst, AccFirst사용

1 HmDir 방향으로 원점센서를 고속 검출하고 HAccF 감속도로 감속정지

VelSecond, AccSecond사용

2 HmDir 반대 방향으로 원점센서의 Down Edge를 검출하고 HAccF 감속도로 감속정지

VelThird사용

3 HmDir 방향으로 원점센서의 Up Edge를 검출하고 급정지

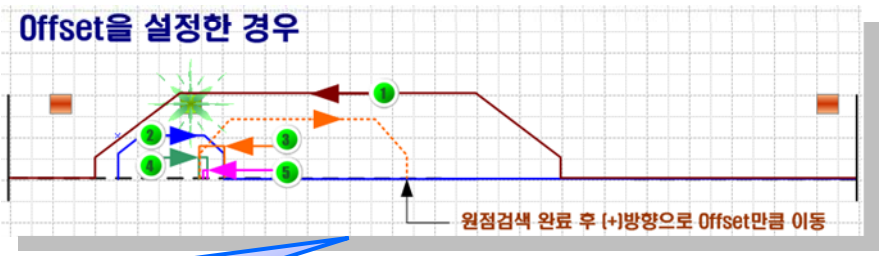
VelLast 사용

4 HmDir 반대 방향으로 Z상의 Down Edge를 검출하고 급정지

5 VelLast 사용

HmDir 방향으로 Z상의 Up Edge를 검출하고 급정지

기구적인 원점센서 검출이 완료된 후 특정 위치로 옮겨 원점을 설정해야 될 경우 HOffset에 값을 지정하면 원점 센서 검출이 완료된 후 자동으로 해당 Offset만큼 이동 완료 후 원점을 재 설정하게 된다. 이때 주의할 것은 (+)나 (-)Limit 을 벗어나지 않는 범위에서 Offset을 지정해야 된다. 그렇지 않을 경우 원점검색을 정상적으로 수행할 수 없습니다.



Offset 이동 완료 후 자동으로 0점으로 설정됩니다.

원점검색이 정상적으로 완료된 후 Soft Limit을 설정할 수 있다. Soft Limit은 기구적으로 Limit센서가 장착되어 있지 않거나 상황에 따라서 Limit을 변경해야 될 경우 사용하면 된다. 이때 한가지 주의하실 것은 NLimit(-)보다 PLimit(+)값이 커야 정상적으로 Soft Limit이 설정된다. 만약 NSwLimit값이 PSwLimit 값보다 같거나 클 경우 Soft Limit는 설정되지 않는다.

AxmHomeSetSignalLevel

Purpose

지정 축의 Home 입력 신호의 출력 레벨을 설정한다.

Format

C

```
DWORD AxmHomeSetSignalLevel (long lAxisNo, DWORD uLevel);
```

Visual Basic

```
Function AxmHomeSetSignalLevel (ByVal lAxisNo As Long, ByVal uLevel As Long) As Long
```

Delphi

```
function AxmHomeSetSignalLevel (lAxisNo : LongInt; uLevel : DWord) : DWord; stdcall;
```

Input / Output

Name	in/out	Format	Init Value	Explanation
AxisNo	in	long	–	축번호(0 ~ (최대축수 – 1))
Level	In	DWORD	1	Home 신호의 Active Level

Level

#define	Value	Explanation
LOW	00h	B 접점(Normal Close)
HIGH	01h	A 접점(Normal Open)

Return AXT_RT_SUCCESS(0000) : Successful execution of API

* See error code Table more information on status error codes

Description

사용자가 지정한 축의 Home 입력 레벨을 설정한다. 현재의 Home 신호 Level을 읽기 위해서는 AxmHomeReadSignal 함수를 사용하여 확인할 수 있다.

주의사항 : 레벨을 잘못 설정시 현재의 홈 상태에 따라 –방향으로 설정해도 +방향으로 동작할수 있으며, 홈을 찾는데 있어 문제가 될 수 있다.

C Example

```
// 0 축에 홈입력 신호 레벨를 설정한다.
AxmHomeSetSignalLevel(0, HIGH);

// 0축의 Home 입력신호 레벨 값을 확인
DWORD  state;

AxmHomeGetSignalLevel (0, &state);
```

VB Example

```
‘ 0 축에 홈입력 신호 레벨를 설정한다.
AxmHomeSetSignalLevel 0, HIGH

‘0축의 Home 입력신호 레벨 값을 확인
Dim state As Long

AxmHomeGetSignalLevel 0, state
```

Delphi Example

```
{ 0축의 Home 레벨 값을 확인 }
var
state: DWord;

Begin
{ 0 축에 홈 입력신호 레벨를 설정한다. }
AxmHomeSetSignalLevel (0, HIGH);
AxmHomeGetSignalLevel (0, @state);
end;
```

See Also

AxmHomeGetMethod, AxmHomeSetVel, AxmHomeGetVel, AxmHomeSetStart, AxmHomeSetResult, AxmHomeGetResult, AxmHomeGetRate

AxmHomeGetSignalLevel

Purpose

지정 축의 Home 신호의 출력 레벨 설정을 반환한다.

Format

C

```
DWORD AxmHomeGetSignalLevel (long lAxisNo, DWORD *upLevel);
```

Visual Basic

```
Function AxmHomeGetSignalLevel (ByVal lAxisNo As Long, ByRef upLevel As Long) As Long
```

Delphi

```
function AxmHomeGetSignalLevel (lAxisNo : LongInt; upLevel : PDWord) : DWord; stdcall;
```

Input / Output

Name	in/out	Format	Init Value	Explanation
AxisNo	in	long	–	축번호(0 ~ (최대축수 – 1))
Level	out	DWORD*	1	Home 신호의 Active Level

Level

#define	Value	Explanation
LOW	00h	B 접점(Normal Close)
HIGH	01h	A 접점(Normal Open)

Return AXT_RT_SUCCESS(0000) : Successful execution of API

* See error code Table more information on status error codes

Description.

'AxmHomeSetSignalLevel '로 설정한 Home 입력 레벨을 반환한다.

C Example

```
// 0 축에 홈입력 신호 레벨를 설정한다.
AxmHomeSetSignalLevel (0, HIGH);

// 0축의 Home 입력신호 레벨 값을 확인
DWORD  state;
```

```
AxmHomeGetSignalLevel (0, &state);
```

VB Example

```
' 0 축에 홈입력 신호 레벨를 설정한다.
AxmHomeSetSignalLevel 0, HIGH

'0축의 Home 입력신호 레벨 값을 확인
Dim state As Long

AxmHomeGetSignalLevel 0, state
```

Delphi Example

```
{ 0축의 Home 레벨 값을 확인 }
var
state: DWord;

Begin
{ 0 축에 홈 입력신호 레벨를 설정한다. }
AxmHomeSetSignalLevel (0, HIGH);
AxmHomeGetSignalLevel (0, @state);
end;
```

See Also

AxmHomeGetMethod, AxmHomeSetVel, AxmHomeGetVel, AxmHomeSetStart, AxmHomeSetResult, AxmHomeGetResult, AxmHomeGetRate

AxmHomeReadSignal

Purpose

지정 축의 Home 신호의 입력 상태를 반환한다.

Format

C

DWORD AxmHomeReadSignal (long lAxisNo, DWORD *upStatus);

Visual Basic

Function AxmHomeReadSignal (ByVal lAxisNo As Long, ByRef upStatus As Long) As Long

Delphi

function AxmHomeReadSignal (lAxisNo : LongInt; upStatus : PDword) : Dword; stdcall;

Input / Output

Name	in/out	Format	Init Value	Explanation
AxisNo	in	long	-	축번호(0 ~ (최대축수 - 1))
Status	out	DWORD*	-	신호의 입력 상태

Status

#define	Value	Explanation
INACTIVE	00h	비활성화
ACTIVE	01h	활성화

Return AXT_RT_SUCCESS(0000) : Successful execution of API

* See error code Table more information on status error codes

Description

현재 Home 신호의 입력상태 를 반환한다.

C Example

```
// 0축에 Home 신호의 입력 확인
DWORD upStatus;
AxmHomeReadSignal (0, &upStatus);
```

VB Example

```
 ' 0축에 Inpositon 신호의 입력 레벨을 설정
Dim upStatus As Long
AxmSignalReadInpos 0, upStatus
```

Delphi Example

```
{ 0축에 Home 신호의 입력 확인 }
var
upStatus: DWord;

begin
AxmHomeReadSignal (0, @upStatus);
end;
```

See Also

AxmHomeGetMethod, AxmHomeSetVel, AxmHomeGetVel, AxmHomeSetStart, AxmHomeSetResult, AxmHomeGetResult, AxmHomeGetRate

AxmHomeSetMethod

Purpose

해당 축의 원점검색을 수행하기 위해서는 반드시 원점 검색관련 파라메타들이 설정되어 있어야 된다. 만약 AxmMotLoadParaAll함수를 이용해 초기화가 정상적으로 수행됐다면 별도의 설정은 필요하지 않는다. 원점검색 방법 설정에는 검색 진행방향, 원점으로 사용할 신호, 원점센서 Active Level, 엔코더 Z상 검출 여부 등을 설정한다.(자세한 내용은 AxmMotLoadParaAll설명 부분 참조)

Format

C

```
DWORD AxmHomeSetMethod (long lAxisNo, long lHmDir, DWORD uHomeSignal, DWORD uZphas,
double dHomeClrTime, double dHomeOffset);
```

Visual Basic

```
Function AxmHomeSetMethod (ByVal lAxisNo As Long, ByVal nHmDir As Long, ByVal
uHomeSignal As Long, ByVal uZphas As Long, ByVal dHomeClrTime As Double, ByVal
dHomeOffset As Double) As Long
```

Delphi

```
function AxmHomeSetMethod (lAxisNo : LongInt; nHmDir : LongInt; uHomeSignal : DWord;
uZphas : DWord; dHomeClrTime : Double; dHomeOffset : Double) : DWord; stdcall;
```

Input / Output

Name	in/out	Format	Init Value	Explanation
AxisNo	in	long	-	AxmMotLoadParaAll 파일을 이용해 초기값이 설정됐는지 확인 할 채널(축) 번호 축번호(0 ~ (최대축수 - 1))
HmDir	In	long	0	원점 검출 시 ㅣ초기 검색 진행 설정
uHomeSignal	In	DWORD	4	원점 검색에 사용 할 신호설정
uZphas	In	DWORD	0	1차 원점검색 완료 후 엔코더 Z상 검출 유무 설정
dHomeClrTime	In	double	1000	원점검색 완료 후 지령위치와 Encoder위치를 Clear하기 위해 대기하는 시간설정 [mSec단위]
dpHomeOffset	In	double	0	원점검색이 완료된 후 기구 원점으로 이동

				후 원점 재설정 할 위치

lHmDir

#define	Value	Explanation
DIR_CCW	00h	반시계방향
DIR_CW	01h	시계방향

uHomeSignal

#define	Value	Explanation
PosEndLimit	00h	+Elm(End limit) +방향 리미트 센서 신호
NegEndLimit	01h	-Elm(End limit) -방향 리미트 센서 신호
HomeSensor	04h	IN0(ORG) 원점 센서 신호

uZphas

#define	Value	Explanation
DISABLE	00h	Z상 검출 사용하지 않음
+방향	01h	Z상 검출 +방향 사용
-방향	02h	Z상 검출 -방향 사용

Return AXT_RT_SUCCESS(0000) : Successful execution of API
* See error code Table more information on status error codes

Description

해당 축의 원점검색을 수행하기 위해서는 반드시 원점 검색관련 파라메타들이 설정되 있어야 된다. 만약 AxmMotLoadParaAll 설정 파일을 이용해 초기화가 정상적으로 수행됐다면 별도의 설정은 필요하지 않다. 원점검색 방법 설정에는 검색 진행방향, 원점으로 사용할 신호,엔코더 Z상 검출 여부등을 설정 한다. 원점센서 Active Level은 AxmHomeSetSignalLevel함수를 사용하여 별도로 설정한다.(자세한 내용은 AxmMotLoadParaAll설명 부분 참조)

주의사항 : 레벨을 잘못 설정시 -방향으로 설정해도 +방향으로 동작할수 있으며, 홈을 찾는데 있어 문제가 될수있다.

C Example

```
long   lHmDir = 0;
DWORD  lHmsig =  4;
DWORD  dwZphas = 0;
double  dwHClrTim = 2000.0, dwHOffset = 0.0;
```

```
AxmHomeSetMethod(0, lHmDir, lHmsig, dwZphas, dwHClrTim, dwHOffset);
AxmHomeSetVel(0,10000, 5000, 500, 50, 40000, 10000);
AxmHomeSetStart(0);

// (-)방향으로 원점검색, Home Sensor이용, Home Sensor Type A접점, Z상 검출 사용안
       함, Offset이동안함으로 설정

AxmHomeGetMethod (0, &lHmDir, &lHmsig, &dwZphas, &dwHClrTim, &
     dwHOffset);
```

VB Example

```
Dim dwHOffset As Double
Dim dwHClrTim As Double
Dim lHmDir, lHmsig, dwZphas As Long

dwHClrTim = 2000#
dwHOffset = 0#

lHmDir = 0
lHmsig = 4
dwZphas = 0
'(-)방향으로 원점검색, Home Sensor이용, Home Sensor Type A접점, Z상 검출 사용안
       함, Offset이동안함으로 설정

AxmHomeSetMethod 0, lHmDir, lHmsig, dwZphas, dwHClrTim, dwHOffset
AxmHomeSetVel 0, 10000, 5000, 500, 50, 40000, 10000
AxmHomeSetStart 0

AxmHomeGetMethod 0, lHmDir, lHmsig, dwZphas, dwHClrTim, dwHOffset
```

Delphi Example

```
var
   dwZphas, lHmsig : DWORD;
   dwHOffset, dwHClrTim : Double;
   lHmDir : LongInt;
   velocity : Double;

begin

lHmDir := 0;
lHmsig :=  4;
dwZphas := 0;
dwHClrTim := 2000.0;
dwHOffset := 0.0;

AxmHomeSetMethod(0, lHmDir, lHmsig, dwZphas, dwHClrTim,dwHOffset);
AxmHomeSetVel(0,10000, 5000, 500, 50, 40000, 10000);
AxmHomeSetStart(0);
{ (-)방향으로 원점검색, Home Sensor이용, Home Sensor Type A접점, Z상 검출 사용안
       함, Offset이동안함으로 설정 }
```

```
AxmHomeGetMethod (0, @lHmDir, @lHmsig, @dwZphas, @dwHClrTim, @dwHOffset);
End;
```

See Also

AxmHomeGetMethod, AxmHomeSetVel, AxmHomeGetVel, AxmHomeSetStart, AxmHomeSetResult, AxmHomeGetResult, AxmHomeGetRate

AxmHomeGetMethod

Purpose

해당 축의 원점검색을 수행하기 위해서는 반드시 원점 검색관련 파라메타들이 설정되 있어야 된다. 만약 AxmMotLoadParaAll설정 파일을 이용해 초기화가 정상적으로 수행됐다면 별도의 설정은 필요하지 않다. 원점검색 방법 설정에는 검색 진행방향, 원점으로 사용할 신호, 엔코더 Z상 검출 여부등을 설정을 읽어온다..(자세한 내용은 AxmMotLoadParaAll설명 부분 참조)

Format

C

```
DWORD AxmHomeGetMethod (long lAxisNo, long *lpHmDir, DWORD *upHomeSignal, DWORD *upZphas, double *dpHomeClrTime, double *dpHomeOffset);
```

Visual Basic

```
Function AxmHomeGetMethod (ByVal lAxisNo As Long, ByRef lpHmDir As Long, ByRef upHomeSignal As Long, ByRef upZphas As Long, ByRef dpHomeClrTime As Double, ByRef dpHomeOffset As Double) As Long
```

Delphi

```
function AxmHomeGetMethod (lAxisNo : LongInt; lpHmDir : PLongInt; upHomeSignal : PDWord; upZphas : PDWord; dpHomeClrTime : PDouble; dpHomeOffset : PDouble) : DWord; stdcall;
```

Input / Output

Name	in/out	Format	Init Value	Explanation
AxisNo	in	long	–	AxmMotLoadParaAll 파일을 이용해 초기값이 설정됐는지 확인 할 채널(축) 번호(0부터 시작함)
HmDir	out	long*	0	원점검색을 초기에 진행 할 방향설정
uHomeSignal	out	DWORD*	4	원점 검색에 사용 할 신호설정
uZphas	out	DWORD*	0	1차 원점검색 완료 후 엔코더 Z상 검출 유무 설정
dHomeClrTime	out	double*	1000	원점검색 완료 후 지령위치와 Encoder위치를 Clear하기 위해 대기하는 시간설정 [mSec단위]
dpHomeOffs	out	double*	0	원점검색이 완료된 후 기구 원점으로 이동 후 원

et				점 재설정 할 위치

lHmDir

#define	Value	Explanation
DIR_CCW	00h	반시계방향
DIR_CW	01h	시계방향

uHomeSignal

#define	Value	Explanation
PosEndLimit	00h	+Elm(End limit) +방향 리미트 센서 신호
NegEndLimit	01h	-Elm(End limit) -방향 리미트 센서 신호
HomeSensor	04h	IN0(ORG) 원점 센서 신호

uZphas

#define	Value	Explanation
DISABLE	00h	Z상 검출 사용하지 않음
+방향	01h	Z상 검출 +방향 사용
-방향	02h	Z상 검출 -방향 사용

Return AXT_RT_SUCCESS(0000) : Successful execution of API
* See error code Table more information on status error codes

Description

해당 축의 원점검색을 수행하기 위해서는 반드시 원점 검색관련 파라메타들이 설정되 있어야 된다. 만약 AxmMotLoadParaAll설정 파일을 이용해 초기화가 정상적으로 수행됐다면 별도의 설정은 필요하지 않다. 원점검색 방법 설정에는 검색 진행방향, 원점으로 사용할 신호 엔코더 Z상 검출 여부등을 설정 한다. 원점센서 Active Level은 AxmtHomeIGeSignaLevel함수를 사용하여 별도로 읽어온다. (자세한 내용은 AxmMotLoadParaAll설명 부분 참조)

C Example

```
long    lHmDir = 0;
DWORD   lHmsig =  4;
DWORD   dwZphas = 0;
double  dwHClrTim = 2000.0, dwHOffset = 0.0;

AxmHomeSetMethod(0, lHmDir, lHmsig, dwZphas, dwHClrTim, dwHOffset);
AxmHomeSetVel(0,10000, 5000, 500, 50, 40000, 10000);
```

```
AxmHomeSetStart(0);

// (-)방향으로 원점검색, Home Sensor이용, Home Sensor Type A접점, Z상 검출 사용안함,
       Offset이동안함으로 설정

AxmHomeGetMethod (0, &lHmDir, &lHmsig, &dwZphas, &dwHClrTim, & dwHOffset);
```

VB Example

```
Dim dwHOffset As Double
Dim dwHClrTim As Double
Dim lHmDir, lHmsig, dwZphas As Long

dwHClrTim = 2000#
dwHOffset = 0#

lHmDir = 0
lHmsig = 4
dwZphas = 0
'(-)방향으로 원점검색, Home Sensor이용, Home Sensor Type A접점, Z상 검출 사용안함,
       Offset이동안함으로 설정

AxmHomeSetMethod 0, lHmDir, lHmsig, dwZphas, dwHClrTim, dwHOffset
AxmHomeSetVel 0, 10000, 5000, 500, 50, 40000, 10000
AxmHomeSetStart 0

AxmHomeGetMethod 0, lHmDir, lHmsig, dwZphas, dwHClrTim, dwHOffset
```

Delphi Example

```
var
   dwZphas, lHmsig : DWORD;
   dwHOffset, dwHClrTim : Double;
   lHmDir : LongInt;
   velocity : Double;

begin

lHmDir := 0;
lHmsig :=  4;
dwZphas := 0;
dwHClrTim := 2000.0;
dwHOffset := 0.0;

AxmHomeSetMethod(0, lHmDir, lHmsig, dwZphas, dwHClrTim,dwHOffset);
AxmHomeSetVel(0,10000, 5000, 500, 50, 40000, 10000);
AxmHomeSetStart(0);
{ (-)방향으로 원점검색, Home Sensor이용, Home Sensor Type A접점, Z상 검출 사용안함,
       Offset이동안함으로 설정 }
AxmHomeGetMethod (0, @lHmDir, @lHmsig, @dwZphas, @dwHClrTim, @dwHOffset);
End;
```

See Also

AxmHomeSetMethod, AxmHomeSetVel, AxmHomeGetVel, AxmHomeSetStart, AxmHomeSetResult, AxmHomeGetResult, AxmHomeGetRate

AxmHomeSetVel

Purpose

원점을 빠르고 정밀하게 검색하기 위해 여러 단계의 스텝으로 원점을 검색한다. 이때 각 스텝에 사용될 속도를 설정한다. 이 속도들의 설정값에 따라 원점검색 시간과, 원점검색 정밀도가 결정된다. 각 스텝 별 속도들을 적절히 바꿔가면서 각 축의 원점검색 속도를 설정하면 된다. (자세한 내용은 AxmMotLoadParaAll함수 설명 부분 참조)

Format

C

DWORD AxmHomeSetVel (long lAxisNo, double dVelFirst, double dVelSecond, double dVelThird, double dvelLast, double dAccFirst, double dAccSecond)

Visual Basic

Function AxmHomeSetVel (ByVal lAxisNo As Long, ByVal dVelFirst As Double, ByVal dVelSecond As Double, ByVal dVelThird As Double, ByVal dvelLast As Double, ByVal dAccFirst As Double, ByVal dAccSecond As Double) As Long

Delphi

function AxmHomeSetVel (lAxisNo : LongInt; dVelFirst : Double; dVelSecond : Double; dVelThird : Double; dvelLast : Double; dAccFirst : Double; dAccSecond : Double) : DWord; stdcall;

Input / Output

Name	in/out	Format	Init Value	Explanation
AxisNo	in	long	-	축번호(0 ~ (최대축수 - 1))
dVelFirst	In	double	10000	원점센서가 감지되어 있지않을 때 원점검색 진행방향으로 구동할 속도
dVelSecond	In	double	10000	원점센서가 감지되어 있을 때 원점검색 진행 반대 방향으로 구동 할 속도
dVelThird	In	double	2000	1차 원점센서 검출 후 재 검색할 때 사용될 구동속도
dvelLast	In	double	100	마지막 원점 검출시 사용될 구동속도 (Z상 검출 사용시 속도, 원점센서 정밀 검출 속도)
dAccFirst	In	double	40000	원점센서가 감지되어 있지않을 때 원점검색

				진행 방향으로 구동할 때 사용될 가속/감속도(감속도의 단위는 Unit/pulse를 1/1로 한 경우에 PPS[Pulses/sec])
dAccSecond	In	double	40000	원점센서가 감지되어 있을 때 원점검색 진행 반대 방향으로 구동할 때 사용될 가속/감속도(감속도의 단위는 Unit/pulse를 1/1로 한경우에 PPS[Pulses/sec])

Return AXT_RT_SUCCESS(0000) : Successful execution of API
* See error code Table more information on status error codes

Description

원점을 빠르고 정밀하게 검색하기 위해 여러 단계의 스텝으로 검출한다. 이때 각 스텝에 사용 될 속도를 설정한다. 이 속도들의 설정값에 따라 원점검색 시간과, 원점검색 정밀도가 결정된다. 각 스텝 별 속도들을 적절히 바꿔가면서 각 축의 원점검색 속도를 설정하면 된다. (자세한 내용은 AxmMotLoadParaAll함수 설명 부분 참조)

C Example

```
long    lHmDir = 0;
DWORD   lHmsig =  4;
DWORD   dwZphas = 0;
double  dwHClrTim = 2000.0, dwHOffset = 0.0;

AxmHomeSetMethod(0, lHmDir, lHmsig, dwZphas, dwHClrTim, dwHOffset);
AxmHomeSetVel(0,10000, 5000, 500, 50, 40000, 10000);
AxmHomeSetStart(0);

// (-)방향으로 원점검색, Home Sensor이용, Home Sensor Type A접점, Z상 검출 사용안
       함, Offset이동안함으로 설정

AxmHomeGetMethod (0, &lHmDir, &lHmsig, &dwZphas, &dwHClrTim, &
       dwHOffset);
```

VB Example

```
Dim dwHOffset As Double
Dim dwHClrTim As Double
Dim lHmDir, lHmsig, dwZphas As Long

dwHClrTim = 2000#
dwHOffset = 0#
```

```
lHmDir = 0
lHmsig = 4
dwZphas = 0
'(-)방향으로 원점검색, Home Sensor이용, Home Sensor Type A접점, Z상 검출 사용안
       함, Offset이동안함으로 설정

AxmHomeSetMethod 0, lHmDir, lHmsig, dwZphas, dwHClrTim, dwHOffset
AxmHomeSetVel 0, 10000, 5000, 500, 50, 40000, 10000
AxmHomeSetStart 0

AxmHomeGetMethod 0, lHmDir, lHmsig, dwZphas, dwHClrTim, dwHOffset
```

Delphi Example

```
var
   dwZphas, lHmsig : DWORD;
   dwHOffset, dwHClrTim : Double;
   lHmDir : LongInt;
   velocity : Double;

begin

lHmDir := 0;
lHmsig :=  4;
dwZphas := 0;
dwHClrTim := 2000.0;
dwHOffset := 0.0;

AxmHomeSetMethod(0, lHmDir, lHmsig, dwZphas, dwHClrTim,dwHOffset);
AxmHomeSetVel(0,10000, 5000, 500, 50, 40000, 10000);
AxmHomeSetStart(0);
{ (-)방향으로 원점검색, Home Sensor이용, Home Sensor Type A접점, Z상 검출 사용안
       함, Offset이동안함으로 설정 }
AxmHomeGetMethod (0, @lHmDir, @lHmsig, @dwZphas, @dwHClrTim, @dwHOffset);
End;
```

See Also

AxmHomeSetMethod, AxmHomeGetMethod, AxmHomeGetVel, AxmHomeSetStart, AxmHomeSetResult, AxmHomeGetResult, AxmHomeGetRate

AxmHomeGetVel

Purpose

원점을 빠르고 정밀하게 검색하기 위해 여러 단계의 스텝으로 검출한다.

Format

C

```
DWORD AxmHomeGetVel (long lAxisNo, double *dpVelFirst, double *dpVelSecond, double
*dpVelThird, double *dpVelLast, double *dpAccFirst, double *dpAccSecond);
```

Visual Basic

```
Function AxmHomeGetVel (ByVal lAxisNo As Long, ByRef dpVelFirst As Double, ByRef
dpVelSecond As Double, ByRef dpVelThird As Double, ByRef dpVelLast As Double, ByRef
dpAccFirst As Double, ByRef dpAccSecond As Double) As Long
```

Delphi

```
function AxmHomeGetVel (lAxisNo : LongInt; dpVelFirst : PDouble; dpVelSecond : PDouble;
dpVelThird : PDouble; dpVelLast : PDouble; dpAccFirst : PDouble; dpAccSecond : PDouble) :
DWord; stdcall;
```

Input / Output

Name	in/out	Format	Init Value	Explanation
AxisNo	in	long	-	축번호(0 ~ (최대축수 - 1))
dVelFirst	out	double*	10000	원점센서가 감지되어 있지않을 때 원점검색 진행방향으로 구동할 속도
dVelSecond	out	double*	10000	원점센서가 감지되어 있을 때 원점검색 진행 반대 방향으로 구동 할 속도
dVelThird	out	double*	2000	1차 원점센서 검출 후 재 검색할 때 사용될 구동속도
dvelLast	out	double*	100	마지막 원점 검출시 사용될 구동속도 (Z상 검출 사용시 속도, 원점센서 정밀 검출 속도)
dAccFirst	out	double*	40000	원점센서가 감지되어 있지않을 때 원점검색 진행 방향으로 구동할 때 사용될 가속/감

				속도(감속도의 단위는 Unit/pulse를 1/1로 한경우에 PPS[Pulses/sec])
dAccSecond	out	double*	40000	원점센서가 감지되어 있을 때 원점검색 진행 반대 방향으로 구동할 때 사용될 가속/감속도(감속도의 단위는 Unit/pulse를 1/1로 한경우에 PPS[Pulses/sec])

Return AXT_RT_SUCCESS(0000) : Successful execution of API
* See error code Table more information on status error codes

Description

참고 : 필요한 데이터만 가지고 올 경우 필요하지 않는 변수에 NULL값을 넣어 사용 할 수 있다.
원점을 빠르고 정밀하게 검색하기 위해 여러 단계의 스텝으로 검출한다. 이때 각 스텝에 사용 될 속도를 설정한다. 이 속도들의 설정값에 따라 원점검색 시간과, 원점검색 정밀도가 결정된다. 각 스텝 별 속도들을 적절히 바꿔가면서 각 축의 원점검색 속도를 설정하면 된다. (자세한 내용은 AxmMotLoadParaAll함수 설명 부분 참조)

C Example

```
long    lHmDir = 0;
DWORD   lHmsig =  4;
DWORD   dwZphas = 0;
double  dwHClrTim = 2000.0, dwHOffset = 0.0;

AxmHomeSetMethod(0, lHmDir, lHmsig, dwZphas, dwHClrTim, dwHOffset);
AxmHomeSetVel(0,10000, 5000, 500, 50, 40000, 10000);
AxmHomeSetStart(0);

// (-)방향으로 원점검색, Home Sensor이용, Home Sensor Type A접점, Z상 검출 사용안
       함, Offset이동안함으로 설정

AxmHomeGetMethod (0, &lHmDir, &lHmsig, &dwZphas, &dwHClrTim, &
       dwHOffset);
AxmHomeGetVel(0, & dVelFirst, & dVelSecond, & dVelThird, & dVelLast,
              &dAccFirst, & dAccSecond);
```

VB Example

```
Dim dpSlOffset As Double
Dim dSlORange As Double
Dim bSlHomeUse, bGatryOn, bMaSlChange As Long
Dim dVelFirst, dVelSecond, dVelThird,dVelLast, dAccFirst,dAccSecond As
       Double
```

```
Dim dwHOffset As Double
Dim dwHClrTim As Double
Dim lHmDir, lHmsig, dwZphas As Long

dwHClrTim = 2000#
dwHOffset = 0#

lHmDir = 0
lHmsig = 4
dwZphas = 0
'(-)방향으로 원점검색, Home Sensor이용, Home Sensor Type A접점, Z상 검출 사용안
       함, Offset이동안함으로 설정

AxmHomeSetMethod 0, lHmDir, lHmsig, dwZphas, dwHClrTim, dwHOffset
AxmHomeSetVel 0, 10000, 5000, 500, 50, 40000, 10000
AxmHomeSetStart 0

AxmHomeGetMethod 0, lHmDir, lHmsig, dwZphas, dwHClrTim, dwHOffset
AxmHomeGetVel 0, dVelFirst, dVelSecond, dVelThird, dVelLast, dAccFirst,
       dAccSecond
```

Delphi Example

```
var
   dwZphas, bSlHomeUse, bGatryOn, bMaSlChange : DWORD;
   dwHOffset, dwHClrTim, dpSlOffset,dSlORange : Double;
   dVelFirst, dVelSecond, dVelThird,dVelLast, dAccFirst,dAccSecond :
      Double;
   lHmDir, lHmsig : LongInt;
   velocity : Double;

begin

lHmDir := 0;
lHmsig :=  4;
dwZphas := 0;
dwHClrTim := 2000.0;
dwHOffset := 0.0;

AxmHomeSetMethod(0, lHmDir, lHmsig, dwZphas, dwHClrTim,dwHOffset);
AxmHomeSetVel(0,10000, 5000, 500, 50, 40000, 10000);
AxmHomeSetStart(0);
{ (-)방향으로 원점검색, Home Sensor이용, Home Sensor Type A접점, Z상 검출 사용안
       함, Offset이동안함으로 설정 }
AxmHomeGetMethod (0, @lHmDir, @lHmsig, @dwZphas, @dwHClrTim, @dwHOffset);
AxmHomeGetVel (0, @dVelFirst, @dVelSecond, @dVelThird, @dVelLast,
              @dAccFirst, @dAccSecond);
End;
```

See Also

AxmHomeSetMethod, AxmHomeGetMethod, AxmHomeSetVel, AxmHomeSetStart, AxmHomeSetResult, AxmHomeGetResult, AxmHomeGetRate

AxmHomeSetStart

Purpose

원점검색 시작 함수를 사용 하면 라이브러리 내부에서 해당 축의 원점검색을 순차적으로 수행 한 후 자동 종료 된다.

Format

C

DWORD AxmHomeSetStart (long lAxisNo)

Visual Basic

Function AxmHomeSetStart (ByVal lAxisNo As Long) As Long

Delphi

function AxmHomeSetStart (lAxisNo : LongInt) : DWord; stdcall;

Input / Output

Name	in/out	Format	Init Value	Explanation
AxisNo	in	long	-	축번호(0 ~ (최대축수 - 1))

Return AXT_RT_SUCCESS(0000) : Successful execution of API

* See error code Table more information on status error codes

Description

원점검색 시작 함수를 사용 하면 라이브러리 내부에서 해당 축의 원점검색을 순차적으로 수행 한 후 자동 종료 된다.

주의사항: 진행방향과 반대방향의 리미트 센서가 들어와도 진행방향의 센서가 ACTIVE되지않으면 동작한다. 원점 검색이 시작되어 진행방향이 리밋트 센서가 들어오면 리밋트 센서가 감지되었다고 생각하고 다음단계로 진행된다.

C Example

```
DWORD uHomeResult;
if(AxmHomeSetStart (0) == AXT_RT_SUCCESS)  // 원점검색 시작함수 실행
 printf("0축 원점검색을 시작한다.");
else{
AxmHomeGetResult (0, & uHomeResult);
 printf("0축 원점검색을 시작할 수 없습니다. [에러코드:%d]", uHomeResult);
}
```

VB Example

```
Dim nCount As Integer
Dim strData As String
Dim uHomeResult As Long
If (AxmHomeSetStart (0) = AXT_RT_SUCCESS )Then    ‘ 원점검색 시작함수 실행
 MsgBox “0축 원점검색을 시작한다..”, vbOKCancel
Else
AxmHomeGetResult 0, uHomeResult
 nCount = uHomeResult
 strData  = “0축 원점검색을 시작할 수 없습니다. 에러코드” + CStr(nCount) + ” 입니
       다.”
MsgBox strData
End If
```

Delphi Example

```
var
nCount : Word;  strData : String;
uHomeResult: Dword

begin
if (AxmHomeSetStart (0) = AXT_RT_SUCCESS ) then    { 원점검색 시작함수 실행 }
 Application.MessageBox(‘0축 원점검색을 시작한다..’, ‘Ajinextek’, MB_OK)
else
 begin
AxmHomeGetResult(0, @ uHomeResult);
nCount    := uHomeResult;
strData     :=  ‘0축 원점검색을 시작할 수 없습니다. 에러코드 ’ + IntToStr(nCount)
       + ‘입니다.’;
Application.MessageBox (PCHAR(strData), ‘Ajinextek’, MB_OK);
end;
End;
```

See Also

AxmHomeSetMethod, AxmHomeGetMethod, AxmHomeSetVel, AxmHomeGetVel, AxmHomeSetResult, AxmHomeGetResult, AxmHomeGetRate

AxmHomeSetResult

Purpose

원점검색 함수를 이용해 성공적으로 원점검색이 수행되고 나면 검색 결과를 출력한다.
원점검색 함수를 이용해 성공적으로 원점검색이 수행되고 나면 검색 결과가 HOME_SUCCESS로 설정된다. 이 함수는 사용자가 원점검색을 수행하지 않고 결과를 임의로 설정할 수 있다.
실제 원점검색 완료후 예외상황이 발생하면 원점검색을 다시 수행한다. 원점검색이 정상적으로 완료되었을때 그결과를 HOME_SUCCESS로 설정하지 않고 ERROR로 설정하여 원점검색을 다시 수행하도록 한다.

Format

C

```
DWORD AxmHomeSetResult (long lAxisNo, DWORD uHomeResult)
```

Visual Basic

```
Function AxmHomeSetResult (ByVal lAxisNo As Long, ByVal uHomeResult As Long) As Long
```

Delphi

```
function AxmHomeSetResult (lAxisNo : LongInt; uHomeResult : DWord) : DWord; stdcall;
```

Input / Output

Name	in/out	Format	Init Value	Explanation
AxisNo	in	long	-	원점검색 결과를 임의로 설정 할 채널(축) 번호(0 ~ (최대축수 - 1)))
HomeResult	out	double	FFh	원점검색 결과

HomeResult

#define	Value	Explanation
HOME_SUCCESS	01h	원점 검색이 성공적으로 종료 됐을 경우
HOME_SEARCHING	02h	현재 원점검색이 진행중일 경우
HOME_ERR_GNT_RANGE	10h	겐트리 구동축의 Master축과 Slave축의 원점검색 결과가 설정한 OffsetRange를 벗어났을경우
HOME_ERR_USER_BREAK	11h	원점 검색중 사용자가 정지 명령을 실행했을 경우
HOME_ERR_VELOCITY	12h	원점 검색의 속도 설정 값 중 한 개라도 0보다 작거나 같을경우

HOME_ERR_AMP_FAULT	13h	원점 검색중 서보팩 알람이 발생할 경우
HOME_ERR_NEG_LIMIT	14h	(+)방향으로 원점센서 검출 중 (-)리미트 센서가 감지될 경우
HOME_ERR_POS_LIMIT	15h	(-)방향으로 원점센서 검출 중 (+)리미트 센서가 감지될 경우
HOME_ERR_NOT_DETECT	16h	원점센서가 감지되지 않을경우
HOME_ERR_UNKNOWN	FFh	알수없는 채널(축) 번호(0부터 시작함)로 원점검색을 시작하려 했을경우

Return AXT_RT_SUCCESS(0000) : Successful execution of API

* See error code Table more information on status error codes

Description

원점검색 함수를 이용해 성공적으로 원점검색이 수행되고 나면 검색 결과가 HOME_SUCCESS로 설정된다. 이 함수는 사용자가 원점검색을 수행하지 않고 결과를 임의로 설정할 수 있다. 실제 원점검색 완료 후 정상 구동 중 예외상황이 발생하면 원점검색 결과를 HOME_ ERR_USER_BREAK 등으로 설정해 원점검색을 다시 수행 하도록 할 수 있다.

C Example

```
if(AxmHomeSetResult (0, HOME_ERR_USER_BREAK) == AXT_RT_SUCCESS){
 printf("0축 원점검색 결과를 사용자 BREAK로 설정하였습니다.");
}else{
 printf("0축 원점검색 결과 설정을 실패하였습니다.");
}

BOOL bRun = TRUE;
DWORD uHomeResult;
AxmHomeSetStart (0);

while(bRun)
{
// 원점검색 함수를 이용해 성공적으로 원점검색이 수행되고나면 검색 결과를 출력한다.
AxmHomeGetResult(0, &uHomeResult);
 switch(uHomeResult)
 {
 case HOME_ERR_UNKNOWN:   printf("알수 없는 축번호 임");
  bRun=FALSE;             break;
 case HOME_ERR_GNT_RANGE: printf("Gantry Offset을 벗어났음");
  bRun=FALSE;             break;
 case HOME_ERR_USER_BREAK:    printf("원점검색을 정지함");
  bRun=FALSE;             break;
 case HOME_ERR_VELOCITY:  printf("속도설정이 되어있지않음.");
```

```
 bRun=FALSE;          break;
case HOME_ERR_AMP_FAULT:     printf("서보팩 알람이 발생했음");
 bRun  = FALSE;       break;
case HOME_ERR_NEG_LIMIT:     printf("(-)리미트 센서가 검출됨");
 bRun  = FALSE;       break;
case HOME_ERR_POS_LIMIT:     printf("(+)리미트 센서가 검출됨");
 bRun  = FALSE;       break;
case HOME_SEARCHING:     printf("현재 원점검색 진행중 임");
 Sleep(100);             break;
case HOME_SUCCESS:       printf("원점검색 성공함");
 bRun  = FALSE;       break;
}
}
```

VB Example

```
If (AxmHomeSetResult (0, HOME_ERR_USER_BREAK) = AXT_RT_SUCCESS )Then
 MsgBox "0축 원점검색 결과를 사용자 BREAK로 설정하였습니다.", vbOKCancel
Else
 MsgBox "0축 원점검색 결과 설정을 실패하였습니다.", vbOKCancel
End If

Dim bFlag As Byte
Dim  uID As Long
Dim uHomeResult As Long
AxmHomeSetStart  0
'원점검색 함수를 이용해 성공적으로 원점검색이 수행되고나면 검색 결과를 출력한다.
bFlag   =  True
Do While bFlag
AxmHomeGetResult 0, uHomeResult
uID    = uHomeResult
Select Case uID
Case HOME_ERR_UNKNOWN:
 MsgBox "알수 없는 축번호 임", vbOKCancel
bFlag   =  False
Case HOME_ERR_GNT_RANGE:
 MsgBox "Gantry Offset을 벗어났음", vbOKCancel
bFlag   =  False
Case HOME_ERR_USER_BREAK:
 MsgBox "원점검색을 정지함", vbOKCancel
bFlag   =  False
Case HOME_ERR_VELOCITY:
 MsgBox "속도 설정이 되어있지않음.", vbOKCancel
bFlag   =  False
Case HOME_ERR_AMP_FAULT:
 MsgBox "서보팩 알람이 발생했음", vbOKCancel
bFlag   =  False
Case HOME_ERR_NEG_LIMIT:
 MsgBox "(-)리미트 센서가 검출됨", vbOKCancel
bFlag   =  False
Case HOME_ERR_POS_LIMIT:
 MsgBox "(+)리미트 센서가 검출됨", vbOKCancel
bFlag   =  False
```

```
Case HOME_SEARCHING:
 MsgBox "현재 원점검색 진행중 임", vbOKCancel
bFlag  = False
Case HOME_SUCCESS:
 MsgBox "원점검색 성공함", vbOKCancel
bFlag  = False
End Select
Loop
```

Delphi Example

```
begin
if (AxmHomeSetResult (0, HOME_ERR_USER_BREAK) = AXT_RT_SUCCESS) then
 Application.MessageBox('0축 원점검색 결과를 사용자 BREAK로 설정하였습니다.',
       'Ajinextek', MB_OK)
else
 Application.MessageBox('0축 원점검색 결과 설정을 실패하였습니다.', 'Ajinextek',
       MB_OK);
End;


var
uID : Word; bFlag : Boolean; bFlag := True;
uHomeResult: Dword

begin
AxmHomeSetStart (0);
{ 원점검색 함수를 이용해 성공적으로 원점검색이 수행되고나면 검색 결과를 출력한다.}
while bFlag do
begin
AxmHomeGetResult(0, @uHomeResult);
uID  := uHomeResult;
case (uID) of
HOME_ERR_UNKNOWN:  begin
Application.MessageBox ('알수 없는 축번호 임.', 'Ajinextek', MB_OK); bFlag :=
       False;  end;
HOME_ERR_GNT_RANGE: begin
Application.MessageBox ('Gantry Offset을 벗어났음', 'Ajinextek', MB_OK);
       bFlag := False;  end;
HOME_ERR_USER_BREAK: begin
Application.MessageBox ('원점검색을 정지함.', 'Ajinextek', MB_OK); bFlag :=
       False;  end;
HOME_ERR_VELOCITY: begin
Application.MessageBox ('속도 설정이 되어있지않음.', 'Ajinextek', MB_OK);
       bFlag := False;  end;
HOME_ERR_AMP_FAULT: begin
Application.MessageBox ('서보팩 알람이 발생했음.', 'Ajinextek', MB_OK);
       bFlag := False;  end;
HOME_ERR_NEG_LIMIT: begin
Application.MessageBox ('(-)리미트 센서가 검출됨.', 'Ajinextek', MB_OK);
       bFlag := False;  end;
HOME_ERR_POS_LIMIT: begin
Application.MessageBox ('(+)리미트 센서가 검출됨', 'Ajinextek', MB OK);
```

```
        bFlag := False;  end;
HOME_SEARCHING: begin
Application.MessageBox (‘현재 원점검색 진행중 임’, ‘Ajinextek’, MB_OK);
        bFlag := False;  end;
HOME_ SUCCESS: begin
Application.MessageBox (‘원점검색 성공함’, ‘Ajinextek’, MB_OK); bFlag :=
        False;  end;
end; end;
End;
```

See Also

AxmHomeSetMethod, AxmHomeGetMethod, AxmHomeSetVel, AxmHomeGetVel, AxmHomeSetStart, AxmHomeGetResult, AxmHomeGetRate

AxmHomeGetResult

Purpose

원점검색 함수의 검색 결과를 확인한다. 원점검색이 시작되면 HOME_SEARCHING으로 설정되며 원점검색에 실패하면 실패원인이 설정된다. 실패 원인을 제거한 후 다시 원점검색을 진행하면 된다.

Format

C

```
DWORD AxmHomeGetResult (long lAxisNo, DWORD *upHomeResult)
```

Visual Basic

```
Function AxmHomeGetResult (ByVal lAxisNo As Long, ByRef upHomeResult As Long) As Long
```

Delphi

```
function AxmHomeGetResult (lAxisNo : LongInt; upHomeResult : PDWord) : DWord; stdcall;
```

Input / Output

Name	in/out	Format	Init Value	Explanation
AxisNo	in	long	-	원점검색 결과를 임의로 설정 할 채널(축) 번호(0 ~ (최대축수 - 1)))
HomeResult	out	DWORD *	FFh	원점검색 결과

HomeResult

#define	Value	Explanation
HOME_SUCCESS	01h	원점 검색이 성공적으로 종료 됐을 경우
HOME_SEARCHING	02h	현재 원점검색이 진행중일 경우
HOME_ERR_GNT_RANGE	10h	겐트리 구동축의 Master축과 Slave축의 원점검색 결과가 설정한 OffsetRange를 벗어났을경우
HOME_ERR_USER_BREAK	11h	원점 검색중 사용자가 정지 명령을 실행했을 경우
HOME_ERR_VELOCITY	12h	원점 검색의 속도 설정 값 중 한 개라도 0보다 작거나 같을경우
HOME_ERR_AMP_FAULT	13h	원점 검색중 서보팩 알람이 발생할 경우
HOME_ERR_NEG_LIMIT	14h	(+)방향으로 원점센서 검출 중 (-)리미트 센서가 감지될 경우

HOME_ERR_POS_LIMIT	15h	(-)방향으로 원점센서 검출 중 (+)리미트 센서가 감지될 경우
HOME_ERR_NOT_DETECT	16h	원점센서가 감지되지 않을경우
HOME_ERR_UNKNOWN	FFh	알수없는 채널(축) 번호(0부터 시작함)로 원점검색을 시작하려 했을 경우 또는 원점검색을 실시 하지 않았을 경우

Return AXT_RT_SUCCESS(0000) : Successful execution of API

* See error code Table more information on status error codes

※ 알림: 원점 검색 성공 후 다시 원점검색을 수행하게되면 원점검색 결과가 초기화 되어 재설정 된다. 다시 진행중 정지명령(AxmMoveSStop)함수를 이용해 정지하면 HOME_USER_BREAK로 설정된다.

Description

원점검색 함수의 검색 결과를 확인한다. 원점검색이 시작되면 HOME_SEARCHING으로 설정되며 원점검색에 실패하면 실패원인이 설정된다. 실패 원인을 제거한 후 다시 원점검색을 진행하면 된다

C Example

```
if(AxmHomeSetResult (0, HOME_ERR_USER_BREAK) == AXT_RT_SUCCESS){
 printf("0축 원점검색 결과를 사용자 BREAK로 설정하였습니다.");
}else{
 printf("0축 원점검색 결과 설정을 실패하였습니다.");
}

BOOL bRun = TRUE;
DWORD uHomeResult;
AxmHomeSetStart (0);

while(bRun)
{
// 원점검색결과 확인
AxmHomeGetResult(0, &uHomeResult);
 switch(uHomeResult)
 {
 case HOME_ERR_UNKNOWN:   printf("알수 없는 축번호 임");
  bRun  = FALSE;       break;
 case HOME_ERR_GNT_RANGE: printf("Gantry Offset을 벗어났음");
  bRun  = FALSE;       break;
 case HOME_ERR_USER_BREAK:    printf("원점검색을 정지함");
  bRun  = FALSE;       break;
 case HOME_ERR_VELOCITY:  printf("속도 설정이 되어있지않음.");
  bRun  = FALSE;       break;
 case HOME_ERR_AMP_FAULT:     printf("서보팩 알람이 발생했음");
```

```
  bRun  = FALSE;        break;
 case HOME_ERR_NEG_LIMIT:     printf("(-)리미트 센서가 검출됨");
  bRun  = FALSE;        break;
 case HOME_ERR_POS_LIMIT:     printf("(+)리미트 센서가 검출됨");
  bRun  = FALSE;        break;
 case HOME_SEARCHING:     printf("현재 원점검색 진행중 임");
  Sleep(100);              break;
 case HOME_SUCCESS:       printf("원점검색 성공함");
  bRun  = FALSE;        break;
 }
 }
```

VB Example

```
If (AxmHomeSetResult (0, HOME_ERR_USER_BREAK) = AXT_RT_SUCCESS )Then
 MsgBox "0축 원점검색 결과를 사용자 BREAK로 설정하였습니다.", vbOKCancel
Else
 MsgBox "0축 원점검색 결과 설정을 실패하였습니다.", vbOKCancel
End If

Dim bFlag As Byte
Dim  uID As Long
Dim uHomeResult As Long
AxmHomeSetStart  0

bFlag  =  True
Do While bFlag
' 원점검색결과 확인
AxmHomeGetResult 0, uHomeResult
uID    = uHomeResult
Select Case uID
Case HOME_ERR_UNKNOWN:
 MsgBox "알수 없는 축번호 임", vbOKCancel
bFlag  =  False
Case HOME_ERR_GNT_RANGE:
 MsgBox "Gantry Offset을 벗어났음", vbOKCancel
bFlag  =  False
Case HOME_ERR_USER_BREAK:
 MsgBox "원점검색을 정지함", vbOKCancel
bFlag  =  False
Case HOME_ERR_VELOCITY:
 MsgBox "속도 설정이 되어있지않음.", vbOKCancel
bFlag  =  False
Case HOME_ERR_AMP_FAULT:
 MsgBox "서보팩 알람이 발생했음", vbOKCancel
bFlag  =  False
Case HOME_ERR_NEG_LIMIT:
 MsgBox "(-)리미트 센서가 검출됨", vbOKCancel
bFlag  =  False
Case HOME_ERR_POS_LIMIT:
 MsgBox "(+)리미트 센서가 검출됨", vbOKCancel
bFlag  =  False
Case HOME_SEARCHING:
```

```
MsgBox "현재 원점검색 진행중 임", vbOKCancel
bFlag  = False
Case HOME_SUCCESS:
MsgBox "원점검색 성공함", vbOKCancel
bFlag  = False
End Select
Loop
```

Delphi Example

```
begin
if (AxmHomeSetResult (0, HOME_ERR_USER_BREAK) = AXT_RT_SUCCESS) then
Application.MessageBox('0축 원점검색 결과를 사용자 BREAK로 설정하였습니다.',
       'Ajinextek', MB_OK)
else
Application.MessageBox('0축 원점검색 결과 설정을 실패하였습니다.', 'Ajinextek',
       MB_OK);
End;

var
uID : Word; bFlag : Boolean; bFlag := True;
uHomeResult: Dword

begin
AxmHomeSetStart (0);

while bFlag do
begin
{ 원점검색결과 확인 }
AxmHomeGetResult(0, @uHomeResult);
uID  := uHomeResult;
case (uID) of
HOME_ERR_UNKNOWN:  begin
Application.MessageBox ('알수 없는 축번호 임.', 'Ajinextek', MB_OK); bFlag :=
       False;  end;
HOME_ERR_GNT_RANGE: begin
Application.MessageBox ('Gantry Offset을 벗어났음', 'Ajinextek', MB_OK);
       bFlag := False;  end;
HOME_ERR_USER_BREAK: begin
Application.MessageBox ('원점검색을 정지함.', 'Ajinextek', MB_OK); bFlag :=
       False;  end;
HOME_ERR_VELOCITY: begin
Application.MessageBox ('속도 설정이 되어있지않음.', 'Ajinextek', MB_OK);
       bFlag := False;  end;
HOME_ERR_AMP_FAULT: begin
Application.MessageBox ('서보팩 알람이 발생했음.', 'Ajinextek', MB_OK);
       bFlag := False;  end;
HOME_ERR_NEG_LIMIT: begin
Application.MessageBox ('(-)리미트 센서가 검출됨.', 'Ajinextek', MB_OK);
       bFlag := False;  end;
HOME_ERR_POS_LIMIT: begin
Application.MessageBox ('(+)리미트 센서가 검출됨', 'Ajinextek', MB_OK);
       bFlag := False;  end;
```

```
HOME_SEARCHING: begin
Application.MessageBox ('현재 원점검색 진행중 임', 'Ajinextek', MB_OK);
       bFlag := False;  end;
HOME_ SUCCESS: begin
Application.MessageBox ('원점검색 성공함', 'Ajinextek', MB_OK); bFlag :=
       False;  end;
end; end;
End;
```

See Also

AxmHomeSetMethod, AxmHomeGetMethod, AxmHomeSetVel, AxmHomeGetVel, AxmHomeSetStart, AxmHomeSetResult, AxmHomeGetRate

AxmHomeGetRate

Purpose

원점검색 시작되면 진행율을 확인할 수 있다. 원점검색이 완료되면 성공여부와 관계없이 100을 반환하게 된다. 원점검색 성공여부는 GetHomeResult함수를 이용해 확인할 수 있다.

Format

C

```
DWORD AxmHomeGetRate (long lAxisNo DWORD * upHomeMainStepNumber, DWORD *upHomeStepNumber)
```

Visual Basic

```
Function AxmHomeGetRate (ByVal lAxisNo As Long, ByRef upHomeMainStepNumber As Long, ByRef upHomeStepNumber As Long) As Long
```

Delphi

```
function AxmHomeGetRate (lAxisNo : LongInt; upHomeStepNumber : PDWord ; upHomeStepNumber : PDWord) : DWord; stdcall;
```

Input / output

Name	in/out	Format	Init Value	Explanation
AxisNo	in	long	-	원점검색 진행율을 확인 할 채널(축) 번호 (0 ~ (최대축수 - 1)))
HomeMainStep Number	out	DWORD*	0	Main Step 진행율(0 – 100) -겐트리 FALSE일 경우upHomeMainStep Number : 0 일때면 선택한 축만 진행사항이고 홈 진행율은 upHomeStepNumber 표시한다. -겐트리 TRUE일 경우 upHomeMainStep Number : 0 일때면 마스터 홈을 진행사항이고 마스터 홈 진행율은 upHomeStep Number 표시한다. -겐트리 TRUE일 경우 upHomeMainStep Number : 10 일때면 슬레이브 홈을 진행

				사항이고 마스터 홈 진행율은 upHomeStepNumber 로 알 수 있음
HomeStepNumber	out	DWORD*	0	원점검색 진행율을 확인

HomeStepNumber

upHomeStepNumber	Value	Explanation
DWORD value	0	라이브러리가 초기화 되어 있지않을 경우, 유효하지않은 채널(축) 번호(0부터 시작함) 일 경우 원점검색이 진행중이지 않을 경우(단위:%)
DWORD value	1~99	원점검색 진행율(단위: %)
DWORD value	100	원점검색이 완료됐을 경우(단위:%)

Return AXT_RT_SUCCESS(0000) : Successful execution of API

* See error code Table more information on status error codes

Description

원점검색 시작되면 진행율을 확인할 수 있다. 원점검색이 완료되면 성공여부와 관계없이 100을 반환하게 된다. 원점검색 성공여부는 GetHomeResult함수를 이용해 확인할 수 있다.

C Example

```
BOOL bRun = TRUE;
DWORD uHomeResult, uHomeStepNumber,uHomeMainStepNumber;
AxmHomeSetStart (0);

while(bRun)
{
AxmHomeGetResult(0, &uHomeResult);
 switch(uHomeResult)
 {
 case HOME_ERR_UNKNOWN:   printf("알수 없는 축번호 임");
bRun  = FALSE;            break;
 case HOME_ERR_GNT_RANGE: printf("Gantry Offset을 벗어났음");
bRun = FALSE;             break;
 case HOME_ERR_USER_BREAK:    printf("원검검색을 정지함");
bRun = FALSE;             break;
 case HOME_ERR_VELOCITY:  printf("속도 설정이 되어있지않음.");
bRun = FALSE;             break;
 case HOME_ERR_AMP_FAULT:     printf("서보팩 알람이 발생했음");
bRun = FALSE;             break;
```

```
 case HOME_ERR_NEG_LIMIT:     printf("(-)리미트 센서가 검출됨");
bRun = FALSE;         break;
 case HOME_ERR_POS_LIMIT:     printf("(+)리미트 센서가 검출됨");
bRun = FALSE;         break;
 case HOME_SEARCHING:
// 진행율 확인
AxmHomeGetRate( 0, &uHomeMainStepNumber , &uHomeStepNumber);
printf("\t현재 원점검색  %d진행중", uHomeStepNumber);
Sleep(100);                                    break;
 case HOME_SUCCESS:          printf("원점검색 성공함");           bRun = FALSE
         break;
}
}
```

VB Example

```
Dim bFlag As Byte
Dim  uID As Long
Dim uHomeResult , uHomeMainStepNumber ,uHomeStepNumber As Long
bFlag   =  True
AxmHomeSetStart  0
Do While bFlag
AxmHomeGetResult 0, uHomeResult
uID    = uHomeResult
Select Case uID
Case HOME_ERR_UNKNOWN:
 MsgBox "알수 없는 축번호 임", vbOKCancel
bFlag   =  False
Case HOME_ERR_GNT_RANGE:
 MsgBox "Gantry Offset을 벗어났음", vbOKCancel
bFlag   =  False8
Case HOME_ERR_USER_BREAK:
 MsgBox "원점검색을 정지함", vbOKCancel
bFlag   =  False
Case HOME_ERR_VELOCITY:
 MsgBox "속도 설정이 되어있지않음.", vbOKCancel
bFlag   =  False
Case HOME_ERR_AMP_FAULT:
 MsgBox "서보팩 알람이 발생했음", vbOKCancel
bFlag   =  False
Case HOME_ERR_NEG_LIMIT:
 MsgBox "(-)리미트 센서가 검출됨", vbOKCancel
bFlag   =  False
Case HOME_ERR_POS_LIMIT:
 MsgBox "(+)리미트 센서가 검출됨", vbOKCancel
bFlag   =  False
Case HOME_SEARCHING:
' 진행율 확인
AxmHomeGetRate  0, uHomeMainStepNumber , uHomeStepNumber
nCount = uHomeStepNumber
       strData = "현재 원점검색 진행중 임. 진행률은 " + CStr(nCount) + " 입니다."
       MsgBox strData
bFlag   =  False
```

```
Case HOME_SUCCESS:
 MsgBox "원점검색 성공함", vbOKCancel
bFlag  = False
End Select
Loop
```

Delphi Example

```
var
uID : Word;  bFlag : Boolean; strData : String;
uHomeResult, uHomeStepNumber, uHomeMainStepNumber : Dword

begin
bFlag := True;

AxmHomeSetStart (0);

while bFlag do
begin
AxmHomeGetResult(0, @uHomeResult);
uID  := uHomeResult;
case (uID) of
HOME_ERR_UNKNOWN:  begin
Application.MessageBox ('알수 없는 축번호 임.', 'Ajinextek', MB_OK);
 bFlag := False;  end;
HOME_ERR_GNT_RANGE: begin
Application.MessageBox ('Gantry Offset을 벗어났음', 'Ajinextek', MB_OK);
       bFlag := False;  end;
HOME_ERR_USER_BREAK: begin
Application.MessageBox ('원점검색을 정지함.', 'Ajinextek', MB_OK);
bFlag := False;  end;
HOME_ERR_VELOCITY: begin
Application.MessageBox ('속도 설정이 되어있지않음.', 'Ajinextek', MB_OK);
       bFlag := False;  end;
HOME_ERR_AMP_FAULT: begin
Application.MessageBox ('서보팩 알람이 발생했음.', 'Ajinextek', MB_OK);
 bFlag := False;  end;
HOME_ERR_NEG_LIMIT: begin
Application.MessageBox ('(-)리미트 센서가 검출됨.', 'Ajinextek', MB_OK);
       bFlag := False;  end;
HOME_ERR_POS_LIMIT: begin
Application.MessageBox ('(+)리미트 센서가 검출됨', 'Ajinextek', MB_OK);
bFlag := False;  end;
HOME_SEARCHING: begin
{ 진행율 확인 }
AxmHomeGetRate( 0, @ uHomeMainStepNumber ,@uHomeStepNumber);
nCount    := uHomeStepNumber ; {// 0번축 원점검색 진행율 확인}
  strData    :=  '현재 원점검색 진행중 임. 진행률은' + IntToStr(nCount) + '입니
       다.';
  Application.MessageBox (PCHAR(strData), 'Ajinextek', MB_OK);
bFlag := False;  end;
HOME_SUCCESS: begin
Application.MessageBox ('원점검색 성공함', 'Ajinextek', MB OK); bFlag :=
```

```
        False;  end;
end;  end;

End;
```

See Also

AxmHomeSetMethod, AxmHomeGetMethod, AxmHomeSetVel, AxmHomeGetVel, AxmHomeSetStart, AxmHomeSetResult, AxmHomeGetResult,

위치구동 함수

이 단원에서는 단축(Single Axis) , 보간제어가 아닌 다축(Multi Axis) 모션제어에 관련된 함수들을 소개한다. 단축모션은 한 축만을 독립적으로 제어하는 작업을 의미하고 보간제어가 아닌 다축모션은 한 축 이상을 제어하는 작업을 의미한다. 단축모션, 보간제어가 아닌 다축모션은 속도, 가속도를 설정하고 이동작업을 수행한다. 그리고 필요에 따라 정지 함수를 사용하여 모션을 정지시킨다.

Function	Description
AxmMoveStartPos	지정 축의 절대 좌표로 설정된 위치까지 설정된 속도와 가속율로 구동을 한다. 속도 프로파일은 AxmMotSetProfileMode(함수에서 설정한다. 펄스가 출력되는 시점에서 함수를 벗어난다.
AxmMovePos	지정 축의 절대 좌표로 설정된 위치까지 설정된 속도와 가속율로 구동을 한다. 속도 프로파일은 AxmMotSetProfileMode(함수에서 설정한다. 펄스 출력이 종료되는 시점에서 함수를 벗어난다.
AxmMoveVel	지정 축에 대하여 설정된 속도와 가속율로 지속적으로 속도 모드 구동을 한다. 펄스 출력이 시작되는 시점에서 함수를 벗어난다.
AxmMoveStartMultiVel	지정된 다축에 대하여 설정된 속도와 가속율로 지속적으로 속도 모드 구동을 한다. 펄스 출력이 시작되는 시점에서 함수를 벗어난다.
AxmMoveSignalSearch	설정된 속도와 가속율로 설정된 신호가 검출될 때까지 구동을 한다. 펄스가 출력되는 시점에서 함수를 벗어난다.
AxmMoveSignalCapture	지정 축에서 설정된 신호를 검출하고 그 위치를 저장하기 위해 이동하는 함수이다.
AxmMoveGetCapturePos	‘AxmMoveSignalCapture’ 함수에서 저장된 위치값을 확인하는 함수이다.

AxmMoveStartMultiPos	지정 축들의 절대 좌표로 설정된 위치까지 설정된 속도와 가속율로 구동을 한다. 속도 프로파일은 AxmMotSetProfileMode(함수에서 설정한다. 펄스 출력이 종료되는 시점에서 함수를 벗어난다.
AxmMoveMultiPos	지정 축들의 절대 좌표로 설정된 위치까지 설정된 속도와 가속율로 구동을 한다. 속도 프로파일은 AxmMotSetProfileMode(함수에서 설정한다. 펄스가 출력되는 시점에서 함수를 벗어난다.
AxmMoveStop	지정 축을 설정한 감속도(감속도의 단위는 Unit/pulse를 1/1로 한경우에 PPS[Pulses/sec])로 감속 정지 한다.
AxmMoveEStop	지정 축을 급 정지 한다.
AxmMoveSStop	지정 축을 감속 정지한다.

* 단축구동 이란 일반적으로 모션 구동을 할 때 가장 기본이 되는 구동을 단축 구동이라고 한다. 단축구동은 크게 위치구동(Pos Drive), 속도구동(Velocity Drive), 신호 검색구동(Signal Search Drive)으로 이루어져 있다

* 속도구동 이란 속도, 가속도(가속도의 단위는 Unit/pulse를 1/1로 한경우에 PPS[Pulses/sec]), 가속시간을 정하여 종료 명령이 나올 때까지 계속 움직이는 형태의 구동이며, 속도구동 함수는 모두 펄스가 시작되는 시점에서 함수를 벗어나는 펄스 시점탈출함수이다.

*신호검색구동 이란 Limit 신호나 범용입출력 신호등의 검출대상을 미리 설정하고, 일정 속도로 움직이다가 설정한 신호가 감지될 경우 정지하는 구동을 뜻한다. 신호검색구동은 특정 신호가 검색되면 감속 또는 급정지하는 AxmMoveSignalSearch 함수와 신호가 검색되면 그 위치를 저장하는 AxmMoveSignalCapture함수가 있다. 검색할 수 있는 신호는 아래의 표에서와 같이 8개의 신호가 있는데, 각각 Up Edge와 Down Edge가 있으므로 총 16개의 신호가 있다.

*다축구동 이란 일반적으로 장비에서의 구동시스템은 여러 축을 한꺼번에 일정한 위치로 이동시켜야 하거나, 일정한 패턴으로 각 축들이 서로 연관된 운동을 해야 하는 경우도 많이 있다. 이러한 구동을 위해서 다축구동 함수를 제공한다. 다축구동을 위한 설정은 앞에서 해왔던 단축 구동설정을 모든 축에 대해서 각각 설정 해주면 된다. 예를 들어 축이 2개가 있다면, 단축구동함수에서의 설정을 0축과 1축에 대해서 각각 설정해주면 되는 것이다. 다축구동 함수는 여러 축을 설정된 위치와 속도로 한꺼번에 움직이게 하는 다축 위치구동과 특정 위치까지의 두 축 보간구동이 있다

AxmMoveStartPos

Purpose

지정 축의 설정된 위치까지 설정된 속도와 가속율로 구동을 한다.
펄스 출력이 시작되는 시점에서 함수를 벗어난다.

Format

C

```
DWORD AxmMoveStartPos (long lAxisNo, double dPos, double dVel, double dAccel, double dDecel);
```

Visual Basic

```
Function AxmMoveStartPos (ByVal lAxisNo As Long, ByVal dPos As Double, ByVal dVel As Double, ByVal dAccel As Double, ByVal dDecel As Double) As Long
```

Delphi

```
function AxmMoveStartPos (lAxisNo : LongInt; dPos : Double; dVel : Double; dAccel : Double; dDecel : Double) : DWord; stdcall;
```

Input/ Output

Name	in/out	Format	Init Value	Explanation
AxisNo	in	long	-	채널(축) 번호(0 ~ (최대축수 - 1))
Pos	in	double	200	구동 거리
Vel	in	double	200	속도
Accel	in	double	400	가속도 (가속도의 단위는 Unit/pulse를 1/1로 한경우에 PPS[Pulses/sec^2])
Decel	in	double	400	감속도(감속도의 단위는 Unit/pulse를 1/1로 한경우에 PPS[Pulses/sec^2])

Return AXT_RT_SUCCESS(0000) : Successful execution of API
* See error code Table more information on status error codes

Description

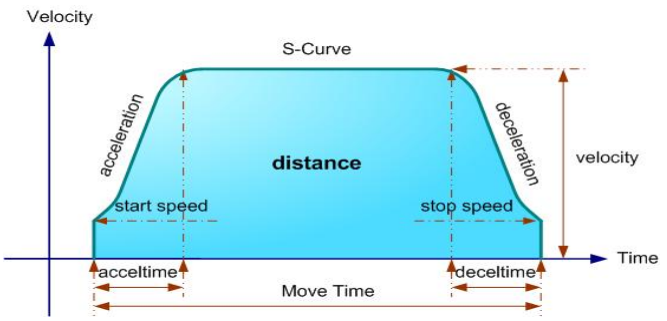
하나의 축에 대하여 현재의 위치에서 지정한 거리(절대/상대) 만큼 이동을 수행한다.
AxmMotSetAbsRelMode 함수를 이용하여 절대/상대를 선택하고 AxmMotSetProfileMode 함수를 이

용해 프로파일 선택한다. 절대위치 모드인 경우에는 설정한 위치값이 이동할 절대 위치값이 되고, 상대위치 모드인 경우는 설정한 위치값은 이동할 거리량이 된다. 예를 들어 위치 값을 '100'으로 입력한 경우, 절대 위치 모드인 경우는 '100'의 위치로 이동하여 정지하게 되며, 상대 위치 모드인 경우는 현재 위치에서 '100' 만큼 이동하게 된다.

▶ AxmStatusReadInMotion 함수에 의해 모션 중인지 확인할 수 있다.

▶ AxmSignalSetInpos함수를 사용하는 경우에는 Inpos 입력신호가 Enable로 설정되어 있다면 Command 펄스 출력이 완료해도 INP 입력이 ON 되기 전까지는 모션이 완료되지 않은 것으로 간주되어 반환 되지 않는다. 이 내용은 AxmSignalSetInpos함수를 참고하기 바란다.

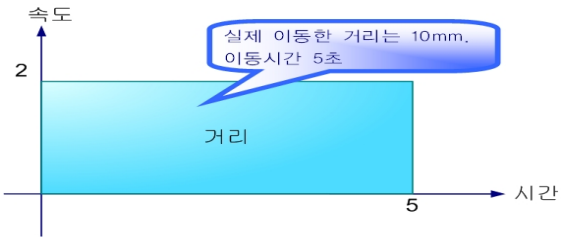
▶ AxmGantrySetEnable 을 이용해서 마스터,슬레이브를 사용할 경우 슬레이브축에 명령을 내렸을 경우에는 명령이 무시된다.



속도의 이해

속도 = 거리의 변화 / 시간의 변화
거리 = 속도 x 시간

속도는 시간의 변화에 대한 거리의 변화 비율이므로, 속도의 단위는 mm/s와 같은 것을 이용한다. 여기서는 mm는 이동거리 밀리미터를 나타내고, s는 시간 초를 나타낸다. 따라서 속도가 2 mm/s라고 하면 시간이 1초변할



가속도(가속도의 단위는 Unit/pulse 를 1/1 로 한경우에 PPS[Pulses/sec])의 이해

가속도가 0이라면 속도의 변화가 없다는 것을 의미한다. 즉 일정한 속도로움직이게 된다. 가속의 단위는 mm/s^2 을 사용한다. 여기서는 mm는 이동거리 밀리미터를 나타내고, s^{2}는 시간의 제곱이다.

가속도 = 속도의 변화 / 시간의 변화

C Example

```
// 0축에 절대 좌표로 2000만큼 100의 속도와 200의 가속율로 사다리꼴 구동,
   구동이 시작되면 함수를 빠져 나온다.
AxmMoveStartPos (0, 2000, 100, 200, 200);
```

VB Example

```
' 0축에 절대 좌표로 2000만큼 100의 속도와 200의 가속율로 사다리꼴 구동,
  구동이 시작되면 기다렸다 함수를 빠져 나온다.
AxmMoveStartPos0, 2000, 100, 200, 200
```

Delphi Example

```
begin
{ 0축에 절대 좌표로 2000만큼 100의 속도와 200의 가속율로 사다리꼴 구동,
   구동이 시작되면 기다렸다 함수를 빠져 나온다. }
AxmMoveStartPos (0, 2000, 100, 200, 200);
End;
```

See Also

AxmMovePos, AxmMoveVel, AxmMoveStartMultiVel, AxmMoveSignalSearch, AxmMoveSignalCapture, AxmMoveGetCapturePos, AxmMoveStartMultiPos, AxmMoveMultiPos, AxmMoveStop, AxmMoveEStop, AxmMoveSStop

AxmMovePos

Purpose

지정 축의 설정된 위치까지 설정된 속도와 가속율로 구동을 한다.
펄스 출력이 종료되는 시점에서 함수를 벗어난다.

Format

C

```
DWORD AxmMovePos (long lAxisNo, double dPos, double dVel, double dAccel, double dDecel);
```

Visual Basic

```
Function AxmMovePos (ByVal lAxisNo As Long, ByVal dPos As Double, ByVal dVel As Double,
ByVal dAccel As Double, ByVal dDecel As Double) As Long
```

Delphi

```
function AxmMovePos (lAxisNo : LongInt; dPos : Double; dVel : Double; dAccel : Double; dDecel :
Double) : DWord; stdcall;
```

Input/ Output

Name	in/out	Format	Init Value	Explanation
AxisNo	in	long	-	채널(축) 번호(0 ~ (최대축수 - 1))
Pos	in	double	200	구동 거리
Vel	in	double	200	속도
Accel	in	double	400	가속도 (가속도의 단위는 Unit/pulse를 1/1로 한경우에 PPS[Pulses/sec^2])
Decel	in	double	400	감속도(감속도의 단위는 Unit/pulse를 1/1로 한경우에 PPS[Pulses/sec^2])

Return AXT_RT_SUCCESS(0000) : Successful execution of API
* See error code Table more information on status error codes

Description

하나의 축에 대하여 현재의 위치에서 지정한 거리(절대/상대) 만큼 이동을 수행한다.
AxmMotSetAbsRelMode 함수를 이용하여 절대/상대를 선택하고 AxmMotSetProfileMode 함수를 이용하여 프로파일 설정한다. 지정 축의 설정된 위치까지 설정된 속도와 가속도로 구동을 한다. 펄스 출력이 종료되는 시점에서 함수를 벗어난다. 함수 시작 후 구동이 종료할 때까지 CPU의 process를 점

유하게 되므로 주의해서 사용하여야 한다. AxmStatusReadInMotion 함수에 의해 모션 중인지 확인할 수 없다. 절대위치 모드인 경우에는 설정한 위치값이 이동할 절대 위치값이 되고, 상대위치 모드인 경우는 설정한 위치값은 이동할 거리량이 된다.

예를 들어 위치 값을 '100'으로 입력한 경우, 절대 위치 모드인 경우는 '100'의 위치로 이동하여 정지하게 되며, 상대 위치 모드인 경우는 현재 위치에서 '100' 만큼 이동하게 된다.

▶ AxmSignalSetInpos함수를 사용하는 경우에는 Inpos 입력신호가 Enable로 설정되어 있다면 Command 펄스 출력이 완료해도 INP 입력이 ON 되기 전까지는 모션이 완료되지 않은 것으로 간주되어 반환 되지 않는다. 이 내용은 AxmSignalSetInpos함수를 참고하기 바란다.

▶ AxmGantrySetEnable 을 이용해서 마스터,슬레이브를 사용할 경우 슬레이브축에 명령을 내렸을 경우에는 명령이 무시된다.

C Example

```
// 0축에 절대 좌표로 2000만큼 100의 속도와 200의 가속율로 사다리꼴 구동, 구동이 종료될
       때까지 기다렸다 함수를 빠져 나온다.
AxmMovePos (0, 2000, 100, 200, 200);
```

VB Example

```
' 0축에 절대 좌표로 2000만큼 100의 속도와 200의 가속율로 사다리꼴 구동, 구동이 종료될 때
       까지 기다렸다 함수를 빠져 나온다.
AxmMovePos0, 2000, 100, 200, 200
```

Delphi Example

```
begin
{ 0축에 절대 좌표로 2000만큼 100의 속도와 200의 가속율로 사다리꼴 구동, 구동이 종료될 때
       까지 기다렸다 함수를 빠져 나온다. }
AxmMovePos (0, 2000, 100, 200, 200);
End;
```

See Also

AxmMoveStartPos, AxmMoveVel, AxmMoveStartMultiVel, AxmMoveRpm, AxmMoveSignalSearch, AxmMoveSignalCapture, AxmMoveGetCapturePos, AxmMoveStartMultiPos, AxmMoveMultiPos, AxmMoveStop, AxmMoveEStop, AxmMoveSStop

AxmMoveVel

Purpose

지정 축의 설정된 속도로 구동하는 속도모드 구동 함수이다.
펄스 출력이 시작되는 시점에서 함수를 벗어난다.

Format

C

```
DWORD AxmMoveVel (long lAxisNo, double dVel, double dAccel, double dDecel);
```

Visual Basic

```
Function AxmMoveVel (ByVal lAxisNo As Long, ByVal dVel As Double, ByVal dAccel As Double,
ByVal dDecel As Double) As Long
```

Delphi

```
function AxmMoveVel (lAxisNo : LongInt; dVel : Double; dAccel : Double; dDecel : Double) :
DWord; stdcall;
```

Input/ Output

Name	in/out	Format	Init Value	Explanation
AxisNo	in	long	-	채널(축) 번호(0 ~ (최대축수 - 1))
Vel	in	double	200	속도(+방향: 0 이상값 , -방향: 0이하값)
Accel	in	double	400	가속도 (가속도의 단위는 Unit/pulse를 1/1 로 한경우에 PPS[Pulses/sec^2])
Decel	in	double	400	감속도(감속도의 단위는 Unit/pulse를 1/1 로 한경우에 PPS[Pulses/sec^2])

Return AXT_RT_SUCCESS(0000) : Successful execution of API
* See error code Table more information on status error codes

Description

작업속도까지 가속한 후에 작업속도를 유지하며 정지함수가 호출될 때 또는 외부 센서 신호에 의해 정지 시킬 수 있으며 지정한 방향으로의 모션을 계속 수행한다. AxmStatusReadInMotion 함수에 의해 모션 중인지 확인할 수 있다.

▶ AxmGantrySetEnable 을 이용해서 마스터,슬레이브를 사용할 경우 슬레이브축에 명령을 내렸을 경

우에는 명령이 무시된다.

C Example

```
// 0축에 절대 좌표로 100의 속도와 200의 가속율로 속도 구동을 한다.
AxmMoveVel (0, 100, 200, 200);
```

VB Example

```
'0축에 절대 좌표로 100의 속도와 200의 가속율로 속도 구동을 한다.
AxmMoveVel0, 100, 200, 200
```

Delphi Example

```
begin
{ 0축에 절대 좌표로 100의 속도와 200의 가속율로 속도 구동을 한다. }
AxmMoveVel (0, 100, 200, 200);
End;
```

See Also

AxmMoveStartPos, AxmMovePos, AxmMoveStartMultiVel, AxmMoveRpm, AxmMoveSignalSearch, AxmMoveSignalCapture, AxmMoveGetCapturePos, AxmMoveStartMultiPos, AxmMoveMultiPos, AxmMoveStop, AxmMoveEStop, AxmMoveSStop

AxmMoveStartMultiVel

Purpose

지정 된 다축의 설정된 속도로 구동하는 속도모드 구동 함수이다.
펄스 출력이 시작되는 시점에서 함수를 벗어난다.

Format

C

```
DWORD AxmMoveStartMultiVel (long lArraySize, long *lpAxisNo, double *dpVel, double *dpAccel,
double *dpDecel);
```

Visual Basic

```
Function AxmMoveStartMultiVel (ByVal lArraySize As Long, ByRef lpAxisNo As Long, ByRef dpVel
As Double, ByRef dpAccel As Double, ByRef dpDecel As Double) As Long
```

Delphi

```
function AxmMoveStartMultiVel (lArraySize : LongInt; lpAxisNo : PLongInt; dpVel : PDouble;
dpAccel : PDouble; dpDecel : PDouble) : DWord; stdcall;
```

Input/ Output

Name	in/out	Format	Init Value	Explanation
ArraySize	in	long	-	축 사이즈
AxisNo	in	Long*	-	채널(축) 번호(0 ~ (최대축수 - 1)) 배열
Vel	in	Double*	200	속도(+방향: 0 이상값 , -방향: 0이하값) 배열
Accel	in	Double*	400	가속도 배열 (가속도의 단위는 Unit/pulse를 1/1로 한경우에 PPS[Pulses/sec^2])
Decel	in	Double*	400	감속도 배열 (감속도의 단위는 Unit/pulse를 1/1로 한경우에 PPS[Pulses/sec^2])

Return AXT_RT_SUCCESS(0000) : Successful execution of API

* See error code Table more information on status error codes

Description

다축 동시제어는 여러 개의 축을 완전한 동기를 맞추어 동시에 제어하는 기능을 말한다. 만일 속도 프로파일을 동일하게 설정하였다면 여러 개의 제어 대상 축이 시작 및 종료 시점은 물론이고 가속/감속 구간까지 완전히 동기를 맞추어 제어될 수 있다. 여러 개의 축에 대하여 Velocity Move 작업을 동시에 시작한다. Velocity Move는 작업속도까지 가속한 후에 작업속도를 유지하며 정지함수가 호출될 때까지 또는 외부 센서에 의해 정지할 수 있으며 지정한 방향으로 모션을 계속 수행한다. 이 함수를 사용하면 여러 개의 축이 동시에 작업을 시작하고 여러 축이 동기를 맞추어 작업을 시작해야 하는 경우 유용하게 사용될 수 있다.

▶ AxmStatusReadInMotion 함수에 의해 모션 중인지 확인할 수 있다.

▶ 한 노드에서 동기 구동을 할 수 있는 축은 최대 8축까지 지원된다

C Example

```
// 0축, 1축을  S커브 프로파일로 100의 속도와 200의 가속율로 속도 구동을 한다.

long   lAxis[2];
double dPos[2],dVel[2];
double dAccel[2],dDecel[2];

for( int i=0; i<2; i++)
{
     AxmMotSetProfileMode(i, 4);
     AxmMotSetAbsRelMode(i, 0);
     lAxis[i]              = i;
     dVel[i]  = 100;
     dAccel[i]  = 200;
     dDecel[i]  = 200;
}

AxmMoveStartMultiVel(2, lAxis, dVel, dAccel, dDecel);
```

VB Example

```
‘0축, 1축을 S커브 프로파일로 100의 속도와 200의 가속율로 속도 구동을 한다.
Dim dVelocity As Double
Dim dAcceleration As Double
Dim dDeceleration As Double
Dim axes(0 To 1) As Long
Dim velocities(0 To 1) As Double
Dim accelerations(0 To 1) As Double
Dim decelerations(0 To 1) As Double
Dim i As Long

dVelocity = 100
dAcceleration = 200
dDeceleration = 200
```

```
For i = 0 To 1
  AxmMotSetProfileMode i, 4
  axes(i) = i
  velocities(i) = dVelocity
  accelerations(i) = dAcceleration
  decelerations(i) = dDeceleration

Next i

AxmMoveStartMultiVel 2, axes(0), velocities2(0), accelerations(0),
       decelerations(0) ‘ 감속정지 설정
```

Delphi Example

```
var
   velocity, accel : Double;
   i : LongInt;
   lAxis : array [0..1] of LongInt;
   dVel : array [0..1] of Double;
   dAccel   : array [0..1] of Double;
   dDecel   : array [0..1] of Double;

begin
{ 0축, 1축을 S커브 프로파일로 100의 속도와 200의 가속율로 속도 구동을 한다. }

velocity := 100;
   accel := 200;

for i := 0 to 1 do
begin
  AxmMotSetProfileMode(i, 4);
  dVel[i]  := velocity;
  dAccel[i]  := accel;
  dDecel[i]  := accel;
  lAxis[i]                := i;
end;

AxmMoveStartMultiVel (2, @lAxis, @dVel, @dAccel, @dDecel)

End;
```

See Also

AxmMoveStartPos, AxmMovePos, AxmMoveVel, AxmMoveSignalSearch, AxmMoveSignalCapture, AxmMoveGetCapturePos, AxmMoveStartMultiPos, AxmMoveMultiPos, AxmMoveStop, AxmMoveEStop, AxmMoveSStop

AxmMoveSignalSearch

Purpose

지정 축에서 설정된 신호를 검출한다.

Format

C

```
DWORD AxmMoveSignalSearch (long lAxisNo, double dVel, double dAccel, long lDetectSignal,
long lSignalEdge, long lSignalMethod);
```

Visual Basic

```
Function AxmMoveSignalSearch (ByVal lAxisNo As Long, ByVal dVel As Double, ByVal dAccel As
Double, ByVal lDetectSignal As Long, ByVal lSignalEdge As Long, ByVal lSignalMethod As Long)
As Long
```

Delphi

```
function AxmMoveSignalSearch (lAxisNo : LongInt; dVel : Double; dAccel : Double; lDetectSignal :
LongInt; lSignalEdge : LongInt; lSignalMethod : LongInt) : DWord; stdcall;
```

Input

Name	in/out	Format	Init Value	Explanation
AxisNo	in	long	-	채널(축) 번호(0 ~ (최대축수 - 1))
Vel	in	double	200	속도(+방향: 0 이상값 , -방향: 0이하값)
Accel	in	double	400	가속도 (가속도의 단위는 Unit/pulse를 1/1로 한경우에 PPS[Pulses/sec^2])
DetectSignal	in	long	4	검출 신호 설정
SignalEdge	in	long	-	검출 신호의 Edge 방향 설정
SignalMethod	In	long	0	정지 모드

lDetectSignal

#define	Value	Explanation
PosEndLimit	00h	+Elm(End limit) +방향 리미트 센서 신호
NegEndLimit	01h	-Elm(End limit) -방향 리미트 센서 신호
PosSloLimit	02h	+Slm(Slow Down limit) 신호 - 사용하지 않음
NegSloLimit	03h	-Slm(Slow Down limit) 신호 - 사용하지 않음

HomeSensor	04h	IN0(ORG) 원점 센서 신호
EncodZPhase	05h	IN1(Z상) Encoder Z상 신호
UniInput02	06h	IN2(범용) 범용 입력 2번 신호
UniInput03	07h	IN3(범용) 범용 입력 3번 신호

lSignalEdge

#define	Value	Explanation
SIGNAL_DOWN_EDGE	00h	다운 에지
SIGNAL_UP_EDGE	01h	업 에지

lSignalMethod

#define	Value	Explanation
EMERGENCY_STOP	00h	급정지
SLOWDOWN_STOP	01h	감속정지

Return AXT_RT_SUCCESS(0000) : Successful execution of API
* See error code Table more information on status error codes

Description

지정 축에서 설정된 신호를 검출한다.
원점 센서신호, 서보 드라이버의 Z상 신호 또는 범용 입력 신호들의 Edge를 검출할 수 있다.
주로 원점검색 구동 함수 구현 시 사용하는 함수이다.

주의사항: lDetectSignal이 PosEndLimit , NegEndLimit(0,1) 을 찾을경우 신호의 Active 상태를 검출하게된다

▶ AxmStatusReadInMotion 함수에 의해 모션 중인지 확인할 수 있다.
▶ AxmSignalSetInpos함수를 사용하는 경우에는 Inpos 입력신호가 Enable로 설정되어 있다면 Command 펄스 출력이 완료해도 INP 입력이 ON 되기 전까지는 모션이 완료되지 않은 것으로 간주되어 반환 되지 않는다. 이 내용은 AxmSignalSetInpos함수를 참고하기 바란다.
주의사항: SignalMethod를 EMERGENCY_STOP(0)로 사용할경우 가감속이 무시되며 지정된 속도로 가속 급정지하게된다.

C Example

```
// 0축의 Z상 신호를 만나면 감속정지하면서 Falling Edge검출
AxmMoveSignalSearch (0, 100, 200, 5, 0, 0);
```

VB Example

```
'0축의 Z상 신호를 만나면 감속정지하면서 Falling Edge검출
AxmMoveSignalSearch 0, 100, 200, 5, 0, 0
```

Delphi Example

```
begin
{ 0축의 Z상 신호를 만나면 감속정지하면서 Falling Edge검출 }
AxmMoveSignalSearch (0, 100, 200, 5, 0, 0);
End;
```

See Also

AxmMoveStartPos, AxmMovePos, AxmMoveVel, AxmMoveStartMultiVel, AxmMoveSignalCapture, AxmMoveGetCapturePos, AxmMoveStartMultiPos, AxmMoveMultiPos, AxmMoveStop, AxmMoveEStop, AxmMoveSStop

AxmMoveSignalCapture

Purpose

지정 축에서 설정된 신호를 검출하고 그 위치를 저장하기 위해 이동하는 함수이다.

Format

C

```
DWORD AxmMoveSignalCapture (long lAxisNo, double dVel, double dAccel, long lDetectSignal,
long lSignalEdge, long lTarget, long lSignalMethod);
```

Visual Basic

```
Function AxmMoveSignalCapture (ByVal lAxisNo As Long, ByVal dVel As Double, ByVal dAccel As
Double, ByVal lDetectSignal As Long, ByVal lSignalEdge As Long, ByVal lTarget As Long, ByVal
lSignalMethod As Long) As Long
```

Delphi

```
function AxmMoveSignalCapture (lAxisNo : LongInt; dVel : Double; dAccel : Double; lDetectSignal :
LongInt; lSignalEdge : LongInt; lTarget : LongInt; lSignalMethod : LongInt) : DWord; stdcall;
```

Input/ Output

Name	in/out	Format	Init Value	Explanation
AxisNo	in	long	-	채널(축) 번호(0 ~ (최대축수 - 1))
Vel	in	double	200	속도(+방향: 0 이상값 , -방향: 0이하값)
Accel	in	double	400	가속도 (가속도의 단위는 Unit/pulse를 1/1로 한경우에 PPS[Pulses/sec^2])
DetectSignal	in	long	4	검출 신호 설정
SignalEdge	in	long	-	검출 신호의 Edge 방향 설정
Target	in	long	0	검출할 위치 소스 설정
SignalMethod	In	long	0	정지 모드

lDetectSignal

#define	Value	Explanation
PosEndLimit	00h	+Elm(End limit) +방향 리미트 센서 신호
NegEndLimit	01h	-Elm(End limit) -방향 리미트 센서 신호
PosSloLimit	02h	+Slm(Slow Down limit) 신호 - 사용하지 않음

NegSloLimit	03h	-Slm(Slow Down limit) 신호 - 사용하지 않음
HomeSensor	04h	IN0(ORG) 원점 센서 신호
EncodZPhase	05h	IN1(Z상) Encoder Z상 신호
UniInput02	06h	IN2(범용) 범용 입력 2번 신호
UniInput03	07h	IN3(범용) 범용 입력 3번 신호

lSignalEdge

#define	Value	Explanation
SIGNAL_DOWN_EDGE	00h	다운 에지
SIGNAL_UP_EDGE	01h	업 에지

lTarget

#define	Value	Explanation
COMMAND	00h	목표 위치 (Cmd 위치를 저장함)
ACTUAL	01h	실제 위치 (ACT (1) : Encoder 위치를 저장함)

lSignalMethod

#define	Value	Explanation
EMERGENCY_STOP	00h	급정지
SLOWDOWN_STOP	01h	감속정지

Return AXT_RT_SUCCESS(0000) : Successful execution of API
* See error code Table more information on status error codes

Description

지정 축에서 설정된 신호를 검출하고 그 위치를 저장하기 위해 이동하는 함수이다.
원점 센서신호, 서보 드라이버의 Z상 신호 또는 범용 입력 신호들의 Edge를 검출할 수 있다.
신호 검출 시 Cmd 위치나 Encoder 위치를 저장할 수 있다.
저장된 위치는 'AxmMoveGetCapturePos' 함수를 이용하여 확인할 수 있다.

주의사항: lDetectSignal이 PosEndLimit , NegEndLimit(0,1) 을 찾을경우 신호의레벨 Active 상태를 검출하게된다.

▶ AxmStatusReadInMotion 함수에 의해 모션 중인지 확인할 수 있다.

▶AxmSignalSetInpos함수를 사용하는 경우에는 Inpos 입력신호가 Enable로 설정되어 있다면 Command 펄스 출력이 완료해도 INP 입력이 ON 되기 전까지는 모션이 완료되지 않은 것으로 간주되

어 반환 되지 않는다. 이 내용은 AxmSignalSetInpos함수를 참고하기 바란다.

주의사항: SignalMethod를 EMERGENCY_STOP(0)로 사용할경우 가감속이 무시되며 지정된 속도로 가속 급정지하게된다.

C Example

```
// 0축의 Z상 신호 Falling Edge 검출 시 감속정지하면서 Encoder 위치 저장.
AxmMoveSignalCapture (0, 100, 200, 5, 0, ACTUAL, 0);
```

VB Example

```
'0축의 Z상 신호 Falling Edge 검출 시 감속정지하면서 Encoder 위치 저장.
AxmMoveSignalCapture 0, 100, 200, 5, 0, ACTUAL, 0
```

Delphi Example

```
begin
{ 0축의 Z상 신호 Falling Edge 검출 시 감속정지하면서 Encoder 위치 저장. }
AxmMoveSignalCapture (0, 100, 200, 5, 0, ACTUAL, 0);
End;
```

See Also

AxmMoveStartPos, AxmMovePos, AxmMoveVel, AxmMoveStartMultiVel, AxmMoveSignalSearch, AxmMoveGetCapturePos, AxmMoveStartMultiPos, AxmMoveMultiPos, AxmMoveStop, AxmMoveEStop, AxmMoveSStop

AxmMoveGetCapturePos

Purpose

'AxmMoveSignalCapture' 함수에서 저장된 위치 값을 확인하는 함수이다.

Format

C

```
DWORD AxmMoveGetCapturePos (long lAxisNo, double *dpCapPos);
```

Visual Basic

```
Function AxmMoveGetCapturePos (ByVal lAxisNo As Long, ByRef dpCapPos As Double) As Long
```

Delphi

```
function AxmMoveGetCapturePos (lAxisNo : LongInt; dpCapPotition : PDouble) : DWord; stdcall;
```

Input / Output

Name	in/out	Format	Init Value	Explanation
AxisNo	in	long	-	채널(축) 번호(0 ~ (최대축수 - 1))
dpCapPos	out	double	-	신호 검출 시 위치값

Return AXT_RT_SUCCESS(0000) : Successful execution of API

* See error code Table more information on status error codes

Description

'AxmMoveSignalCapture' 함수에서 저장된 위치 값을 확인하는 함수이다. 최초 실행시 AxmMoveSignalCapture에 의해 성공적으로 위치 값이 저장되어 있다면 "AXT_RT_SUCCESS"를 리턴하지만 그러지 않다면 "AXT_RT_NOT_CAPTURED"를 리턴하므로 함수 실행 결과를 확인하여야 한다.
주의) 이 함수를 사용하여 한번 읽혀진 결과는 초기화 된다.

▶ AxmStatusReadInMotion 함수에 의해 모션 중인지 확인할 수 있다.

▶AxmSignalSetInpos함수를 사용하는 경우에는 Inpos 입력신호가 Enable로 설정되어 있다면 Command 펄스 출력이 완료해도 INP 입력이 ON 되기 전까지는 모션이 완료되지 않은 것으로 간주되어 반환 되지 않는다. 이 내용은 AxmSignalSetInpos함수를 참고하기 바란다.

C Example

```
// Z상 검출 시 위치 확인.
double dPos;
AxmMoveGetCapturePos (0, &dPos);
```

VB Example

```
'Z상 검출 시 위치 확인.
Dim dPos As Double
AxmMoveGetCapturePos0, dPos
```

Delphi Example

```
{ Z상 검출 시 위치 확인. }
var
dAccel : Double;

begin
AxmMoveGetCapturePos (0, @dPos);
end;
```

See Also

AxmMoveStartPos, AxmMovePos, AxmMoveVel, AxmMoveStartMultiVel, AxmMoveSignalSearch, AxmMoveSignalCapture, AxmMoveStartMultiPos, AxmMoveMultiPos, AxmMoveStop, AxmMoveEStop, AxmMoveSStop

AxmMoveStartMultiPos

Purpose

여러 개의 축에 대해서 현재의 위치에서 지정한 거리만큼 이동을 동시에 시작한다. 이 함수를 사용하면 여러 개의 축이 동시에 작업을 시작한다. 이 함수는 여러 축이 동기를 맞추어 작업을 시작해야 하는 경우에 사용한다. 펄스 출력이 시작되는 시점에서 함수를 벗어난다.

Format

C

```
DWORD AxmMoveStartMultiPos ( long lArraySize, long *lpAxisNo, double *dpPos, double *dpVel,
double *dpAccel, double *dpDecel);
```

Visual Basic

```
Function AxmMoveStartMultiPos ByVal lArraySize As Long, ByRef lpAxisNo As Long, ByRef dpPos
As Double, ByRef dpVel As Double, ByRef dpAccel As Double, ByRef dpDecel As Double) As
Long
```

Delphi

```
function AxmMoveStartMultiPos (lArraySize : LongInt; lpAxisNo : PLongInt; dpPos : PDouble;
dpVel : PDouble; dpAccel : PDouble; dpDecel : PDouble) : DWord; stdcall;
```

Input/ Output

Name	in/out	Format	Init Value	Explanation
ArraySize	in	long	-	축 사이즈
AxisNo	in	long*	-	채널(축) 번호(0 ~ (최대축수 - 1)) 배열
Pos	in	double*	-	구동 거리 배열
Vel	in	double*	-	속도 배열
Accel	in	double*	-	가속도 (가속도의 단위는 Unit/pulse를 1/1로 한경우에 PPS[Pulses/sec^2]) 배열
Decel	in	double *	-	감속도(감속도의 단위는 Unit/pulse를 1/1로 한경우에 PPS[Pulses/sec^2]) 배열

Return AXT_RT_SUCCESS(0000) : Successful execution of API

* See error code Table more information on status error codes

Description

하나의 축에 대하여 현재의 위치에서 지정한 거리(절대/상대) 만큼 이동을 수행한다.
AxmMotSetAbsRelMode 함수를 이용하여 절대/상대를 선택하고 AxmMotSetProfileMode 함수를 이용해 프로파일을 선택한다. AxmStatusReadInMotion 함수에 의해 모션 중인지 확인할 수 있다.
절대위치 모드인 경우에는 설정한 위치값이 이동할 절대 위치값이 되고, 상대위치 모드인 경우는 설정한 위치값은 이동할 거리량이 된다. 예를 들어 위치 값을 '100'으로 입력한 경우, 절대 위치 모드인 경우는 '100'의 위치로 이동하여 정지하게 되며, 상대 위치 모드인 경우는 현재 위치에서 '100' 만큼 이동하게 된다.
다축 동시제어는 여러 개의 축을 완전한 동기를 맞추어 동시에 제어하는 기능을 말한다. 만일 속도 프로파일을 동일하게 설정하였다면 여러 개의 제어 대상 축이 시작 및 종료 시점은 물론이고 가속/감속 구간까지 완전히 동기를 맞추어 제어될 수 있다.

▶ AxmStatusReadInMotion 함수에 의해 모션 중인지 확인할 수 있다.

▶ AxmSignalSetInpos함수를 사용하는 경우에는 Inpos 입력신호가 Enable로 설정되어 있다면 Command 펄스 출력이 완료해도 INP 입력이 ON 되기 전까지는 모션이 완료되지 않은 것으로 간주되어 반환 되지 않는다. 이 내용은 AxmSignalSetInpos함수를 참고하기 바란다.

▶ 한 노드에서 동기 구동을 할 수 있는 축은 최대 8축까지 지원된다.

C Example

```
// 절대 좌표로 2000만큼 100의 속도와 200의 가속율로 사다리꼴 구동, 구동이 시작되면 함수
       를 빠져나온다.
long   lAxis[2];
double dPos[2],dVel[2], dAccel[2];
for( int i=0; i<2; i++)
{
       AxmMotSetProfileMode(i, 3);
       AxmMotSetAbsRelMode(i, 1);
       lAxis[i]            = i;
       dPos[i]         = 2000;
       dVel[i]  = 100;
dAccel[i]     = 200;
}
AxmMoveStartMultiPos (2, lAxis, dPos, dVel, dAccel, dAccel);
```

VB Example

```
' 절대 좌표로 2000만큼 100의 속도와 200의 가속율로 사다리꼴 구동, 구동이 시작되면 기다렸
       다 함수를 빠져나온다.
Dim Distance(0 To 2) As Double
Dim velocities(0 To 2) As Double
Dim accelerations(0 To 2) As Double
Dim axis(0 To 2) As Long
Dim i As Long

For i = 0 To 2
```

```
      AxmMotSetProfileMode i, 3
      AxmMotSetAbsRelMode i, 1
      axis(i) = i
      Distance(i) = 2000
      velocities(i) = 100
      accelerations(i) = 200

Next i

AxmMoveStartMultiPos 2 , axis(0), Distance(0), velocities(0),
       accelerations(0), accelerations(0)
```

Delphi Example

```
{ 절대 좌표로 2000만큼 100의 속도와 200의 가속율로 사다리꼴 구동, 구동이 시작되면 기다렸
       다 함수를 빠져나온다. }
var
i : LongInt;
   lAxis : array [0..1] of Longint;
   dPos : array [0..1] of Double;
   dVel : array [0..1] of Double;
dAccel : array [0..1] of Double
uStopMode : DWORD;

Begin
for i := 0 to 2 do
 begin
     AxmMotSetProfileMode(i, 3);
      AxmMotSetAbsRelMode (i, 1);
dPos[i]   := 2000;
dVel[i]  := 100;
dAccel[i]     := 200;
     lAxis[i]              := i;

 end;

AxmMoveStartMultiPos (2, @lAxis, @dPos, @dVel, @ dAccel, @dAccel);
end;
```

See Also

AxmMoveStartPos, AxmMovePos, AxmMoveVel, AxmMoveStartMultiVel, AxmMoveSignalSearch, AxmMoveSignalCapture, AxmMoveGetCapturePos, AxmMoveMultiPos, AxmMoveStop, AxmMoveEStop, AxmMoveSStop

AxmMoveMultiPos

Purpose

여러 개의 축에 대해서 현재의 위치에서 지정한 거리만큼 이동을 동시에 시작한다. 이 함수를 사용하면 여러 개의 축이 동시에 작업을 시작한다. 이 함수는 여러 축이 동기를 맞추어 작업을 시작해야 하는 경우에 사용한다. 펄스 출력이 종료되는 시점에서 함수를 벗어난다.

Format

C

```
DWORD AxmMoveMultiPos (long lArraySize, long *lpAxisNo, double *dpPos, double *dpVel,
double *dpAccel, double *dpDecel);
```

Visual Basic

```
Function AxmMoveMultiPos (ByVal lArraySize As Long, ByRef lpAxisNo As Long, ByRef dpPos As
Double, ByRef dpVel As Double, ByRef dpAccel As Double, ByRef dpDecel As Double) As Long
```

Delphi

```
function AxmMoveMultiPos(lArraySize : LongInt; lpAxisNo : PLongInt; dpPos : PDouble; dpVel :
PDouble; dpAccel : PDouble; dpDecel : PDouble) : DWord; stdcall;
```

Input/ Output

Name	in/out	Format	Init Value	Explanation
ArraySize	in	long	-	축 사이즈
AxisNo	in	long*	-	채널(축) 번호(0 ~ (최대축수 - 1)) 배열
Pos	in	double*	-	구동 거리 배열
Vel	in	double*	-	속도 배열
Accel	in	double*	-	가속도 (가속도의 단위는 Unit/pulse를 1/1로 한경우에 PPS[Pulses/sec^2]) 배열
Decel	in	double *	-	감속도(감속도의 단위는 Unit/pulse를 1/1로 한경우에 PPS[Pulses/sec^2]) 배열

Return AXT_RT_SUCCESS(0000) : Successful execution of API

* See error code Table more information on status error codes

Description

다축에 대하여 현재의 위치에서 지정한 거리(절대/상대) 만큼 이동을 수행한다.

AxmMotSetAbsRelMode 함수를 이용하여 절대/상대를 선택하고 AxmMotSetProfileMode 함수를 이용해 프로파일을 선택한다. 지정 축의 설정된 위치까지 설정된 속도와 가속도로 구동을 한다. 펄스 출력이 종료되는 시점에서 함수를 벗어난다. 함수 시작 후 구동이 종료할 때까지 CPU의 process를 점유하게 되므로 주의해서 사용하여야 한다. AxmStatusReadInMotion 함수에 의해 모션 중인지 확인할 수 없다. 절대위치 모드인 경우에는 설정한 위치값이 이동할 절대 위치값이 되고, 상대위치 모드인 경우는 설정한 위치값은 이동할 거리량이 된다.

예를 들어 위치 값을 '100'으로 입력한 경우, 절대 위치 모드인 경우는 '100'의 위치로 이동하여 정지하게 되며, 상대 위치 모드인 경우는 현재 위치에서 '100' 만큼 이동하게 된다.

▶ AxmStatusReadInMotion 함수에 의해 모션 중인지 확인할 수 있다.

▶ AxmSignalSetInpos함수를 사용하는 경우에는 Inpos 입력신호가 Enable로 설정되어 있다면 Command 펄스 출력이 완료해도 INP 입력이 ON 되기 전까지는 모션이 완료되지 않은 것으로 간주되어 반환 되지 않는다. 이 내용은 AxmSignalSetInpos함수를 참고하기 바란다. *

▶ 한 노드에서 동기 구동을 할 수 있는 축은 최대 8축까지 지원된다

C Example

```
// 절대 좌표로 2000만큼 100의 속도와 200의 가속율로 사다리꼴 구동, 구동이 시작되면 함수
       를 빠져나온다.

long   lAxis[2];
double dPos[2],dVel[2], dAccel[2];

for( int i=0; i<2; i++)
     {
       AxmMotSetProfileMode(i, 3);
       AxmSetAbsRealMode(i, 1);
       lAxis[i]            = i;
       dPos[i]         = 2000;
       dVel[i]  = 100;
dAccel[i]     = 200;
}
AxmMoveMultiPos (2, lAxis, dPos, dVel, dAccel, dAccel);
```

VB Example

```
' 절대 좌표로 2000만큼 100의 속도와 200의 가속율로 사다리꼴 구동, 구동이 시작되면 기다렸
       다 함수를 빠져나온다.

Dim Distance(0 To 2) As Double
Dim velocities(0 To 2) As Double
Dim accelerations(0 To 2) As Double
Dim axis(0 To 2) As Long
Dim i As Long
```

```
StopMode = 1; ‘ 스톱모드 감속정지설정
For i = 0 To 2

      AxmMotSetProfileMode i, 3
      AxmSetAbsRealMode i, 1
      axis(i) = i
      Distance(i) = 2000
      velocities(i) = 100
      accelerations(i) = 200

Next i

AxmMoveMultiPos 2 , axis(0), Distance(0), velocities(0), accelerations(0),
       accelerations(0)
```

Delphi Example

```
{ 절대 좌표로 2000만큼 100의 속도와 200의 가속율로 사다리꼴 구동, 구동이 시작되면 기다렸
       다 함수를 빠져나온다. }
var
i : LongInt;
   lAxis : array [0..1] of Longint;
   dPos : array [0..1] of Double;
   dVel : array [0..1] of Double;
dAccel : array [0..1] of Double

Begin
uStopMode = 1; { 스톱모드 감속정지설정}
for i := 0 to 2 do
 begin
     AxmMotSetProfileMode(i, 3);
    AxmSetAbsRealMode(i, 1);
dPos[i]   := 2000;
dVel[i]  := 100;
dAccel[i]     := 200;
     lAxis[i]              := i;

 end;

AxmMoveMultiPos (2, @lAxis, @dPos, @dVel, @ dAccel, @dAccel);
end;
```

See Also

AxmMoveStartPos, AxmMovePos, AxmMoveVel, AxmMoveStartMultiVel, AxmMoveSignalSearch, AxmMoveSignalCapture, AxmMoveGetCapturePos, AxmMoveStartMultiPos, AxmMoveStop, AxmMoveEStop, AxmMoveSStop

AxmMoveStop

Purpose

현재 구동 중인 모션 동작을 사용자가 준 감속율로 정지하는 함수이다.

Format

C

```
DWORD AxmMoveStop (long lAxisNo, double dDecel);
```

Visual Basic

```
Function AxmMoveStop (ByVal lAxisNo As Long, ByVal dDecel As Double) As Long
```

Delphi

```
function AxmMoveStop (lAxisNo : LongInt; dDecel : Double) : DWord; stdcall;
```

Input/ Output

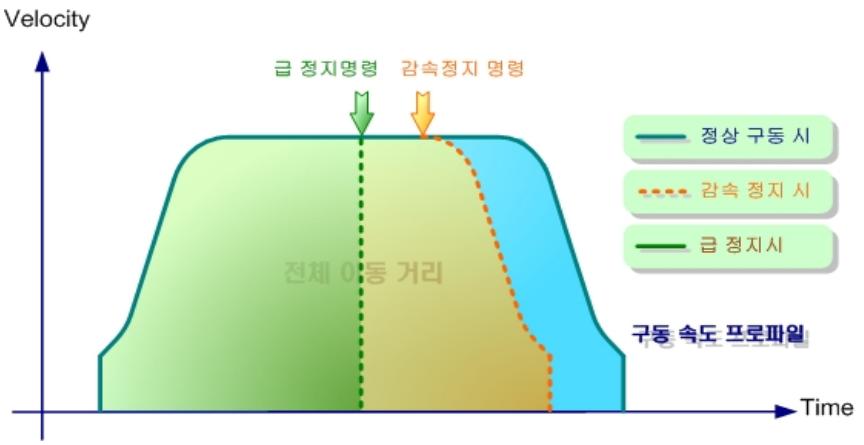
Name	in/out	Format	Init Value	Explanation
AxisNo	in	long	-	채널(축) 번호(0 ~ (최대축수 - 1))
Decel	in	double	400	정지 시 감속율값

Return AXT_RT_SUCCESS(0000) : Successful execution of API

* See error code Table more information on status error codes

Description

현재 구동 중인 모션 동작을 임의의 감속율로 모션을 중지하는 함수이다.



C Example

```
// 0축 모션 구동 급정지
AxmMoveStop(0, 0);
```

VB Example

```
' 0축 모션 구동 급정지
AxmMoveStop 0, 0
```

Delphi Example

```
begin
{ 0축 모션 구동 급정지 }
AxmMoveStop(0, 0);
End;
```

See Also

AxmMoveStartPos, AxmMovePos, AxmMoveVel, AxmMoveStartMultiVel, AxmMoveSignalSearch, AxmMoveSignalCapture, AxmMoveGetCapturePos, AxmMoveStartMultiPos, AxmMoveMultiPos, AxmMoveEStop, AxmMoveSStop

AxmMoveEStop

Purpose

현재 구동 중인 모션 동작을 급정지하는 함수이다.

Format

C

```
DWORD AxmMoveEStop (long lAxisNo);
```

Visual Basic

```
Function AxmMoveEStop (ByVal lAxisNo As Long) As Long
```

Delphi

```
function AxmMoveEStop (lAxisNo: LongInt) : Dword ; stdcall;
```

Input/ Output

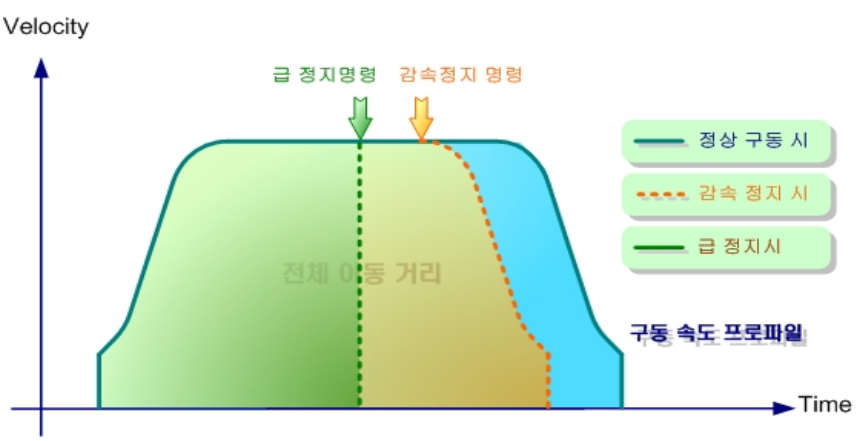
Name	in/out	Format	Init Value	Explanation
AxisNo	in	long	-	채널(축) 번호(0 ~ (최대축수 - 1))

Return AXT_RT_SUCCESS(0000) : Successful execution of API

* See error code Table more information on status error codes

Description

현재 구동 중인 모션 동작을 감속 없이 급정지하는 함수이다.



C Example

```
// 0축 모션 구동 급정지
AxmMoveEStop(0);
```

VB Example

```
'0축 모션 구동 급정지
AxmMoveEStop 0
```

Delphi Example

```
begin
{ 0축 모션 구동 급정지 }
AxmMoveEStop(0);
End;
```

See Also

AxmMoveStartPos, AxmMovePos, AxmMoveVel, AxmMoveStartMultiVel, AxmMoveSignalSearch, AxmMoveSignalCapture, AxmMoveGetCapturePos, AxmMoveStartMultiPos, AxmMoveMultiPos, AxmMoveStop, AxmMoveSStop

AxmMoveSStop

Purpose

현재 구동 중인 모션 동작을 감속 정지하는 함수이다.

Format

C

```
DWORD AxmMoveSStop (long lAxisNo);
```

Visual Basic

```
Function AxmMoveSStop (ByVal lAxisNo As Long) As Long
```

Delphi

```
function AxmMoveSStop (lAxisNo: LongInt) : Dword ; stdcall;
```

Input/ Output

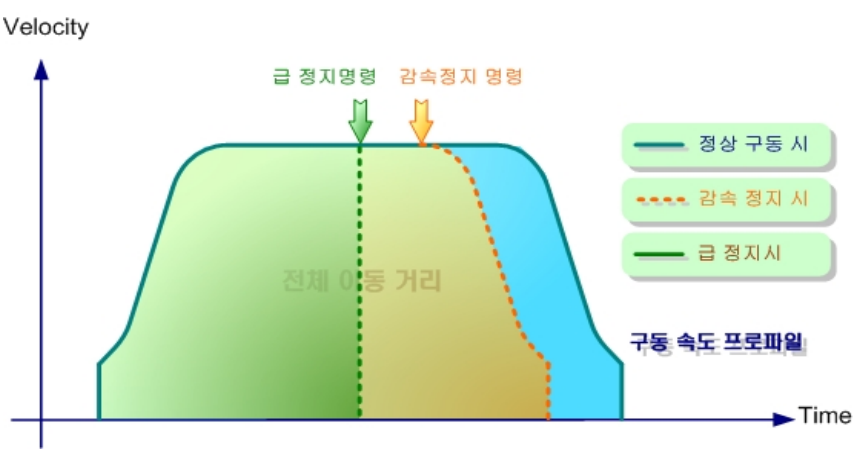
Name	in/out	Format	Init Value	Explanation
AxisNo	in	long	-	채널(축) 번호(0 ~ (최대축수 - 1))

Return AXT_RT_SUCCESS(0000) : Successful execution of API

* See error code Table more information on status error codes

Description

현재 구동 중인 처음에 사용자가 입력한 가속도만큼 모션 동작을 감속하면서 정지하는 함수이다.



C Example

```
// 0축 모션 구동 감속정지
AxmMoveSStop(0);
```

VB Example

```
'0축 모션 구동 감속정지
AxmMoveSStop 0
```

Delphi Example

```
begin
{ 0축 모션 구동 감속정지 }
AxmMoveSStop(0);
End;
```

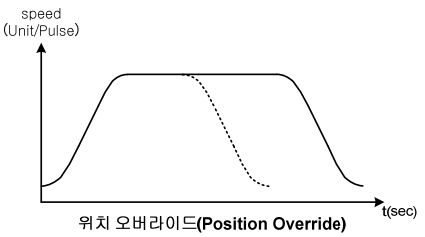
See Also

AxmMoveStartPos, AxmMovePos, AxmMoveVel, AxmMoveStartMultiVel, AxmMoveSignalSearch, AxmMoveSignalCapture, AxmMoveGetCapturePos,AxmMoveStartMultiPos, AxmMoveMultiPos, AxmMoveStop, AxmMoveEStop

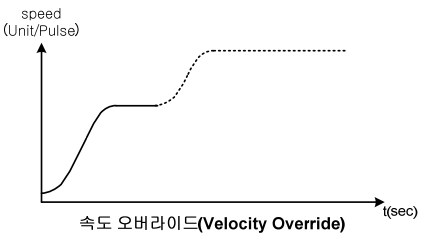
위치 및 속도 오버라이드 구동 함수

이 장에서는 속도 및 위치 오버라이드 구동에 대해서 소개한다. 속도 오버라이드 구동은 모션이 진행되는 도중에 작업 속도를 변경하는 것을 의미하고 위치 오버라이드 드라이브는 모션을 수행하고 있는 중에 목표거리를 변경하는 것을 의미한다.

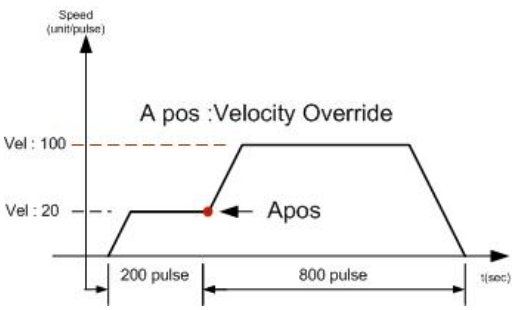
Function	Description
AxmOverridePos	지정 축의 구동이 종료되기 전 지정된 출력 펄스 수를 조정한다.
AxmOverrideSetMaxVel	지정 축의 속도오버라이드 하기전에 오버라이드할 최고속도를 설정한다.
AxmOverrideVel	지정 축의 구동 중에 속도를 가변 설정한다.
AxmOverrideAccelVelDecel	설정 축의 모션 동작 중 구동 속도를 변경하는 함수이다
AxmOverrideVelAtPos	지정 축에 대하여 설정된 속도와 가속율로 지속적으로 속도 모드 구동을 한다. 구동 중 설정한 위치에서 속도오버라이드를 실행한다. 펄스 출력이 시작되는 시점에서 함수를 벗어난다.



위치 오버라이드(Position Override)



속도 오버라이드(Velocity Override)



AxmOverridePos

Purpose

설정 축의 모션 동작 중 설정한 이동 거리를 변경하는 함수이다.

Format

C

```
DWORD AxmOverridePos (long lAxisNo, double dOverridePos);
```

Visual Basic

```
Function AxmOverridePos (ByVal lAxisNo As Long, ByVal dOverridePos As Double) As Long
```

Delphi

```
function AxmOverridePos (lAxisNo : LongInt; dOverridePos : Double) : DWord; stdcall;
```

Input / Output

Name	in/out	Format	Init Value	Explanation
AxisNo	in	long	-	채널(축) 번호(0 ~ (최대축수 - 1))
OverridePos	In	double	-	변경할 이동 거리

Return AXT_RT_SUCCESS(0000) : Successful execution of API

* See error code Table more information on status error codes

Description

AxmMoveStartPos와 같은 단축 위치 지정 모션 함수를 이용하여 모션 중일 때 이 함수를 사용하여 구동 시작시 설정한 최종 목표 위치를 새로운 위치값으로 변경한다.

▶ 모션 가속, 등속 중일때 사용 가능하며, 모션이 감속 중이거나 정지한 후에 지령을 내려도 무시된다. 따라서 사용자가 오버라이드 지령을 내려도 무시되면 이미 모션이 완료된 상태라고 인지하여야 한다.

▶ 직선, 원호, 스플라인, 헬리컬 보간작업을 수행하는 경우는 오버라이드를 사용할 수 없다.

주의사항 1: 오버라이드 할 위치값은 현재 구동 방향에 대한 Relative 형태의 이동 거리값으로 들어가기 때문에 위치는 반드시 Relative 형태의 위치값으로 넣어준다.
주의사항 2 : 경우 최초 구동 거리보다 작은 값으로 오버라이드시 현재 구동 거리가 오버라이드 거리를 이미 지났을 경우 감속 정지 후 오버라이드 위치인 반대 방향으로 구동을 한다. 이때 또 다시 오버라이드를 실행하면 무시된다.

C Example

```
AxmMotSetProfileMode(0, 4);      // 비대칭 S Curve
AxmMotSetAbsRelMode(0, 0);
AxmMoveStartPos(0, 1000, 200, 800, 800);

// 0축 구동 중 위치 2000 위치로 오버라이드.
AxmOverridePos (0, 2000);
```

VB Example

```
AxmMotSetProfileMode 0, 4 ‘ 비대칭 S Curve
AxmMotSetAbsRelMode 0, 0
AxmMoveStartPos 0, 1000, 200, 800, 800

‘0축 구동 중 위치 2000 위치로 오버라이드.
AxmOverridePos 0, 2000
```

Delphi Example

```
begin
{ 0축 구동 중 위치 2000 위치로 오버라이드. }

AxmMotSetProfileMode(0, 4);  { 비대칭 S Curve}
AxmMotSetAbsRelMode(0, 0);
AxmMoveStartPos(0, 1000, 200, 800, 800);

AxmOverridePos (0, 2000);
End;
```

See Also

AxmOverrideSetMaxVel, AxmOverrideVel, AxmOverrideAccelVelDecel, AxmOverrideVelAtPos

AxmOverrideSetMaxVel

Purpose

지정 축의 속도오버라이드 하기 전에 오버라이드할 최고속도를 설정한다. 구동중 현재 구동하려는 속도 보다 높은 속도로 오버라이드 해야 될경우 해당함수를 이용하여 구동전 예상되는 최대 속도를 지정한다. 만약 현재 구동하려는 속도보다 낮은 속도로만 오버라이드 할 경우는 해당함수를 사용하지 않아도 된다.

Format

C

DWORD AxmOverrideSetMaxVel (long lAxisNo, double dOverrideMaxVel);

Visual Basic

Function AxmOverrideSetMaxVel (ByVal lAxisNo As Long, ByVal dOverrideMaxVel As Double) As Long

Delphi

function AxmOverrideSetMaxVel (lAxisNo : LongInt; dOverrideMaxVel: Double) : Dword ; stdcall;

Input / Output

Name	in/out	Format	Init Value	Explanation
AxisNo	in	long	-	채널(축) 번호(0 ~ (최대축수 - 1))
OverrideMaxVel	In	double	-	오버라이드할 최대 속도

Return AXT_RT_SUCCESS(0000) : Successful execution of API

* See error code Table more information on status error codes

Description

단축 모션이 진행되고 있는 중에 속도를 오버라이드 하고자 할 때 사용한다. 먼저 이 함수를 사용하기 위해서 AxmMoveStartPos와 같은 단축모션 함수를 이용하여 모션 중이 아닐 때 사용하여야 한다.

주의점 : 속도오버라이드를 5번 한다면 그 중에 최고 속도를 설정해야 된다.

C Example

```
AxmOverrideSetMaxVel(0, 500);  // 오버라이드 최대값 설정

AxmMotSetProfileMode(0, 4);      // 비대칭 S Curve
AxmMotSetAbsRelMode(0, 0);
```

```
AxmMoveStartPos(0, 1000, 200, 800, 800);

// 0축 모션 구동 중 속도값을 500으로 오버라이드.
AxmOverrideVel (0, 500);
```

VB Example

```
AxmOverrideSetMaxVel 0, 500  ‘ 오버라이드 최대값 설정

AxmMotSetProfileMode 0, 4 ‘ 비대칭 S Curve
AxmMotSetAbsRelMode 0, 0
AxmMoveStartPos 0, 1000, 200, 800, 800

‘0축 모션 구동 중 속도값을 500으로 오버라이드.
AxmOverrideVel 0, 500
```

Delphi Example

```
Begin

AxmOverrideSetMaxVel(0, 500);  { 오버라이드 최대값 설정}

AxmMotSetProfileMode(0, 4);  { 비대칭 S Curve}
AxmMotSetAbsRelMode(0, 0);
AxmMoveStartPos(0, 1000, 200, 800, 800);

{ 0축 모션 구동 중 속도값을 500으로 오버라이드. }
AxmOverrideVel (0, 500);
End;
```

See Also

AxmOverridePos, AxmOverrideVel, AxmOverrideAccelVelDecel, AxmOverrideVelAtPos

AxmOverrideVel

Purpose

설정 축의 모션 동작 중 구동 속도를 변경하는 함수이다.

Format

C

```
DWORD AxmOverrideVel (long lAxisNo, double dOverrideVelocity);
```

Visual Basic

```
Function AxmOverrideVel (ByVal lAxisNo As Long, ByVal dOverrideVelocity As Double) As Long
```

Delphi

```
function AxmOverrideVel (lAxisNo : LongInt; dOverrideVelocity: Double ) : Dword ; stdcall;
```

Input / Output

Name	in/out	Format	Init Value	Explanation
AxisNo	in	long	-	채널(축) 번호(0 ~ (최대축수 - 1)))
OverrideVelocity	In	double	-	변경할 구동 속도[Unit/Sec]

Return AXT_RT_SUCCESS(0000) : Successful execution of API

* See error code Table more information on status error codes

Description

단축 모션이 진행되고 있는 중에 속도를 오버라이드 하고자 할 때 사용한다. 먼저 이 함수를 사용하기 위해서 AxmMoveStartPos와 같은 단축모션 함수를 이용하여 모션 중일 때 사용하여야 한다.

주의점: AxmOverrideVel 함수를 사용하기 전에. AxmOverrideSetMaxVel 최고로 설정할 수 있는 속도를 설정해놓는다. 가속구간과 감속구간에는 오버라이드를 할수없으므로 반드시 등속구간에 오버라이드를 수행한다.

EX> 속도오버라이드를 두 번한다면

1. 두 개 중에 높은 속도를 AxmOverrideSetMaxVel 설정 최고 속도값 설정.
2. AxmMoveStartPos실행 지정 축의 구동 중(Move함수 모두 포함)에 속도를 첫번째 속도로 AxmOverrideVel 가변 설정한다.
3. 지정 축의 구동 중(Move함수 모두 포함)에 속도를 두번째 속도로 AxmOverrideVel 가변 설정한다.

▶ 가속, 등속 중 일때 사용 가능하며, 감속 중 이거나 정지한후에 명령을 내리면 무시된다.

따라서 사용자가 오버라이드 지령이 내려도 무시되면 이미 모션이 완료되었거나 감속중이라고 인지하여야 한다.

▶ 직선, 원호, 스플라인, 헬리컬 보간작업을 수행하는 경우는 오버라이드를 사용할 수 없다.

Example

```
AxmMotSetProfileMode(0, 4);      // 비대칭 S Curve
AxmMotSetAbsRelMode(0, 0);
AxmMoveStartPos(0, 1000, 200, 800, 800);

// 0축 모션 구동 중 속도값을 500으로 오버라이드.
AxmOverrideVel (0, 500);
```

VB Example

```
AxmMotSetProfileMode 0, 4 ‘ 비대칭 S Curve
AxmMotSetAbsRelMode 0, 0
AxmMoveStartPos 0, 1000, 200, 800, 800

‘0축 모션 구동 중 속도값을 500으로 오버라이드.
AxmOverrideVel 0, 500
```

Delphi Example

```
Begin
AxmMotSetProfileMode(0, 4);  { 비대칭 S Curve}
AxmMotSetAbsRelMode(0, 0);
AxmMoveStartPos(0, 1000, 200, 800, 800);

{ 0축 모션 구동 중 속도값을 500으로 오버라이드. }
AxmOverrideVel (0, 500);
End;
```

See Also

AxmOverridePos, AxmOverrideSetMaxVel, AxmOverrideAccelVelDecel, AxmOverrideVelAtPos

AxmOverrideAccelVelDecel

Purpose

설정 축의 모션 동작 중 구동 속도를 변경하는 함수이다.

Format

C

```
DWORD AxmOverrideAccelVelDecel (long lAxisNo, double dOverrideVelocity, double dMaxAccel,
double dMaxDecel);
```

Visual Basic

```
Function AxmOverrideAccelVelDecel (ByVal lAxisNo As Long, ByVal dOverrideVelocity As Double,
ByVal dMaxAccel As Double, ByVal dMaxDecel As Double) As Long
```

Delphi

```
function AxmOverrideAccelVelDecel (lAxisNo : LongInt; dOverrideVelocity: Double; dMaxAccel:
Double; dMaxDecel: Double ) : Dword ; stdcall;
```

Input / Output

Name	in/out	Format	Init Value	Explanation
AxisNo	in	long	-	채널(축) 번호(0 ~ (최대축수 - 1))
OverrideVelocity	In	double	-	변경할 구동 속도[Unit/Sec]
dMaxAccel	In	double	-	변경할 구동 가속도[Unit/Sec^2]
dMaxDecel	In	double	-	변경할 구동 감속도[Unit/Sec^2]

Return AXT_RT_SUCCESS(0000) : Successful execution of API.
* See error code Table more information on status error codes

Description

단축 모션이 진행되고 있는 중에 속도를 오버라이드 하고자 할 때 사용한다. 먼저 이 함수를 사용하기 위해서 AxmMoveStartPos와 같은 단축모션 함수를 이용하여 모션 중일 때 사용하여야 한다.

주의점: AxmOverrideAccelVelDecel 함수를 사용하기 전에. AxmOverrideSetMaxVel 최고로 설정할 수 있는 속도를 설정해놓는다. 가속구간과 감속구간에는 오버라이드를 할수없으므로 반드시 등속구간에 오버라이드를 수행한다.

EX> 속도오버라이드를 두 번한다면

1. 두 개 중에 높은 속도를 AxmOverrideSetMaxVel 설정 최고 속도값 설정.

2. AxmMoveStartPos실행 지정 축의 구동 중(Move함수 모두 포함)에 속도를 첫번째 속도로 AxmOverrideAccelVelDecel 가변 설정한다.
3. 지정 축의 구동 중(Move함수 모두 포함)에 속도를 두번째 속도로 AxmOverrideAccelVelDecel 가변 설정한다.

▶ 모션 가속, 등속중일때 사용가능하며, 모션이 감속이 정지한후에 지령이 내려도 무시된다.
따라서 사용자가 오버라이드 지령이 내려도 무시되면 이미 모션이 완료된 상태라고 인지하여야 한다.
▶ 직선, 원호, 스플라인, 헬리컬 보간작업을 수행하는 경우는 오버라이드를 사용할 수 없다.

C Example

```
AxmOverrideSetMaxVel(0, 500);  // 오버라이드 최대값 설정

AxmMotSetProfileMode(0, 4);      // 비대칭 S Curve
AxmMotSetAbsRelMode(0, 0);
AxmMoveStartPos(0, 1000, 200, 800, 800);

// 0축 모션 구동 중 속도값을 500으로 오버라이드.
AxmOverrideAccelVelDecel (0, 500, 4000, 4000);
```

VB Example

```
AxmOverrideSetMaxVel 0, 500  ‘ 오버라이드 최대값 설정

AxmMotSetProfileMode 0, 4 ‘ 비대칭 S Curve
AxmMotSetAbsRelMode 0, 0
AxmMoveStartPos 0, 1000, 200, 800, 800

‘0축 모션 구동 중 속도값을 500으로 오버라이드.
AxmOverrideAccelVelDecel 0, 500, 4000, 4000
```

Delphi Example

```
Begin

AxmOverrideSetMaxVel(0, 500);  { 오버라이드 최대값 설정}

AxmMotSetProfileMode(0, 4);  { 비대칭 S Curve}
AxmMotSetAbsRelMode(0, 0);
AxmMoveStartPos(0, 1000, 200, 800, 800);

{ 0축 모션 구동 중 속도값을 500으로 오버라이드. }
AxmOverrideAccelVelDecel (0, 500, 4000, 4000);
End;
```

See Also

AxmOverridePos, AxmOverrideSetMaxVel, AxmOverrideVel, AxmOverrideVelAtPos

AxmOverrideVelAtPos

Purpose

설정 축의 모션 동작 중 사용자가 지정한 위치에서 구동 속도를 변경하는 함수이다.

Format

C

```
DWORD AxmOverrideVelAtPos (long lAxisNo, double dPos, double dVel, double dAccel, double dDecel, double dOverridePos, double dOverrideVelocity, long lTarget);
```

Visual Basic

```
Function AxmOverrideVelAtPos (ByVal lAxisNo As Long, ByVal dPos As Double, ByVal dVel As Double, ByVal dAccel As Double, ByVal dDecel As Double, ByVal dOverridePos As Double, ByVal dOverrideVelocity As Double, ByVal lTarget As Long) As Long
```

Delphi

```
function AxmOverrideVelAtPos (lAxisNo : LongInt; dPos : Double; dVel : Double; dAccel : Double; dDecel : Double; dOverridePos : Double; dOverrideVelocity : Double; lTarget : LongInt) : DWord; stdcall;
```

Input / Output

Name	in/out	Format	Init Value	Explanation
AxisNo	in	long	-	채널(축) 번호(0 ~ (최대축수 - 1))
Pos	in	double	200	구동 거리
Vel	in	double	200	구동속도(+방향: 0 이상값 , -방향: 0이하 값)
Accel	in	double	400	가속도 (가속도의 단위는 Unit/pulse를 1/1 로 한경우에 PPS[Pulses/sec^2])
Decel	in	double	400	감속도 (감속도의 단위는 Unit/pulse를 1/1 로 한경우에 PPS[Pulses/sec^2])
OverridePos	in	double	-	속도를 변경할 위치
OverrideVelocity	in	double	-	변경할 구동 속도
Target	In	long	-	속도를 변경할 위치 소스 선택

Target

#define	Value	Explanation
COMMAND	00h	목표 위치(Cmd 위치)
ACTUAL	01h	실제 위치(Encoder 위치)

Return AXT_RT_SUCCESS(0000) : Successful execution of API
* See error code Table more information on status error codes

Description

설정한 위치까지 이동하면서 사용자가 지정한 특정 위치에서 설정한 속도로 변경한다. 특정 위치에서 고속이나 저속으로 이동해야 할 경우 사용하는 함수이다.

주의 점 : AxmOverrideVelAtPos 함수를 사용하기 전에, AxmOverrideSetMaxVel 최고로 설정할 수 있는 속도를 설정해 놓는다.

▶ AxmStatusReadInMotion 함수에 의해 모션 중인지 확인할 수 있다.
▶ AxmSignalSetInpos함수를 사용하는 경우에는 Inpos 입력신호가 Enable로 설정되어 있다면 Command 펄스 출력이 완료해도 INP 입력이 ON 되기 전까지는 모션이 완료되지 않은 것으로 간주되어 반환 되지 않는다. 이 내용은 AxmSignalSetInpos함수를 참고하기 바란다.

C Example

```
// 0축 50000만큼 100의 속도와 200의 가속율로 구동, 10000 위치에서 500 속도 오버라이
       드.
AxmOverrideVelAtPos (0, 50000, 100, 200, 200, 10000, 500,  ACT);
```

VB Example

```
'0축 50000만큼 100의 속도와 200의 가속율로 구동, 10000 위치에서 500 속도 오버라이드.
AxmOverrideVelAtPos 0, 50000, 100, 200, 200, 10000, 500, ACT
```

Delphi Example

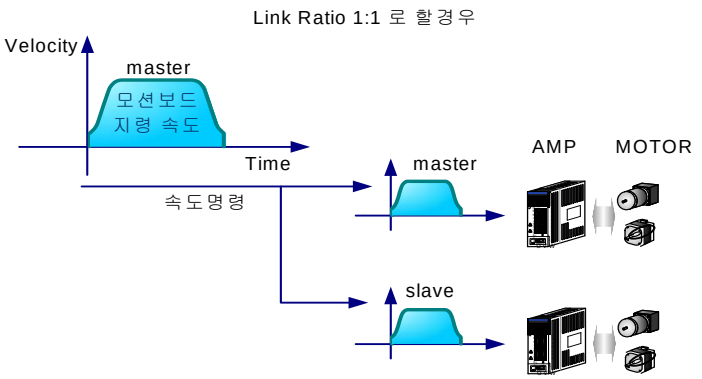
```
begin
{ 0축 50000만큼 100의 속도와 200의 가속율로 구동, 10000 위치에서 500 속도 오버라이
       드. }
AxmOverrideVelAtPos (0, 50000, 100, 200, 200, 10000, 500, ACT);
End;
```

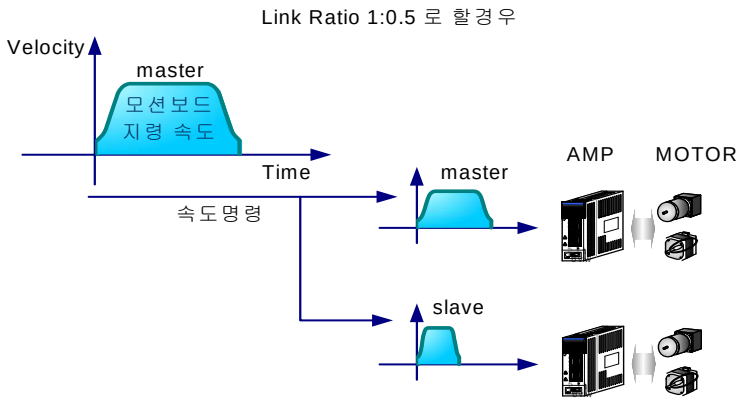
See Also
AxmOverrideSetMaxVel, AxmOverridePos, AxmOverrideVel, AxmOverrideAccelVelDecel

MASTER, SLAVE 구동 함수

Function	Description
AxmLinkSetMode	Master 축과 Slave축간의 전자기어비를 설정한다.
AxmLinkGetMode	Master 축과 Slave축간의 전자기어비를 반환한다.
AxmLinkResetMode	Master 축과 Slave축간의 전자기어비를 설정 해제 한다.

동기구동이란 Electric Gear Mode 라고도 하는데, 독립된 두 개의 축을 하나의 축처럼 제어하는 방식을 말한다. 사용자가 특정 Master축과 Slave축을 설정하고, 두 축간의 구동비를 설정해주면 Master축이 움직일 때 설정된 구동비로 Slave축이 같이 움직이는 구동이다.





AxmLinkSetMode

Purpose

Master 축과 Slave축간의 전자기어비를 설정한다.

Format

C

DWORD AxmLinkSetMode(long lMasterAxisNo, long lSlaveAxisNo, double dSlaveRatio);

Visual Basic

Function AxmLinkSetMode (ByVal lMasterAxisNo As Long, ByVal lSlaveAxisNo As Long, ByVal dSlaveRatio As Double) As Long

Delphi

function AxmLinkSetMode (lMasterAxisNo : LongInt; lSlaveAxisNo : LongInt; dSlaveRatio : Double) : DWord; stdcall;

Input / Output

Name	in/out	Format	Init Value	Explanation
MasterAxisNo	in	long	-	마스터채널(축) 번호(0 ~ (최대축수 - 1))
SlaveAxisNo	in	long	-	슬레이브채널(축) 번호(0 ~ (최대축수 - 1))
SlaveRatio	in	double	-	마스터축(1)에 대한 슬레이브의 기어비(0.0 ~ 1.0)

Return AXT_RT_SUCCESS(0000) : Successful execution of API

* See error code Table more information on status error codes

Description

지정한 축에 대하여 이송명령을 내렸을 때 이송동작의 슬레이브축이 마스터축의 동작 상황에 동기되어 시작되도록 할 때 사용하는 함수다. 사용자가 지정한 Axis에 master축이 움직일 때 slave 축이 같이 구동하게 master, slave 축을 지정한다. ratio의 구동 비는 master 축이 한 바퀴 돌 때 slave 축이 구동할 비를 말한다.(SlaveRatio 0 ~ 100% 0 : 0%, 0.5 : 50%, 1.0 : 100%).

C Example

```
// 마스터축과 슬레이브축의 비율을 0.5로 넣었기 때문에 움직일 때
// 마스터가 1 움직이면 슬레이브는 0.5만큼만 움직인다.
AxmLinkSetMode(0, 1, 0.5);
```

```
// Master 축과 Slave축간의 전자기어비를 반환한다.
double dpGearRatio;
long lpSlaveAxis;
AxmLinkGetMode(0, &lpSlaveAxis, &dpGearRatio);
```

VB Example

```
'마스터축과 슬레이브축의 비율을 0.5로 넣었기 때문에 움직일 때
'마스터가 1 움직이면 슬레이브는 0.5만큼만 움직인다..
AxmLinkSetMode 0, 1, 0.5

'Master 축과 Slave축간의 전자기어비를 반환한다.
Dim dpGearRatio As Double
Dim lpSlaveAxis As Long
AxmLinkGetMode 0, lpSlaveAxis, dpGearRatio
```

Delphi Example

```
{ Master 축과 Slave축간의 전자기어비를 반환한다.}
var
dpGearRatio: Double;
lpSlaveAxis: LongInt;
Begin

{ 마스터축과 슬레이브축의 비율을 0.5로 넣었기 때문에 움직일 때
마스터가 1 움직이면 슬레이브는 0.5만큼만 움직인다.}
AxmLinkSetMode(0, 1, 0.5);

AxmLinkGetMode(0, @lpSlaveAxis, @dpGearRatio);
end;
```

See Also

AxmLinkGetMode, AxmLinkResetMode

AxmLinkGetMode

Purpose

Master 축과 Slave축간의 전자기어비를 반환한다.

Format

C

DWORD AxmLinkGetMode (long lMasterAxisNo, long *lpSlaveAxisNo, double *dpGearRatio);

Visual Basic

Function AxmLinkGetMode (ByVal lMasterAxisNo As Long, ByRef lpSlaveAxisNo As Long, ByRef dpGearRatio As Double) As Long

Delphi

function AxmLinkGetMode (lMasterAxisNo : LongInt; lpSlaveAxisNo : PLongInt; dpGearRatio : PDouble) : DWord; stdcall;

Input / Output

Name	in/out	Format	Init Value	Explanation
MasterAxisNo	in	long	-	마스터채널(축) 번호(0부터 시작함)
SlaveAxisNo	out	long	-	슬레이브채널(축) 번호
SlaveRatio	out	double*	-	마스터축(1)에 대한 슬레이브의 기어비

Return AXT_RT_SUCCESS(0000) : Successful execution of API

* See error code Table more information on status error codes

Description

Master 축과 Slave축간의 전자기어비를 반환한다.
ratio의 구동 비는 master 축이 한 바퀴 돌 때 slave 축이 구동할 비를 말한다.

C Example

```
// 마스터축과 슬레이브축의 비율을 0.5로 넣었기 때문에 움직일 때
// 마스터가 1 움직이면 슬레이브는 0.5만큼만 움직인다.
AxmLinkSetMode(0, 1, 0.5);

// Master 축과 Slave축간의 전자기어비를 반환한다.
double dpGearRatio;
long lpSlaveAxis;
```

```
AxmLinkGetMode(0, &lpSlaveAxis, &dpGearRatio);
```

VB Example

```
'마스터축과 슬레이브축의 비율을 0.5로 넣었기 때문에 움직일 때
'마스터가 1 움직이면 슬레이브는 0.5만큼만 움직인다..
AxmLinkSetMode 0, 1, 0.5

'Master 축과 Slave축간의 전자기어비를 반환한다.
Dim dpGearRatio As Double
Dim lpSlaveAxis As Long
AxmLinkGetMode 0, lpSlaveAxis, dpGearRatio
```

Delphi Example

```
{ Master 축과 Slave축간의 전자기어비를 반환한다.}
var
dpGearRatio: Double;
lpSlaveAxis: LongInt;
Begin

{ 마스터축과 슬레이브축의 비율을 0.5로 넣었기 때문에 움직일 때
마스터가 1 움직이면 슬레이브는 0.5만큼만 움직인다.}
AxmLinkSetMode(0, 1, 0.5);

AxmLinkGetMode(0, @lpSlaveAxis, @dpGearRatio);
end;
```

See Also

AxmLinkSetMode, AxmLinkResetMode

AxmLinkResetMode

Purpose

마스터 축과 슬레이브 축의 전자 기어비를 해제 한다.

Format

C

DWORD AxmLinkResetMode (long lMasterAxisNo);

Visual Basic

Function AxmLinkResetMode (ByVal lMasterAxisNo As Long) As Long

Delphi

function AxmLinkResetMode (lMasterAxisNo : LongInt) : DWord; stdcall;

Input / Output

Name	in/out	Format	Init Value	Explanation
MasterAxisNo	in	long	-	마스터채널(축) 번호(0 ~ (최대축수 - 1))

Return AXT_RT_SUCCESS(0000) : Successful execution of API

* See error code Table more information on status error codes

Description

사용자가 지정한 Axis에 master축이 움직일 때 slave 축이 같이 구동하게 master, slave 축을 지정한 ratio를 해제시킨다. ratio의 구동 비는 master 축이 한 바퀴 돌 때 slave 축이 구동할 비를 말한다.

C Example

```
// Master 축과 Slave축간의 전자기어비를 설정 해제한다.
AxmLinkResetMode (0);
```

VB Example

```
'Master 축과 Slave축간의 전자기어비를 설정 해제한다.
AxmLinkResetMode 0
```

Delphi Example

```
{ Master 축과 Slave축간의 전자기어비를 설정 해제한다.}
AxmLinkResetMode (0);
```

See Also

AxmLinkSetMode, AxmLinkGetMode

보간 구동 함수(Line, Circle)

이 단원에서는 보간(Interpolation) 모션제어에 대해서 설명한다. 보간(Interpolation) 이란 두 축 이상의 축이 연동되어 직선보간, 원호보간 등을 수행하는 것을 의미한다.
여러 축을 동시에 제어한다는 것은 다축 모션제어와 똑같은 기능이라 할 수 있지만 보간(Interpolation)은 사용자가 원하는 경로를 추종하기 위해서 보간 이동에 관련된 각 축의 속도를 자동으로 조절하면서 이동을 한다는 점이 다축 동시모션제어와 가장 큰 차이가 난다.
본 단원에서는 직선보간과 원호보간에 일반보간사용법에 대해서 설명한다.

Function	Description
AxmLineMove	시작점과 종료점을 지정하여 다축 직선 보간 구동하는 함수이다. 구동 시작 후 함수를 벗어난다.
AxmLineArcMove	시작점(x,y,z) , 중간점(x,y,z) , 끝점(x,y,z) , 중간점에서의 거리(d)를 넣고 직선보간을 이용하여 원호보간을 한다.
AxmCircleCenterMove	시작점, 종료점과 중심점을 지정하여 원호 보간 구동하는 함수이다. 구동 시작 후 함수를 벗어난다.
AxmCirclePointMove	중간점, 종료점을 지정하여 원호 보간 구동하는 함수이다. 구동 시작 후 함수를 벗어난다.
AxmCircleRadiusMove	시작점, 종료점과 반지름을 지정하여 원호 보간 구동하는 함수이다. 구동 시작 후 함수를 벗어난다.
AxmCircleAngleMove	시작점, 회전각도와 반지름을 지정하여 원호 보간 구동하는 함수이다. 구동 시작 후 함수를 벗어난다.
AxmContiSetAxisMap	지정된 좌표계에 보간 축 맵핑을 설정한다.
AxmContiGetAxisMap	지정된 좌표계에 보간 축 맵핑을 반환한다.
AxmContiSetAbsRelMode	지정된 좌표계에 보간 축 절대/상대 모드를 설정한다.
AxmContiGetAbsRelMode	지정된 좌표계에 보간 축 절대/상대 모드를 반환한다.
AxmContiIsMotion	보간 구동 중인지 확인 하는 함수이다.

앞에서 언급한 다축 위치구동은 두 축이 동시에 출발해서 각각의 동작을 수행하는 다축구동이었다. 하지만, 두 축이 움직이는데 있어서 상호 연관된 움직임을 통해 직선 또는 원호의 지정된 경로를 따라 이동해야 하는 경우가 발생하는데, 이러한 구동을 위해 직선보간, 원호보간 등을 제공한다. 이러한 보간구동의 속도프로파일 및 절대/상대좌표 구동에 대한 설정은 보간구동되는 축들 중에 가장 빠른 축 번호를 따라 가도록 되어 있으므로 설정에 주의하기 바란다.

▶주의점: 보간하는 축은 Unit/Pulse를 똑같이 맞춘다.

▶주의점: EIP-M2/4Q를 사용하여 8축 이상을 구성 하였을 때
8축 모션이상 모듈은 Axis 0 – 3의 4개의 축이 축그룹 1로 구분되고 Axis4 – 7 4개의 축이 축그룹 2로 구분되고 Axis8 – 11 4개의 축이 축그룹 3으로 ..... 구분된다. 동일 그룹에 속하는 축들간의 직선보간 구동, 원호보간구동을 사용할 수 있지만, 서로 다른 축그룹에 속한 축들간의 직선보간구동과 원호보간구동은 사용할 수 없다.

AxmLineMove

Purpose

지정된 좌표계에 시작점과 종료점을 지정하여 다축 직선 보간 구동하는 함수이다.

Format

C

```
DWORD AxmLineMove (long lCoord, double *dpPos, double dVel, double dAccel, double dDecel);
```

Visual Basic

```
Function AxmLineMove (ByVal lCoord As Long, ByRef dpPos As Double, ByVal dVel As Double,
ByVal dAccel As Double, ByVal dDecel As Double) As Long
```

Delphi

```
function AxmLineMove (lCoord : LongInt; dpPos : PDouble; dVel : Double; dAccel : Double;
dDecel : Double) : DWord; stdcall;
```

Input / Output

Name	in/out	Format	Init Value	Explanation
Coord	in	long	-	좌표계 번호(0부터 시작함)
Pos	in	double	-	위치 배열
Vel	in	double	-	구동 속도값 (+방향: 0 이상값 , -방향: 0 이하값, 구동속도의 단위는 Unit/pulse를 1/1로 한경우에 PPS[Pulses/sec])
Accel	in	double	-	구동 가속도 (가속도의 단위는 Unit/pulse를 1/1로 한경우에 PPS[Pulses/sec^2])값
Decel	in	double	-	구동 감속도 (감속도의 단위는 Unit/pulse를 1/1로 한경우에 PPS[Pulses/sec^2])값

Return AXT_RT_SUCCESS(0000) : Successful execution of API

* See error code Table more information on status error codes

Description

구동 시작 후 함수를 벗어난다. 함수를 수행하기 전에 AxmContiSetAxisMap함수는 보간작업을 수행할 축들을 맵번호로 축맵핑한다. 일반보간제어에 관련된 함수들도 파라미터의 전달이 간편하므로 적용

된다. 따라서 일반보간제어에 관련된 함수들을 사용하기 전에 가장 먼저 이 함수를 사용하여 사용할 축들을 맵핑한다.

▶ 주의사항: AxmContiSetAxisMap함수를 이용하여 축 맵핑을 하며 축은 낮은 순서부터 맵핑을 한다. 원호보간의 경우에 반드시 낮은 축부터 배열에 넣어야 한다.

▶ 주의점: +LIMIT, -LIMIT 신호가 들어와있을경우 구동이 되지않는다.

▶ AxmSStop, AxmEstop 함수에 의해 모션감속정지, 모션급정지할수있다. 인자는 반드시 Coordinate의 마스터축을 넣어서 정지시킨다.

▶ AxmContioInMotion함수에 의해 모션 중인지 확인할 수 있다.

▶ AxmSignalSetInpos함수를 사용하는 경우에는 Inpos 입력신호가 Enable로 설정되어 있다면 Command 펄스 출력이 완료해도 INP 입력이 ON 되기 전까지는 모션이 완료되지 않은 것으로 간주되어 반환 되지 않는다. 이 내용은 AxmSignalSetInpos함수를 참고하기 바란다. **

▶주의점: 보간하는 축은 Unit/Pulse를 똑같이 맞춘다. Unit/Pulse를 똑같이 맞추지 않으면 단위가 틀려 지므로 반드시 똑같이 맞추어 보간제어를 하여야 한다.

▶주의점: 사용하여 8축 이상을 구성 하였을 때
8축 모션이상 모듈은 Axis 0 – 3의 4개의 축이 축그룹 1로 구분되고 Axis4 – 7 4개의 축이 축그룹 2로 구분되고 Axis8 – 11 4개의 축이 축그룹 3으로 구분된다. 동일 그룹에 속하는 축들간의 직선보간구동, 원호보간구동을 사용할 수 있지만, 서로 다른 축그룹에 속한 축들간의 직선보간구동과 원호보간구동은 사용할 수 없다.

▶ 보간 설정 함수 리스트

Function	Description
AxmContiSetAxisMap	지정된 좌표계에 보간 축 맵핑을 설정한다.
AxmContiGetAxisMap	지정된 좌표계에 보간 축 맵핑을 반환한다.
AxmContiSetAbsRelMode	지정된 좌표계에 보간 축 절대/상대 모드를 설정한다.
AxmContiGetAbsRelMode	지정된 좌표계에 보간 축 절대/상대 모드를 반환한다.

일반구동함수 사용예제>

C Example

```
long   lAxis[2];
double  dPos[2];
long   lPosSize = 2;
long lCoordinate = 0;

dPos[0] = 4000;
dPos[1] = 4000;
lAxis[0] = 0;
lAxis[1] = 1;

AxmContiSetAxisMap(lCoordinate, lPosSize, lAxis);
//상대위치 구동
AxmContiSetAbsRelMode(lCoordinate, 1);
// 직선보간 구동
 AxmLineMove(lCoordinate, dPos, 200, 400, 400);
```

VB Example

```
Dim lCoordinate As Long
Dim Distance(0 To 2) As Double
Dim axes(0 To 2) As Long
Dim i As Long

lCoordinate = 0

 For i = 0 To 2
 axes(i) = i
 Next i

 Distance(0) = 4000
 Distance(1) = 4000

AxmContiSetAxisMap lCoordinate, 2, axes(0)
 ‘ 상대위치 구동
AxmContiSetAbsRelMode lCoordinate, 1
‘ 직선보간 구동
 AxmLineMove lCoordinate, Distance(0), 200, 400, 400
```

Delphi Example

```
var
   i : LongInt;
   lCoordinate : LongInt;
   lAxis : array [0..1] of Longint;
   dPos : array [0..1] of Double;
   lPosSize : LongInt;
begin

lCoordinate := 0;
lPosSize := 2;

for i := 0 to 1 do
```

```
begin
     lAxis[i] := i;
end;

dPos[0] := 4000;
dPos[1] := 4000;

AxmContiSetAxisMap(lCoordinate, lPosSize, @lAxis);

{ 상대위치 구동 }
AxmContiSetAbsRelMode(lCoordinate, 1);
{ 직선보간 구동 }
AxmLineMove(lCoordinate, @dPos, 200, 400, 400);
end;
```

See Also

AxmCircleCenterMove, AxmCirclePointMove, AxmCircleRadiusMove, AxmCircleAngleMove, AxmContiSetAxisMap, AxmContiGetAxisMap, AxmContiSetAbsRelMode, AxmContiGetAbsRelMode, AxmContiIsMotion

AxmCircleCenterMove

Purpose

지정된 좌표계에 시작점, 종료점과 중심점을 지정하여 원호 보간 구동하는 함수이다, 구동 시작 후 함수를 벗어난다.

Format

C

```
DWORD AxmCircleCenterMove (long lCoord, long *lAxisNo, double *dCenterPos, double
*dEndPos, double dVel, double dAccel, double dDecel, DWORD uCWDir);
```

Visual Basic

```
Function AxmCircleCenterMove (ByVal lCoord As Long, ByVal lAxisNo As Long, ByVal dCenterPos
As Double, ByVal dEndPos As Double, ByVal dVel As Double, ByVal dAccel As Double, ByVal
dDecel As Double, ByVal uCWDir As Long) As Long
```

Delphi

```
function AxmCircleCenterMove (lCoord : LongInt; lAxisNo : PLongInt; dCenterPos : PDouble;
dEndPos : PDouble; dVel : Double; dAccel : Double; dDecel : Double; uCWDir : DWord) : DWord;
stdcall;
```

Input / Output

Name	in/out	Format	Init Value	Explanation
Coord	in	long	-	좌표계 번호(0 ~ (최대축수 - 1))
AxisNo	in	long*	-	보간에 사용되는 채널(축) 번호(0부터 시작함) 배열
CenterPos	In	double*	-	X 축 원점 위치값, Y 축 원점 위치값 배열
EndPos	in	double*	-	X 축 종료 위치값, Y 축 종료 위치값 배열
Vel	in	double	-	구동 속도값 (+방향: 0 이상값 , -방향: 0 이하값, 구동속도의 단위는 Unit/pulse를 1/1로 한경우에 PPS[Pulses/sec])
Accel	in	double	-	구동 가속도 (가속도의 단위는 Unit/pulse를 1/1로 한경우에 PPS[Pulses/sec^2])값
Decel	in	double	-	구동 감속도 (감속도의 단위는 Unit/pulse를 1/1로 한경우에 PPS[Pulses/sec^2])값

CWDir	in	DWORD	-	원호 방향

uCWDir

#define	Value	Explanation
DIR_CCW	00h	반시계방향
DIR_CW	01h	시계방향

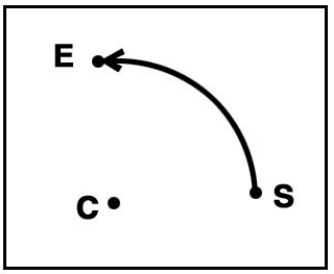
Return AXT_RT_SUCCESS(0000) : Successful execution of API
* See error code Table more information on status error codes

Description

원호보간은 임의의 두 축에 대해서 적용된다. 맵핑 된 두 축을 X,Y 축으로 간주하여 설명한다. 여기서 X축은 맵핑된 두 축 중에서 축번호가 낮은 축을 의미하며 Y축은 축번호가 높은 축을 의미한다. 예를 들어 Z,U축이 맵핑된 두 축이라면 Z축이 X축에 해당되며 U축이 Y축에 해당한다.

함수를 수행하기 전에 AxmContiSetAxisMap함수는 보간작업을 수행할 축들을 맵번호로 축맵핑한다. 일반보간제어에 관련된 함수들은 파라미터의 전달이 간편하므로 적용된다. 따라서 일반보간제어에 관련된 함수들을 사용하기 전에 가장 먼저 이 함수를 사용하여 사용할 축들을 맵핑한다.

현재 위치는 원호 구동의 시작점이 되며 설정한 중점과 종점을 기준으로 원호 구동을 한다.



▶ AxmContiSetAxisMap함수를 이용하여 축 맵핑을 하며 축은 낮은 순서부터 맵핑을 한다. 원호보간의 경우에 반드시 낮은 축부터 배열에 넣어야 한다.

▶ 주의점: +LIMIT, -LIMIT 신호가 들어와있을경우 구동이 되지않는다.

▶ AxmSStop, AxmEstop 함수에 의해 모션감속정지, 모션급정지할수있다. 인자는 반드시 Coordinate의 마스터축을 넣어서 정지시킨다.

▶ AxmStatusReadInMotion 함수에 의해 모션 중인지 확인할 수 있다.

▶ AxmSignalSetInpos함수를 사용하는 경우에는 Inpos 입력신호가 Enable로 설정되어 있다면 Command 펄스 출력이 완료해도 INP 입력이 ON 되기 전까지는 모션이 완료되지 않은 것으로 간주되어 반환 되지 않는다. 이 내용은 AxmSignalSetInpos함수를 참고하기 바란다. **

▶ 주의사항: 보간하는 축은 Unit/Pulse를 똑같이 맞춘다. Unit/Pulse를 똑같이 맞추지 않으면 단위가 틀려 지므로 반드시 똑같이 맞추어 보간제어를 하여야 한다.

▶주의점: 사용하여 8축 이상을 구성 하였을 때
8축 모션이상 모듈은 Axis 0 – 3의 4개의 축이 축그룹 1로 구분되고 Axis4 – 7 4개의 축이 축그룹 2로 구분되고 Axis8 – 11 4개의 축이 축그룹 3으로 구분된다. 동일 그룹에 속하는 축들간의 직선보간구동, 원호보간구동을 사용할 수 있지만, 서로 다른 축그룹에 속한 축들간의 직선보간구동과 원호보간구동은 사용할 수 없다.

▶ 보간 함수 리스트

Function	Description
AxmContiSetAxisMap	지정된 좌표계에 보간 축 맵핑을 설정한다.
AxmContiGetAxisMap	지정된 좌표계에 보간 축 맵핑을 반환한다.
AxmContiSetAbsRelMode	지정된 좌표계에 보간 축 절대/상대 모드를 설정한다.
AxmContiGetAbsRelMode	지정된 좌표계에 보간 축 절대/상대 모드를 반환한다.

<일반보간예제>

C Example

```
long   lAxis[2];
double  dCenPos[2];
double  dEndPos[2];

long   lPosSize = 2;
long lCoordinate = 0;

lAxis[0] = 0;
lAxis[1] = 1;

dCenPos[0] = 500;
dCenPos[1] = 500;
dEndPos[0] = 1000;
dEndPos[1] = 0;
```

```
AxmContiSetAxisMap(lCoordinate, lPosSize, lAxis);
AxmContiSetAbsRelMode(lCoordinate, 1); //상대위치구동

// CW 방향으로 원호 보간구동
AxmCircleCenterMove(lCoordinate, lAxis, dCenPos, dEndPos, 200, 400, 400,
       DIR_CW);
```

VB Example

```
Dim Distance(0 To 2) As Double
Dim CenDistance(0 To 2) As Double
Dim axes(0 To 2) As Long
Dim i As Long
Dim lCoordinate As Long

CenDistance(0) = 500
CenDistance(1) = 500
Distance(0) = 1000
Distance(1) = 0

lCoordinate = 0

For i = 0 To 2
   axes(i) = i
Next i

AxmContiSetAxisMap lCoordinate, 2, axes(0)
AxmContiSetAbsRelMode lCoordinate, 1

' CW 방향으로 원호 보간 구동
AxmCircleCenterMove lCoordinate, axes(0), CenDistance(0), Distance(0),
       200, 400, 400, DIR_CW
```

Delphi Example

```
var
   i : LongInt;
   lCoordinate : LongInt;
   lAxis : array [0..1] of Longint;
   dCenPos : array [0..1] of Double;
   dEndPos : array [0..1] of Double;
   lPosSize : LongInt;
begin

dCenPos[0] := 500;
dCenPos[1] := 500;

dEndPos[0] := 1000;
dEndPos[1] := 0;

lCoordinate := 0;
lPosSize := 2;
```

```
for i := 0 to 1 do
begin
   lAxis[i] := i;
end;

AxmContiSetAxisMap(lCoordinate, lPosSize, @lAxis);
AxmContiSetAbsRelMode(lCoordinate, 1); //상대위치구동

AxmCircleCenterMove(lCoordinate, @lAxis, @dCenPos, @dEndPos, 200, 400,
       400, DIR_CW);

End;
```

See Also

AxmLineMove, AxmCirclePointMove, AxmCircleRadiusMove, AxmCircleAngleMove,AxmContiSetAxisMap, AxmContiGetAxisMap, AxmContiSetAbsRelMode, AxmContiGetAbsRelMode, AxmContiIsMotion

AxmCirclePointMove

Purpose

지정된 좌표계에 중간점, 종료점을 지정하여 원호 보간을 하는 함수이다.

Format

C

```
DWORD AxmCirclePointMove (long lCoord, long *lAxisNo, double *dMidPos, double *dEndPos,
double dVel, double dAccel, double dDecel, long lArcCircle);
```

Visual Basic

```
Function AxmCirclePointMove (ByVal lCoord As Long, ByVal lAxisNo As Long, ByVal dMidPos As
Double, ByVal dEndPos As Double, ByVal dVel As Double, ByVal dAccel As Double, ByVal dDecel
As Double, ByVal lArcCircle As Long) As Long
```

Delphi

```
function AxmCirclePointMove (lCoord : LongInt; lAxisNo : PLongInt; dMidPos : PDouble;
dEndPos : PDouble; dVel : Double; dAccel : Double; dDecel : Double; lArcCircle: LongInt) :
DWord; stdcall;
```

Input / Output

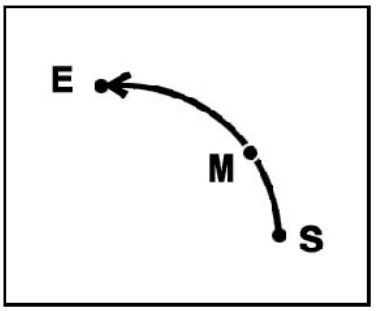
Name	in/out	Format	Init Value	Explanation
Coord	in	long	-	좌표계 번호(0 ~ (최대축수 - 1))
AxisNo	in	long*	-	보간에 사용되는 채널(축) 번호(0부터 시작함) 배열
MidPos	In	double*	-	X 축 중간 위치값, Y 축 중간 위치값 배열
EndPos	in	double*	-	X 축 종료 위치값, Y 축 종료 위치값 배열
Vel	in	double	-	구동 속도값 (+방향: 0 이상값 , -방향: 0 이하값, 구동속도의 단위는 Unit/pulse를 1/1로 한경우에 PPS[Pulses/sec])
Accel	in	double	-	구동 가속도 (가속도의 단위는 Unit/pulse를 1/1로 한경우에 PPS[Pulses/sec^2])값
Decel	in	double	-	구동 감속도 (감속도의 단위는 Unit/pulse를 1/1로 한경우에 PPS[Pulses/sec^2])값

ArcCircle	in	long	-	Arc(0), Cirlcle(1) 선택값

Return AXT_RT_SUCCESS(0000) : Successful execution of API
* See error code Table more information on status error codes

Description

원호보간은 임의의 두 축에 대해서 적용된다. 맵핑된 두 축을 X,Y 축으로 간주하여 설명한다. 여기서 X축은 맵핑된 두 축 중에서 축번호가 낮은 축을 의미하며 Y축은 축번호가 높은 축을 의미한다.
예를 들어 Z,U축이 맵핑된 두 축이라면 Z축이 X축에 해당되며 U축이 Y축에 해당한다.
함수를 수행하기 전에 AxmContiSetAxisMap함수는 보간작업을 수행할 축들을 맵번호로 축맵핑한다. 일반보간제어에 관련된 함수들도 파라미터의 전달이 간편하므로 적용된다. 따라서 일반보간제어에 관련된 함수들을 사용하기 전에 가장 먼저 이 함수를 사용하여 사용할 축들을 맵핑한다.
현재 위치는 원호 구동의 시작점이 되며 중간점을 지나는 원호 구동을 한다.



▶ 주의점: AxmContiSetAxisMap함수를 이용하여 축 맵핑을 하며 축은 낮은 순서부터 맵핑을 한다. 원호보간의 경우에 반드시 낮은 축부터 배열에 넣어야 한다.

▶ 주의점: +LIMIT, -LIMIT 신호가 들어와있을경우 구동이 되지않는다

▶ AxmSStop, AxmEstop 함수에 의해 모션감속정지, 모션급정지할수있다. 인자는 반드시 Coordinate의 마스터축을 넣어서 정지시킨다.

▶ AxmStatusReadInMotion 함수에 의해 모션 중인지 확인할 수 있다.

▶ AxmSignalSetInpos함수를 사용하는 경우에는 Inpos 입력신호가 Enable로 설정되어 있다면 Command 펄스 출력이 완료해도 INP 입력이 ON 되기 전까지는 모션이 완료되지 않은 것으로 간주되어 반환 되지 않는다. 이 내용은 AxmSignalSetInpos함수를 참고하기 바란다. **

▶ 주의사항: 보간하는 축은 Unit/Pulse 를 똑같이 맞춘다. Unit/Pulse를 똑같이 맞추지 않으면 서로

단위가 달라 지므로 똑같이 맞추어 보간제어를 한다.

▶주의점: 8축 이상을 구성 하였을 때
8축 모션이상 모듈은 Axis 0 – 3의 4개의 축이 축그룹 1로 구분되고 Axis4 – 7 4개의 축이 축그룹 2로 구분되고 Axis8 – 11 4개의 축이 축그룹 3으로 구분된다. 동일 그룹에 속하는 축들간의 직선보간구동, 원호보간구동을 사용할 수 있지만, 서로 다른 축그룹에 속한 축들간의 직선보간구동과 원호보간구동은 사용할 수 없다.

▶ 보간 함수 리스트

Function	Description
AxmContiSetAxisMap	지정된 좌표계에 보간 축 맵핑을 설정한다.
AxmContiGetAxisMap	지정된 좌표계에 보간 축 맵핑을 반환한다.
AxmContiSetAbsRelMode	지정된 좌표계에 보간 축 절대/상대 모드를 설정한다.
AxmContiGetAbsRelMode	지정된 좌표계에 보간 축 절대/상대 모드를 반환한다.

<일반 보간함수 예제>

C Example

```
long   lAxis[2];
double  dMidPos[2];
double  dEndPos[2];

long   lPosSize = 2;
long lCoordinate = 0;
double  dVelocity = 200;
double  dAccel = 400;

lAxis[0] = 0;
lAxis[1] = 1;

dMidPos[0] = 500;
dMidPos[1] = 500;

dEndPos[0] = 1000;
dEndPos[1] = 0;

AxmContiSetAxisMap(lCoordinate, lPosSize, lAxis);
AxmContiSetAbsRelMode(lCoordinate, 1);

AxmCirclePointMove(lCoordinate, lAxis, dMidPos, dEndPos, dVelocity,
      dAccel, dAccel, 0);
```

VB Example

```
Dim dVelocity As Double
Dim dAccel As Double
Dim Distance(0 To 2) As Double
Dim MidDistance(0 To 2) As Double
Dim axes(0 To 2) As Long
Dim i As Long
Dim lCoordinate As Long

Distance(0) = 1000
Distance(1) = 0
MidDistance(0) = 500
MidDistance(1) = 500

dVelocity = 200
dAccel = 400
lCoordinate = 0

For i = 0 To 2
   axes(i) = i
Next i

AxmContiSetAxisMap lCoordinate, 2, axes(0)
AxmContiSetAbsRelMode lCoordinate, 1

AxmCirclePointMove lCoordinate, axes(0), MidDistance(0), Distance(0),
       dVelocity, dAccel, dAccel, 0
```

Delphi Example

```
var
   velocity, accel : Double;
   i : LongInt;
   lCoordinate : LongInt;
   lAxis : array [0..1] of Longint;
   dPos : array [0..1] of Double;
   dMidPos : array [0..1] of Double;
   dEndPos : array [0..1] of Double;
   lPosSize : LongInt;
begin

   dMidPos[0] := 500;
   dMidPos[1] := 500;
   dEndPos[0] := 1000;
   dEndPos[1] := 0;

   velocity := 200;
   accel := 400;
```

```
    lCoordinate := 0;
    lPosSize := 2;

    for i := 0 to 1 do
    begin
      lAxis[i]             := i;
    end;

    AxmContiSetAxisMap(lCoordinate, lPosSize, @lAxis);
    AxmContiSetAbsRelMode lCoordinate, 1
    AxmCirclePointMove(lCoordinate, @lAxis, @dMidPos, @dEndPos, velocity,
       accel, accel, 0);
end;
```

See Also

AxmLineMove, AxmCircleCenterMove, AxmCircleRadiusMove, AxmCircleAngleMove, AxmContiSetAxisMap, AxmContiGetAxisMap, AxmContiSetAbsRelMode, AxmContiGetAbsRelMode AxmContiIsMotion

AxmCircleRadiusMove

Purpose

지정된 좌표계에 시작점, 종료점과 반지름을 지정하여 원호 보간 구동하는 함수이다. 구동 시작 후 함수를 벗어난다.

Format

C

```
DWORD AxmCircleRadiusMove (long lCoord, long *lAxisNo, double dRadius, double *dEndPos,
double dVel, double dAccel, double dDecel, DWORD uCWDir, DWORD uShortDistance);
```

Visual Basic

```
Function AxmCircleRadiusMove (ByVal lCoord  As Long, ByVal lAxisNo As Long, ByVal dRadius As
Double, ByVal dEndPos As Double, ByVal dVel As Double, ByVal dAccel As Double, ByVal dDecel
As Double, ByVal uCWDir As Long, ByVal uShortDistance As Long) As Long
```

Delphi

```
function AxmCircleRadiusMove (lCoord : LongInt; lAxisNo : PLongInt; dRadius : Double; dEndPos :
PDouble; dVel : Double; dAccel : Double; dDecel : Double; uCWDir : DWord; uShortDistance :
DWord) : DWord; stdcall;
```

Input / Output

Name	in/out	Format	Init Value	Explanation
Coord	in	long	-	좌표계 번호(0 ~ (최대축수 - 1))
AxisNo	in	long*	-	보간에 사용되는 채널(축) 번호(0부터 시작함) 배열
Radius	In	double	-	구동할 원호 반지름
EndPos	in	double*	-	X 축 종료 위치값, Y 축 종료 위치값 배열
Vel	in	double	-	구동 속도값 (+방향: 0 이상값 , -방향: 0 이하값, 구동속도의 단위는 Unit/pulse를 1/1로 한경우에 PPS[Pulses/sec])
Accel	in	double	-	구동 가속도 (가속도의 단위는 Unit/pulse를 1/1로 한경우에 PPS[Pulses/sec^2]) 값
Decel	in	double	-	구동 감속도 (감속도의 단위는 Unit/pulse

				를 1/1로 한경우에 PPS[Pulses/sec^2]) 값
CWDir	in	DWORD	-	원호 방향
ShortDistance	in	DWORD	-	중점까지 가는 원의 이동 거리 크기 설정값

CWDir

#define	Value	Explanation
DIR_CCW	00h	반시계방향
DIR_CW	01h	시계방향

ShortDistance

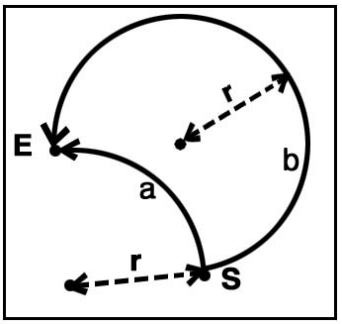
#define	Value	Explanation
SHORT_DISTANCE	00h	짧은 거리의 원호 이동 <그림에서 a>
LONG_DISTANCE	01h	긴 거리의 원호 이동 <그림에서 b>

Return AXT_RT_SUCCESS(0000) : Successful execution of API
* See error code Table more information on status error codes

Description

원호보간은 임의의 두 축에 대해서 적용된다. 맵핑된 두 축을 X,Y 축으로 간주하여 설명한다. 여기서 X축은 맵핑된 두 축중에서 축번호가 낮은 축을 의미하며 Y축은 축번호가 높은 축을 의미한다.
예를 들어 Z,U축이 맵핑된 두축이라면 Z축이 X축에 해당되며 U축이 Y축에 해당한다.
함수를 수행하기전에 AxmContiSetAxisMap함수는 보간작업을 수행할 축들을 맵번호로 축맵핑한다.
일반보간제어에 관련된 함수들도 파라미터의 전달이 간편하므로 적용된다. 따라서 일반보간제어에 관련된 함수들을 사용하기전에 가장 먼저 이 함수를 사용하여 사용할 축들을 맵핑한다..

현재 위치는 원호 구동의 시작점이 되며 설정한 반지름과 종점을 기준으로 원호 구동을 한다.
시작점과 끝점을 연결하는 특정 반지름을 가진 원호를 정의하게 되면, 두개의 중심점이 나타나는데, 이로인해 같은 회전방향이라도 두개의 경로(a,b)가 정의될 수 있다.



▶ 주의점: AxmContiSetAxisMap함수를 이용하여 축 맵핑을 하며 축은 낮은 순서부터 맵핑을 한다. 원호보간의 경우에 반드시 낮은 축부터 배열에 넣어야 한다.

▶주의점: +LIMIT, -LIMIT 신호가 들어와있을경우 구동이 되지않는다.

▶ AxmSStop, AxmEstop 함수에 의해 모션감속정지, 모션급정지할수있다. 인자는 반드시 Coordinate의 마스터축을 넣어서 정지시킨다.

▶ AxmStatusReadInMotion 함수에 의해 모션중인지 확인할 수 있다.

▶ AxmSignalSetInpos함수를 사용하는경우에는 Inpos 입력신호가 Enable로 설정되어 있다면 Command 펄스 출력이 완료해도 INP 입력이 ON 되기 전까지는 모션이 완료되지 않은것으로 간주되어 반환 되지 않는다. 이 내용은 AxmSignalSetInpos함수를 참고하기 바란다.

▶ 주의사항: 보간하는 축은 Unit/Pulse 를 똑같이 맞춘다. Unit/Pulse를 똑같이 맞추지 않으면 서로 단위가 달라 지므로 똑같이 맞추어 보간제어를 한다.

▶ 주의점: 8축 이상을 구성 하였들 때
8축 모션이상 모듈은 Axis 0 – 3의 4개의 축이 축그룹 1로 구분되고 Axis4 – 7 4개의 축이 축그룹 2로 구분되고 Axis8 – 11 4개의 축이 축그룹 3으로 구분된다. 동일 그룹에 속하는 축들간의 직선보간구동, 원호보간구동을 사용할 수 있지만, 서로 다른 축그룹에 속한 축들간의 직선보간구동과 원호보간구동은 사용할 수 없다.

▶ 보간 함수 리스트

Function	Description
AxmContiSetAxisMap	지정된 좌표계에 보간 축 맵핑을 설정한다.
AxmContiGetAxisMap	지정된 좌표계에 보간 축 맵핑을 반환한다.

AxmContiSetAbsRelMode	지정된 좌표계에 보간 축 절대/상대 모드를 설정한다.
AxmContiGetAbsRelMode	지정된 좌표계에 보간 축 절대/상대 모드를 반환한다.

일반보간 구동예제>

C Example

```
long    lAxis[2];
double  dEndPos[2];
DWORD   uShortDistance=0;
long    lPosSize = 2;
long    lCoordinate = 0;
double  dRadius = 500;
double  dVelocity = 200;
double  dAccel = 400;

lAxis[0] = 0;
lAxis[1] = 1;

dEndPos[0] = 1000;
dEndPos[1] = 0;

AxmContiSetAxisMap(lCoordinate, lPosSize, lAxis);
AxmContiSetAbsRelMode(lCoordinate, 1);

AxmCircleRadiusMove(lCoordinate, lAxis, dRadius, dEndPos, dVelocity,
      dAccel, dAccel, 1, uShortDistance);
```

VB Example

```
Dim dVelocity As Double
Dim dAccel As Double
Dim dRadius As Double
Dim Distance(0 To 2) As Double
Dim axes(0 To 2) As Long
Dim i As Long
Dim lCoordinate As Long
Dim uShortDistance As Long

uShortDistance = 0
Distance(0) = 1000
Distance(1) = 0
dRadius = 500
dVelocity = 200
dAccel = 400
lCoordinate = 0

For i = 0 To 2
   axes(i) = i
```

```
Next i

AxmContiSetAxisMap lCoordinate, 2, axes(0)
AxmContiSetAbsRelMode lCoordinate, 1

AxmCircleRadiusMove lCoordinate, axes(0), dRadius, Distance (0),
       dVelocity, dAccel, dAccel, 1, uShortDistance
```

Delphi Example

```
var
   xend, yend : Double;
   velocity, accel : Double;
   dRadius : Double;
   i : LongInt;
   lCoordinate : LongInt;
   lAxis : array [0..1] of Longint;
   dPos : array [0..1] of Double;
   dEndPos : array [0..1] of Double;
   lPosSize : LongInt;
    uShortDistance : DWord;
begin

dEndPos[0] := 1000;
dEndPos[1] := 0;
dRadius := 500;

velocity := 200;
accel := 400;
lCoordinate := 0;
lPosSize := 2;
uShortDistance := 0;

 for i := 0 to 1 do
 begin
    lAxis[i]               := i;
 end;

AxmContiSetAxisMap(lCoordinate, lPosSize, @lAxis);
AxmContiSetAbsRelMode lCoordinate, 1

AxmCircleRadiusMove(lCoordinate, @lAxis, dRadius, @dEndPos, velocity,
     accel, accel, 1, uShortDistance);
end;
```

See Also

AxmLineMove, AxmCircleCenterMove, AxmCirclePointMove, AxmCircleAngleMove, AxmContiSetAxisMap, AxmContiGetAxisMap, AxmContiSetAbsRelMode, AxmContiGetAbsRelMode, AxmContiIsMotion

AxmCircleAngleMove

Purpose

지정된 좌표계에 시작점, 회전각도와 반지름을 지정하여 원호 보간 구동하는 함수이다. 구동 시작 후 함수를 벗어난다.

Format

C

```
DWORD AxmCircleAngleMove(long lCoord, long *lAxisNo, double *dCenterPos, double dAngle,
double dVel, double dAccel, double dDecel, DWORD uCWDir);
```

Visual Basic

```
Function AxmCircleAngleMove (ByVal lCoord As Long, ByVal lAxisNo As Long, ByVal dCenterPos
As Double, ByVal dAngle As Double, ByVal dVel As Double, ByVal dAccel As Double, ByVal dDecel
As Double, ByVal uCWDir As Long) As Long
```

Delphi

```
function AxmCircleAngleMove (lCoord : LongInt; lAxisNo : PLongInt; dCenterPos: PDouble;
dAngle : Double; dVel : Double; dAccel : Double; dDecel : Double; uCWDir : DWord) : DWord;
stdcall;
```

Input / Output

Name	in/out	Format	Init Value	Explanation
Coord	in	long	-	좌표계 번호(0 ~ (최대축수 - 1))
AxisNo	in	long*	-	보간에 사용되는 채널(축) 번호(0부터 시작함) 배열
CenterPos	In	double*	-	X 축 중점 위치값, Y 축 중점 위치값 배열
Angle	in	double	-	구동할 원호 각도값. 단위는 'Degree(도)
Vel	in	double	-	구동 속도값 (+방향: 0 이상값 , -방향: 0 이하값, 구동속도의 단위는 Unit/pulse를 1/1로 한경우에 PPS[Pulses/sec])
Accel	in	double	-	구동 가속도 (가속도의 단위는 Unit/pulse를 1/1로 한경우에 PPS[Pulses/sec^2])값
Decel	in	double	-	구동 감속도 (감속도의 단위는 Unit/pulse를 1/1로 한경우에 PPS[Pulses/sec^2])값

CWDir	in	dWORD	-	원호 방향

CWDir

#define	Value	Explanation
DIR_CCW	00h	반시계방향
DIR_CW	01h	시계방향

Return AXT_RT_SUCCESS(0000) : Successful execution of API

* See error code Table more information on status error codes

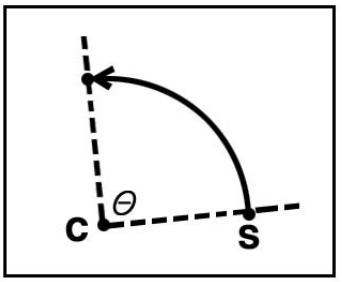
Description

원호보간은 임의의 두 축에 대해서 적용된다. 맵핑된 두축을 X,Y 축으로 간주하여 설명한다. 여기서 X축은 맵핑된 두축중에서 축번호가 낮은 축을 의미하며 Y축은 축번호가 높은 축을 의미한다.

예를 들어 Z,U축이 맵핑된 두축이라면 Z축이 X축에 해당되며 U축이 Y축에 해당한다.

함수를 수행하기 전에 AxmContiSetAxisMap함수는 보간작업을 수행할 축들을 맵번호로 축맵핑한다. 일반보간제어에 관련된 함수들도 파라미터의 전달이 간편하므로 적용된다. 따라서 일반보간제어에 관련된 함수들을 사용하기 전에 가장 먼저 이 함수를 사용하여 사용할 축들을 맵핑한다..

현재 위치는 원호 구동의 시작점이 되며 종점을 기준으로 설정한 각도만큼 원호 구동을 한다.



▶ 주의점: AxmContiSetAxisMap함수를 이용하여 축 맵핑을 하며 축은 낮은 순서부터 맵핑을 한다. 원호보간의 경우에 반드시 낮은 축부터 배열에 넣어야 한다.

▶ 주의점: +LIMIT, -LIMIT 신호가 들어와있을경우 구동이 되지않는다.

▶ AxmSStop, AxmEstop 함수에 의해 모션감속정지, 모션급정지할수있다. 인자는 반드시 Coordinate의 마스터축을 넣어서 정지시킨다.

▶ AxmStatusReadInMotion 함수에 의해 모션중인지 확인할 수 있다.

▶ AxmSignalSetInpos함수를 사용하는 경우에는 Inpos 입력신호가 Enable로 설정되어 있다면 Command 펄스 출력이 완료해도 INP 입력이 ON 되기 전까지는 모션이 완료되지 않은 것으로 간주되어 반환 되지 않는다. 이 내용은 AxmSignalSetInpos함수를 참고하기 바란다. **

▶ 주의사항: 보간하는 축은 Unit/Pulse 를 똑같이 맞춘다. Unit/Pulse를 똑같이 맞추지 않으면 서로 단위가 달라 지므로 똑같이 맞추어 보간제어를 한다.

▶주의점: 사용하여 8축 이상을 구성 하였들 때
8축 모션이상 모듈은 Axis 0 – 3의 4개의 축이 축그룹 1로 구분되고 Axis4 – 7 4개의 축이 축그룹 2로 구분되고 Axis8 – 11 4개의 축이 축그룹 3으로 구분된다. 동일 그룹에 속하는 축들간의 직선보간구동, 원호보간구동을 사용할 수 있지만, 서로 다른 축그룹에 속한 축들간의 직선보간구동과 원호보간구동은 사용할 수 없다.

▶ 보간 함수 리스트

Function	Description
AxmContiSetAxisMap	지정된 좌표계에 보간 축 맵핑을 설정한다.
AxmContiGetAxisMap	지정된 좌표계에 보간 축 맵핑을 반환한다.
AxmContiSetAbsRelMode	지정된 좌표계에 보간 축 절대/상대 모드를 설정한다.
AxmContiGetAbsRelMode	지정된 좌표계에 보간 축 절대/상대 모드를 반환한다.

C Example

```
long    lAxis[2];
double  dCenPos[2];
double dAngle=90;
DWORD  uCWDir=0;
double  dVelocity = 200;
double  dAccel = 400;
long    lPosSize = 2;
long lCoordinate = 0;

lAxis[0] = 0;
lAxis[1] = 1;

dCenPos[0] = 500;
dCenPos[1] = 500;

AxmContiSetAxisMap(lCoordinate, lPosSize, lAxis);
AxmContiSetAbsRelMode(lCoordinate, 1);
```

```
AxmCircleAngleMove(lCoordinate, lAxis, dCenPos, dAngle, dVelocity, dAccel,
       dAccel, 1);
```

VB Example

```
Dim dVelocity As Double
Dim dAccel As Double
Dim dAngle As Double
Dim CenDistance(0 To 2) As Double
Dim axes(0 To 2) As Long
Dim i As Long
Dim lCoordinate As Long

CenDistance(0) = 500
CenDistance(0) = 500
dAngle = 90
dVelocity = 200
dAccel = 400
lCoordinate = 0

For i = 0 To 2
   axes(i) = i
Next i

AxmContiSetAxisMap lCoordinate, 2, axes(0)
AxmContiSetAbsRelMode lCoordinate, 1
AxmCircleAngleMove lCoordinate, axes(0), CenDistance(0), dAngle,
       dVelocity, dAccel, dAccel, 1
```

Delphi Example

```
var
   i : LongInt;
   lCoordinate : LongInt;
   lAxis : array [0..1] of Longint;
   dAngle : Double;
   dCenPos : array [0..1] of Double;
   lPosSize : LongInt;
   dVelocity, dAccel : Double;
begin

dAngle := 90;
dVelocity := 200;
dAccel := 400;
lCoordinate := 0;
lPosSize := 2;

for i := 0 to 1 do
begin
     lAxis[i] := i;
end;
```

```
dCenPos[0] := 500;
dCenPos[1] := 500;

AxmContiSetAxisMap(lCoordinate, lPosSize, @lAxis);
AxmContiSetAbsRelMode(lCoordinate, 1);
AxmCircleAngleMove(lCoordinate, @lAxis, @dCenPos, dAngle, dVelocity,
       dAccel, dAccel, 1);
end;
```

See Also

AxmLineMove, AxmCircleCenterMove, AxmCirclePointMove, AxmCircleRadiusMove,AxmContiSetAxisMap, AxmContiGetAxisMap, AxmContiSetAbsRelMode, AxmContiGetAbsRelMode, AxmContiIsMotion

AxmContiSetAxisMap

Purpose

보간작업을 수행할 축들을 사용자가 지정한 좌표계로 축맵핑 설정한다.

Format

C

```
DWORD AxmContiSetAxisMap (long lCoord, long lSize, long *lpRealAxisNo);
```

Visual Basic

```
Function AxmContiSetAxisMap (ByVal lCoord As Long, ByVal lSize As Long, ByRef lpRealAxisNo
As Long) As Long
```

Delphi

```
function AxmContiSetAxisMap (lCoord : LongInt; lSize : LongInt; lpRealAxisNo : PLongInt) : DWord;
stdcall;
```

Input / Output

Name	in/out	Format	Init Value	Explanation
Coord	in	long	-	좌표계 번호(0 ~ (최대축수 - 1))
lSize	in	long	-	보간에 사용되는 축 수 (2 - 4)
RealAxisNo	In	long*	-	보간에 사용되는 축 배열 2 - 4개축을 설정한다. 설정해야될 축번호(0,1,2,3) 단 한칩내에서 설정가능, (0,1,2,3) 설정 가능, (4,5,6,7) 설정 가능

lSize

#define	Value	Explanation
INTERPOLATION_AXIS2	02h	2축을 보간으로 사용할 때
INTERPOLATION_AXIS3	03h	3축을 보간으로 사용할 때
INTERPOLATION_AXIS4	04h	4축을 보간으로 사용할 때

Return AXT_RT_SUCCESS(0000) : Successful execution of API

* See error code Table more information on status error codes

Description

일반보간제어에 관련된 함수들은 파라미터의 전달이 간편하므로 적용된다. 따라서 일반보간제어에 관련된 함수들을 사용하기 전에 가장 먼저 이 함수를 사용하여 사용할 축들을 맵핑한다.

주의점: 축맵핑할때는 반드시 작은 숫자부터 큰숫자를 넣는다.
여기서 제일 작은숫자 축이 마스터가 된다.

▶주의사항: AxmContiSetAxisMap함수를 이용하여 축맵핑후에 낮은순서축부터 맵핑을 하면서 사용해야된다. 원호보간의 경우에는 반드시 낮은순서축부터 축배열에 넣어야 동작 가능하다.

▶주의점: 사용하여 8축 이상을 구성 하였을 때
8축 모션이상 보드는 Axis 0 – 3의 4개의 축이 축그룹 1로 구분되고 Axis4 – 7 4개의 축이 축그룹 2로 구분되고 Axis8 – 11 4개의 축이 축그룹 3으로 구분된다. 동일 그룹에 속하는 축들간의 직선보간구동, 원호보간구동을 사용할 수 있지만, 서로 다른 축그룹에 속한 축들간의 직선보간구동과 원호보간구동은 사용할 수 없다.

C Example

```
long   lAxis[2];
double  dPos[2];
long   lPosSize = 2;
long lCoordinate = 0;

dPos[0] = 4000;
dPos[1] = 4000;
lAxis[0] = 0;
lAxis[1] = 1;

AxmContiSetAxisMap(lCoordinate, lPosSize, lAxis);
//상대위치 구동
AxmContiSetAbsRelMode(lCoordinate, 1);
// 직선보간 구동
AxmLineMove(lCoordinate, dPos, 200, 400, 400);

long  lAxis1, lAxis2;
AxmContiGetAxisMap(lCoordinate, & lPosSize, lAxis);
// lPosSize 를 이용해서 축수를 확인할 수 있다.
lAxis1 = lAxis[0]; lAxis2 = lAxis[1]; // 맵핑축번호를 가져온다.
```

VB Example

```
Dim lCoordinate As Long
Dim Distance(0 To 2) As Double
Dim axes(0 To 2) As Long
Dim lAxis1 As Long
Dim lAxis2 As Long
Dim i As Long
```

```
lCoordinate = 0

 For i = 0 To 2
 axes(i) = i
 Next i

 Distance(0) = 4000
 Distance(1) = 4000

AxmContiSetAxisMap lCoordinate, 2, axes(0)
 ‘ 상대위치 구동
AxmContiSetAbsRelMode lCoordinate, 1
‘ 직선보간 구동
AxmLineMove lCoordinate, Distance(0), 200, 400, 400

AxmContiGetAxisMap lCoordinate, 2, axes (0)
‘ lPos를 이용해서 축수를 확인할 수 있다.
lAxis1 = axes (0): lAxis2 = axes (1)  ‘ 맵핑축번호를 가져온다.
```

Delphi Example

```
var
   i : LongInt;
   lCoordinate : LongInt;
   lAxis : array [0..1] of Longint;
   dPos : array [0..1] of Double;
   lPosSize : LongInt;
   lAxis1 : LongInt;
   lAxis2 : LongInt;
begin

lCoordinate := 0;
lPosSize := 2;

for i := 0 to 1 do
begin
     lAxis[i] := i;
end;

dPos[0] := 4000;
dPos[1] := 4000;
AxmContiSetAxisMap(lCoordinate, lPosSize, @lAxis);

{ 상대위치 구동 }
AxmContiSetAbsRelMode(lCoordinate, 1);

{ 직선보간 구동 }
AxmLineMove(lCoordinate, @dPos, 200, 400, 400);

AxmContiGetAxisMap(lCoordinate, @lPos, lAxis);
{ lPos를 이용해서 축수를 확인할 수 있다.}
lAxis1 := lAxis[0]; lAxis2 := lAxis[1]; // 맵핑축번호를 가져온다.
```

```
end;
```

See Also

AxmLineMove, AxmLineArcMove, AxmCircleCenterMove, AxmCirclePointMove, AxmCircleRadiusMove, AxmCircleAngleMove, AxmContiGetAxisMap, AxmContiSetAbsRelMode, AxmContiGetAbsRelMode, AxmContiIsMotion, AxmContiIsMotion

AxmContiGetAxisMap

Purpose

사용자가 지정한 좌표계로 축맵핑을 반환한다.

Format

C

```
DWORD AxmContiGetAxisMap (long lCoord, long *lpSize, long *lpRealAxesNo);
```

Visual Basic

```
Function AxmContiGetAxisMap (ByVal lCoord As Long, ByRef lpSize As Long, ByRef lpRealAxisNo
As Long) As Long
```

Delphi

```
function AxmContiGetAxisMap (lCoord : LongInt; lpSize : PLongInt; lpRealAxisNo : PLongInt) :
DWord; stdcall;
```

Input / Output

Name	in/out	Format	Init Value	Explanation
Coord	in	long	-	좌표계 번호(0부터 시작함)
lSize	in	long*	-	보간에 사용되는 축 수 (2 - 4)
RealAxisNo	out	long*	-	보간에 사용되는 축 배열의 주소값 2 - 4개축을 설정한다. 설정해야될 축번호(0,1,2,3) 단 한칩내에서 설정가능, (0,1,2,3) 설정 가능 (4,5,6,7) 설정 가능

lSize

#define	Value	Explanation
INTERPOLATION_AXIS2	02h	2축을 보간으로 사용할 때
INTERPOLATION_AXIS3	03h	3축을 보간으로 사용할 때
INTERPOLATION_AXIS4	04h	4축을 보간으로 사용할 때

Return AXT_RT_SUCCESS(0000) : Successful execution of API

* See error code Table more information on status error codes

Description

‘ AxmContiSetAxisMap ‘로 설정한 보간작업의 축 맵핑을 사용자가 지정한 좌표계의 축맵핑으로 반환한다.

C Example

```
long   lAxis[2];
double  dPos[2];
long   lPosSize = 2;
long lCoordinate = 0;

dPos[0] = 4000;
dPos[1] = 4000;
lAxis[0] = 0;
lAxis[1] = 1;

AxmContiSetAxisMap(lCoordinate, lPosSize, lAxis);
//상대위치 구동
AxmContiSetAbsRelMode(lCoordinate, 1);
// 직선보간 구동
 AxmLineMove(lCoordinate, dPos, 200, 400, 400);

long  lAxis1, lAxis2;
AxmContiGetAxisMap(lCoordinate, & lPosSize, lAxis);
// lPosSize 를 이용해서 축수를 확인할 수 있다.
lAxis1 = lAxis[0]; lAxis2 = lAxis[1]; // 맵핑축번호를 가져온다.
```

VB Example

```
Dim lCoordinate As Long
Dim Distance(0 To 2) As Double
Dim axes(0 To 2) As Long
Dim lAxis1 As Long
Dim lAxis2 As Long
Dim i As Long

lCoordinate = 0

 For i = 0 To 2
 axes(i) = i
 Next i

 Distance(0) = 4000
 Distance(1) = 4000

AxmContiSetAxisMap lCoordinate, 2, axes(0)
 ‘ 상대위치 구동
AxmContiSetAbsRelMode lCoordinate, 1
‘ 직선보간 구동
```

```
AxmLineMove lCoordinate, Distance(0), 200, 400, 400

AxmContiGetAxisMap lCoordinate, 2, axes (0)
‘ lPos를 이용해서 축수를 확인할 수 있다.
lAxis1 = axes (0): lAxis2 = axes (1)  ‘ 맵핑축번호를 가져온다.
```

Delphi Example

```
var
   i : LongInt;
   lCoordinate : LongInt;
   lAxis : array [0..1] of Longint;
   dPos : array [0..1] of Double;
   lPosSize : LongInt;
   lAxis1 : LongInt;
   lAxis2 : LongInt;
begin

lCoordinate := 0;
lPosSize := 2;

for i := 0 to 1 do
begin
     lAxis[i] := i;
end;

dPos[0] := 4000;
dPos[1] := 4000;

AxmContiSetAxisMap(lCoordinate, lPosSize, @lAxis);
{ 상대위치 구동 }
AxmContiSetAbsRelMode(lCoordinate, 1);

{ 직선보간 구동 }
AxmLineMove(lCoordinate, @dPos, 200, 400, 400);

AxmContiGetAxisMap(lCoordinate, @lPos, lAxis);
{ lPos를 이용해서 축수를 확인할 수 있다.}
lAxis1 := lAxis[0]; lAxis2 := lAxis[1]; // 맵핑축번호를 가져온다.
end;
```

See Also

AxmLineMove, AxmLineArcMove, AxmCircleCenterMove, AxmCirclePointMove, AxmCircleRadiusMove, AxmCircleAngleMove, AxmContiSetAxisMap, AxmContiSetAbsRelMode, AxmContiGetAbsRelMode, AxmContiIsMotion, AxmContiIsMotion

AxmContiSetAbsRelMode

Purpose

지정 좌표계의 이동 거리 계산 모드를 설정한다.

Format

C

DWORD AxmContiSetAbsRelMode (long lCoord, DWORD uAbsRelMode);

Visual Basic

Function AxmContiSetAbsRelMode (ByVal lCoord As Long, ByVal uAbsRelMode As Long) As Long

Delphi

function AxmContiSetAbsRelMode (lCoord : LongInt; uAbsRelMode : DWord) : DWord; stdcall;

Input / Output

Name	in/out	Format	Init Value	Explanation
Coord	in	long	-	좌표계 번호(0 ~ (최대축수 - 1))
AbsRelMode	In	DWORD	-	이동거리 계산 모드

uAbsRelMode

#define	Value	Explanation
POS_ABS_MODE	00h	위치구동 절대모드
POS_REL_MODE	01h	위치구동 상대모드

Return AXT_RT_SUCCESS(0000) : Successful execution of API

* See error code Table more information on status error codes

Description

좌표계 내의 모든 축들을 똑 같은 이동 거리 계산 모드를 설정한다.

C Example

```
long    lAxis[2];
double  dPos[2];
long    lPosSize = 2;
long lCoordinate = 0;
```

```
dPos[0] = 4000;
dPos[1] = 4000;
lAxis[0] = 0;
lAxis[1] = 1;

AxmContiSetAxisMap(lCoordinate, lPosSize, lAxis);
//상대위치 구동
AxmContiSetAbsRelMode(lCoordinate, 1);
// 직선보간 구동
 AxmLineMove(lCoordinate, dPos, 200, 400, 400);

// 0축의 이동 모드 확인.
DWORD Mode;
AxmContiGetAbsRelMode (lCoordinate, &Mode);
```

VB Example

```
Dim lCoordinate As Long
Dim Distance(0 To 2) As Double
Dim axes(0 To 2) As Long
Dim lAxis1 As Long
Dim lAxis2 As Long
Dim i As Long

lCoordinate = 0

 For i = 0 To 2
 axes(i) = i
 Next i

 Distance(0) = 4000
 Distance(1) = 4000

AxmContiSetAxisMap lCoordinate, 2, axes(0)
 ‘ 상대위치 구동
AxmContiSetAbsRelMode lCoordinate, 1
‘ 직선보간 구동
 AxmLineMove lCoordinate, Distance(0), 200, 400, 400

‘ 0축의 이동 모드 확인
Dim Mode As Long
AxmContiGetAbsRelMode lCoordinate, Mode
```

Delphi Example

```
var
   i : LongInt;
   lCoordinate : LongInt;
   lAxis : array [0..1] of Longint;
   dPos : array [0..1] of Double;
   lPosSize : LongInt;
   Mode : LongInt;
begin
```

```
lCoordinate := 0;
lPosSize := 2;

for i := 0 to 1 do
 begin
     lAxis[i] := i;
end;

dPos[0] := 4000;
dPos[1] := 4000;

AxmContiSetAxisMap(lCoordinate, lPosSize, @lAxis);

{ 상대위치 구동 }
 AxmContiSetAbsRelMode(lCoordinate, 1);

{ 직선보간 구동 }
AxmLineMove(lCoordinate, @dPos, 200, 400, 400);

{ 0축의 이동 모드 확인 }
AxmContiGetAbsRelMode (lCoordinate, @Mode);
end;
```

See Also

AxmLineMove, AxmLineArcMove, AxmCircleCenterMove, AxmCirclePointMove, AxmCircleRadiusMove, AxmCircleAngleMove, AxmContiSetAxisMap, AxmContiGetAxisMap, AxmContiGetAbsRelMode, AxmContiIsMotion

AxmContiGetAbsRelMode

Purpose

지정 좌표계의 이동 거리 계산 모드를 반환한다.

Format

C

DWORD AxmContiGetAbsRelMode (long lCoord, DWORD *upAbsRelMode);

Visual Basic

Function AxmContiGetAbsRelMode (ByVal lCoord As Long, ByRef upAbsRelMode As Long) As Long

Delphi

function AxmContiGetAbsRelMode (lCoord : LongInt; upAbsRelMode : PDWord) : DWord; stdcall;

Input / Output

Name	in/out	Format	Init Value	Explanation
Coord	in	long	-	좌표계 번호(0 ~ (최대축수 - 1))
AbsRelMode	out	DWORD*	-	이동거리 계산 모드

uAbsRelMode

#define	Value	Explanation
POS_ABS_MODE	00h	위치구동 절대모드
POS_REL_MODE	01h	위치구동 상대모드

Return AXT_RT_SUCCESS(0000) : Successful execution of API

* See error code Table more information on status error codes

Description

'AxmContiSetAbsRelMode' 로 설정한 지정 좌표계의 이동 거리 계산 모드를 반환한다.

C Example

```
long   lAxis[2];
double  dPos[2];
long   lPosSize = 2;
long lCoordinate = 0;
```

```
dPos[0] = 4000;
dPos[1] = 4000;
lAxis[0] = 0;
lAxis[1] = 1;

AxmContiWriteClear(lCoordinate);
AxmContiSetAxisMap(lCoordinate, lPosSize, lAxis);
//상대위치 구동
AxmContiSetAbsRelMode(lCoordinate, 1);
// 직선보간 구동
 AxmLineMove(lCoordinate, dPos, 200, 400, 400);

// 0축의 이동 모드 확인.
DWORD Mode;
AxmContiGetAbsRelMode (lCoordinate, &Mode);
```

VB Example

```
Dim lCoordinate As Long
Dim Distance(0 To 2) As Double
Dim axes(0 To 2) As Long
Dim lAxis1 As Long
Dim lAxis2 As Long
Dim i As Long

lCoordinate = 0

 For i = 0 To 2
 axes(i) = i
 Next i

 Distance(0) = 4000
 Distance(1) = 4000

AxmContiWriteClear lCoordinate
AxmContiSetAxisMap lCoordinate, 2, axes(0)
 ‘ 상대위치 구동
AxmContiSetAbsRelMode lCoordinate, 1
‘ 직선보간 구동
 AxmLineMove lCoordinate, Distance(0), 200, 400, 400

‘ 0축의 이동 모드 확인
Dim Mode As Long
AxmContiGetAbsRelMode lCoordinate, Mode
```

Delphi Example

```
var
   i : LongInt;
   lCoordinate : LongInt;
   lAxis : array [0..1] of Longint;
   dPos : array [0..1] of Double;
```

```
   lPosSize : LongInt;
   Mode : LongInt;
begin

lCoordinate := 0;
lPosSize := 2;

for i := 0 to 1 do
begin
     lAxis[i] := i;
end;

dPos[0] := 4000;
dPos[1] := 4000;

AxmContiWriteClear(lCoordinate);
AxmContiSetAxisMap(lCoordinate, lPosSize, @lAxis);

{ 상대위치 구동 }
AxmContiSetAbsRelMode(lCoordinate, 1);

{ 직선보간 구동 }
AxmLineMove(lCoordinate, @dPos, 200, 400, 400);

{ 0축의 이동 모드 확인 }
AxmContiGetAbsRelMode (lCoordinate, @Mode);
end;
```

See Also

AxmLineMove, AxmLineArcMove, AxmCircleCenterMove, AxmCirclePointMove, AxmCircleRadiusMove, AxmCircleAngleMove, AxmContiSetAxisMap, AxmContiGetAxisMap, AxmContiSetAbsRelMode

AxmContIsMotion

Purpose

보간 구동 중인지 확인 하는 함수이다.

Format

C

DWORD AxmContIsMotion (long lCoord, DWORD *upInMotion);

Visual Basic

Function AxmContIsMotion (ByVal lCoord As Long, ByRef upInMotion As Long) As Long

Delphi

function AxmContIsMotion (lCoord: LongInt; upInMotion : PDWord) : DWord; stdcall;
;

Input / Output

Name	in/out	Format	Init Value	Explanation
Coord	in	long	-	좌표계 번호(0 ~ (최대축수 - 1))
InMotion	out	DWORD*	-	보간 동작 임을 확인

InMotion

#define	Value	Explanation
FALSE	00h	보간 구동 중이 아님
TRUE	01h	보간 구동 중임

Return AXT_RT_SUCCESS(0000) : Successful execution of API

* See error code Table more information on status error codes

Description

AxmLineMove, AxmCircleCenterMove, AxmCirclePointMove, AxmCircleRadiusMove, AxmCircleAngleMove 함수를 보간구동을 하는데. 하나의 AxmLineMove 함수가 1개로 간주된다.

C Example

```
DWORD IsContMotion;
long   lAxis[2];
double  dPos[2];
long   lPosSize = 2;
```

```
long lCoordinate = 0;
double  dVelocity = 200, dAccel = 400;

lAxis[0] = 0;
lAxis[1] = 1;

AxmContiSetAxisMap(lCoordinate, lPosSize, lAxis);
//절대위치 구동등록
AxmContiSetAbsRelMode(lCoordinate, 0);

dPos[0] = 100 , dPos[1] = 100;
AxmLineMove(lCoordinate, dPos, dVelocity, dAccel, dAccel);

//절대위치 구동시작
AxmContiSetAbsRelMode(lCoordinate, 0);

Sleep(100);
AxmContiIsMotion (lCoordinate, &IsContMotion);
if(IsContMotion ){ //  보간중인지 아닌지 확인한다..
MessageBox("현재 리스트 보간 모션이 수행중이다\n");
}
```

VB Example

```
Public Declare Sub Sleep Lib "kernel32" Alias "Sleep" (ByVal
       dwMilliseconds As Long)
'  베이직에서 Sleep 함수를 사용하기위해서 헤더파일에 반드시 선언해서 사용해야된다.
Dim  IsContMotion As Long
Dim strData As String
Dim dVelocity As Double
Dim dAccel As Double
Dim lPosSize As Long
Dim lCoordinate As Long
Dim dPos(0 To 1) As Double
Dim axes(0 To 1) As Long
Dim i As Long

dVelocity = 200
dAccel = 400
lPosSize = 2
lCoordinate = 0

For i = 0 To 1
  axes(i) = i
Next i

AxmContiSetAxisMap lCoordinate, lPosSize, axes(0)

' 절대위치구동 등록시작
   AxmContiSetAbsRelMode lCoordinate, 0

AxmLineMove lCoordinate, dPos(0), dVelocity, dAccel, dAccel
dPos(0) = 200: dPos(1) = 200
```

```
' 절대위치구동시작
AxmContiSetAbsRelMode lCoordinate, 0


Sleep 100
AxmContiIsMotion lCoordinate, IsContMotion
If(IsContMotion = 1) Then
MsgBox "현재 리스트 보간 모션이 수행중이다.", vbOKCancel
End If
```

Delphi Example

```
var
   i : LongInt;
   dVelocity, dAccel : Double;
   lAxis : array [0..1] of LongInt;
   dPos : array [0..1] of Double;
   lCoordinate : LongInt;
   lPosSize : LongInt;
IsContMotion : DWORD;

begin

  dVelocity := 200;
  dAccel := 400;
  lPosSize := 2;
  lCoordinate := 0;

//절대위치구동등록
AxmContiSetAbsRelMode(lCoordinate, 0);

for i := 0 to 1 do
begin
  lAxis[i] := i;
end;


AxmContiSetAxisMap(lCoordinate, 2, @lAxis);

dPos[0] := 100 ; dPos[1] := 100;
AxmLineMove(lCoordinate, @dPos, dVelocity, dAccel, dAccel);
dPos[0] := 200  ; dPos[1] := 200;

// 절대위치 구동시작
AxmMotSetAbsRelMode(lCoordinate, 0);

Sleep(100);
AxmContiIsMotion (lCoordinate, @IsContMotion);
if(IsContMotion = 1 ) then  {보간중인지 아닌지 확인한다.}
begin
Application.MessageBox('현재 리스트 보간 모션이 수행중이다.', 'Ajinextek',
       MB_OK);
```

```
End ;
end;
```

See Also

AxmLineMove, AxmLineArcMove, AxmCircleCenterMove, AxmCirclePointMove, AxmCircleRadiusMove, AxmCircleAngleMove, AxmContiGetAxisMap, AxmContiSetAbsRelMode, AxmContiGetAbsRelMode

Trigger 함수

Function	Description
AxmTriggerSetTimeLevel	지정 축에 트리거 신호 지속 시간 및 트리거 출력 레벨, 트리거 출력방법을 설정한다.
AxmTriggerGetTimeLevel	지정 축에 트리거 신호 지속 시간 및 트리거 출력 레벨, 트리거 출력방법을 반환한다.
AxmTriggerSetAbsPeriod	지정 축에 트리거 기능의 사용 여부, 출력 레벨, 위치 비교기, 트리거 신호 지속 시간 및 트리거 출력 모드를 설정한다.
AxmTriggerGetAbsPeriod	지정 축에 트리거 기능의 사용 여부, 출력 레벨, 위치 비교기, 트리거 신호 지속 시간 및 트리거 출력 모드를 반환한다.
AxmTriggerSetBlock	사용자가 지정한 시작위치부터 종료위치까지 일정구간마다 트리거를 출력하는 한다.
AxmTriggerGetBlock	‘AxmTriggerSetBlock’ 함수의 트리거 설정한 값을 읽는다..
AxmTriggerOneShot	사용자가 한 개의 트리거 펄스를 출력한다..
AxmTriggerOnlyAbs	절대위치 트리거 무한대 절대위치 출력한다.
AxmTriggerSetReset	트리거 설정을 리셋한다.

트리거 신호의 사용

AXM에서는 모션구동 중 특정한 상황에 따라 인터럽트 뿐만 아니라 5V레벨의 트리거 신호를 내보낼 수도 있다. 트리거 신호는 TRIG 신호핀에서 나오며, ActiveHIGH로 설정해두면 평소에는 0V이다가 트리거 신호가 발생하면 설정해준 시간 동안 5V였다가 다시 0V가 된다. ActiveLow일 때는 그와 반대이다. 물론 기구부에서는 24V가 된다. 트리거 신호를 발생시키기 위해서는 먼저 AxmTriggerSetTimeLevel 함수로 트리거의 신호 레벨과 지속시간 등을 설정하고, 특정조건에서 트리거를 발생시키는 함수들을 호출하면 된다. 트리거 발생 함수는 특정위치에서 트리거를 발생하는 AxmTriggerSetAbsPeriod / AxmTriggerOnlyAbs 함수와, 특정구간 동안 일정 간격으로 트리거를 발생시키는 AxmTriggerSetBlock 함수, 그리고 임의의 시점에 한번의 트리거를 발생시키는 AxmTriggerOneShot 함수가 있다.

AxmTriggerSetTimeLevel

Purpose

지정 축에 트리거 출력시간, 트리거레벨, 트리거출력방법, 트리거 출력 시 인터럽트 설정여부를 설정한다.

Format

C

```
DWORD AxmTriggerSetTimeLevel (long lAxisNo, double dTrigTime, DWORD uTriggerLevel,
DWORD uSelect, DWORD uInterrupt);
```

Visual Basic

```
Function AxmTriggerSetTimeLevel (ByVal lAxisNo As Long, ByVal dTrigTime As Double, ByVal
uTriggerLevel As Long, ByVal uSelect As Long ,ByVal uInterrupt As Long) As Long
```

Delphi

```
function AxmTriggerSetTimeLevel (lAxisNo : LongInt; dTrigTime : Double; uTriggerLevel : DWord;
uSelect: DWord :uInterrupt: DWord) : DWord; stdcall;
```

Input / Output

Name	in/out	Format	Init Value	Explanation
AxisNo	in	long	-	채널(축) 번호(0 ~ (최대축수 - 1))
TrigTime	In	double	-	트리거 출력 신호 시간 설정 1 - 50000 (50msec)
TriggerLevel	In	DWORD	-	트리거 출력 신호의 사용 여부 및 Active Level
Select	In	DWORD	-	트리거 기준 위치값 입력 소스 선택
Interrupt	in	DWORD	-	트리거 출력 시 인터럽트 출력 여부

TriggerLevel

#define	Value	Explanation
LOW	00h	B 접점(NORMAL CLOSE) Active Low Level Trigger 신호 사용
HIGH	01h	A 접점(NORMAL OPEN) Active High Level Trigger 신호 사용

UNUSED	02h	Trigger 신호 사용안함
USED	03h	Trigger 신호 사용함 (기존상태유지)

Select

#define	Value	Explanation
COMMAND	00h	목표 위치
ACTUAL	01h	실제 위치

uInterrupt

uUse	Value	Explanation
DISABLE	00h	인터럽트 사용하지 않음
ENABLE	01h	인터럽트 사용함

Return AXT_RT_SUCCESS(0000) : Successful execution of API

* See error code Table more information on status error codes

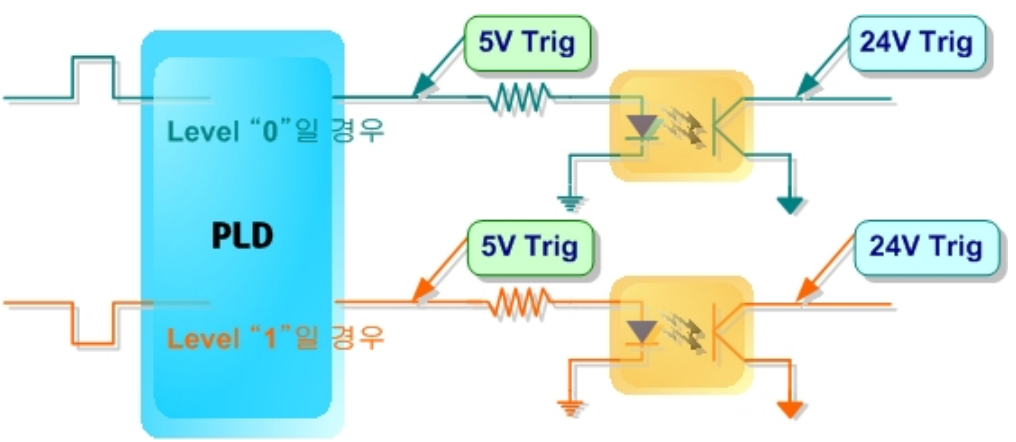
Description

Low(B접점) : 평상시 Close -> 감지되면 Open

High(A점점): 평상시 Open -> 감지되면 Close

Falling Edge : 평상시 High 상태를 유지하다가 트리거 시점에서 Low로 떨어졌다가 다시 High로 올라간다.

Riging Edge : 평상시 Low 상태를 유지하다가 트리거 시점에서 High로 올라갔다가 다시 Low로 떨어진다.



※ 알림 : 5V트리거 출력의 경우 Active Level을 0으로 설정했다면 평상시0V에서 트리거 발생시 5V가 되고 24V트리거의 경우 Open상태에서 트리거가 출력되면 N24V가 된다. 따라서 단자대를 통해 24V로 트리거를 사용할 경우 트리거 출력을 확인하기 위해서는 2kΩ정도의 풀업 저항을 달아 측정해야 된다.

dTrigTime : 트리거 출력 시간

EIP-M2/4Q: 1usec - 최대 50msec (1 - 50000 까지 설정)

C Example

```
// 0 축의 트리거 기능의 트리거 출력시간, 레벨, 출력방법,출력시 인터럽트 설정
AxmTriggerSetTimeLevel(0, 4000, HIGH, 0, 1);

double dTrigTime;
DWORD dwTriggerLevel;
DWORD dwSelect;
DWORD dwInterrupt;

// 0 축의 트리거 기능의 트리거 출력시간, 레벨, 출력방법,출력시 인터럽트 설정여부를 반환한
       다.
AxmTriggerGetTimeLevel(0, &dTrigTime, &dwTriggerLevel, &dwSelect, &
 dwInterrupt);
```

VB Example

```
'0 축의 트리거 기능의 트리거 출력시간, 레벨, 출력방법,출력시 인터럽트 설정
AxmTriggerSetTimeLevel 0, 4000, HIGH, 0, 1

Dim dTrigTime As Double
Dim dwTriggerLevel, dwSelect, dwInterrupt As Long
'0 축의 트리거 기능의 트리거 출력시간, 레벨, 출력방법,출력시 인터럽트 설정여부를 반환한다.
AxmTriggerGetTimeLevel 0, dTrigTime, dwTriggerLevel, dwSelect, dwInterrupt
```

Delphi Example

```
var
dwTriggerLevel, dwSelect, dwInterrupt : Dword;
dTrigTime : Double;

Begin

{ 0 축의 트리거 기능의 트리거 출력시간, 레벨, 출력방법,출력시 인터럽트 설정 }
AxmTriggerSetTimeLevel(0, 4000, HIGH, 0, 1);

{ 0 축의 트리거 기능의 트리거 출력시간, 레벨, 출력방법,출력시 인터럽트 설정여부를 반환한
       다.}
AxmTriggerGetTimeLevel(0, @dTrigTime, @dwTriggerLevel, @dwSelect,
 @dwInterrupt);
End;
```

See Also

AxmTriggerGetTimeLevel, AxmTriggerSetAbsPeriod, AxmTriggerGetAbsPeriod, AxmTriggerSetBlock, AxmTriggerGetBlock, AxmTriggerOneShot, AxmTriggerOnlyAbs, AxmTriggerSetReset

AxmTriggerGetTimeLevel

Purpose

지정 축의 트리거 출력시간, 트리거레벨, 트리거출력방법, 트리거 출력시 인터럽트 설정여부를 반환한다.

Format

C

```
DWORD AxmTriggerGetTimeLevel (long lAxisNo, double *dpTrigTime, DWORD *upTriggerLevel,
DWORD *upSelect, DWORD *upInterrupt);
```

Visual Basic

```
Function AxmTriggerGetTimeLevel (ByVal lAxisNo As Long, ByVal dpTrigTime As Double, ByRef
upTriggerLevel As Long, ByRef upSelect As Long , ByRef upInterrupt As Long) As Long
```

Delphi

```
function AxmTriggerGetTimeLevel (lAxisNo : LongInt; dpTrigTime : PDouble; upTriggerLevel :
PDWord; upSelect : PDWord; upInterrupt : PDWord) : DWord; stdcall;
```

Input / Output

Name	in/out	Format	Init Value	Explanation
AxisNo	in	long	-	채널(축) 번호(0 ~ (최대축수 - 1))
TrigTime	out	double*	-	트리거 출력 신호 시간 설정 최소(1) - 최대(4000) 설정가능 (1 usec) - (4 msec)
TriggerLevel	out	DWORD*	-	트리거 출력 신호의 사용 여부 및 Active Level
Select	out	DWORD*	-	트리거 기준 위치값 입력 소스 선택
Interrupt	out	DWORD*	-	트리거 출력 시 인터럽트 출력 여부

TriggerLevel

#define	Value	Explanation
LOW	00h	B 접점(NORMAL CLOSE) Active Low Level Trigger 신호 사용
HIGH	01h	A 접점(NORMAL OPEN)

		Active High Level Trigger 신호 사용
UNUSED	02h	Trigger 신호 사용안함
USED	03h	Trigger 신호 사용함 (기존상태유지)

Select

#define	Value	Explanation
COMMAND	00h	목표 위치
ACTUAL	01h	실제 위치

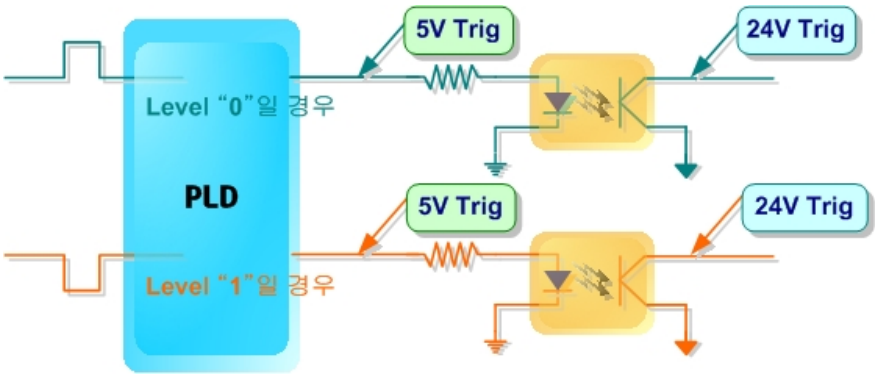
uInterrupt

uUse	Value	Explanation
DISABLE	00h	인터럽트 사용하지 않음
ENABLE	01h	인터럽트 사용함

Return AXT_RT_SUCCESS(0000) : Successful execution of API
* See error code Table more information on status error codes

Description

참고 : 필요한 데이터만 읽고 싶을 때는 필요하지 않는 변수에 NULL 값을 넣어 사용 할 수 있다.
'AxmTriggerSetTimeLevel' 설정된 트리거 출력시간, 트리거레벨, 트리거출력방법, 트리거 출력시 인터럽트 설정여부를 반환한다.
Low(B접점) : 평상시 Close -> 감지되면 Open
High(A점점): 평상시 Open -> 감지되면 Close
Falling Edge : 평상시 High 상태를 유지하다가 트리거 시점에서 Low로 떨어졌다가 다시 High로 올라간다.
Riging Edge : 평상시 Low 상태를 유지하다가 트리거 시점에서 High로 올라갔다가 다시 Low로 떨어진다.



※ 알림 : 5V트리거 출력의 경우 Active Level을 0으로 설정했다면 평상시 0V에서 트리거 발생시 5V가 되고 24V트리거의 경우 Open상태에서 트리거가 출력되면 N24V가 된다. 따라서 단자대를 통해 24V로 트리거를 사용할 경우 트리거 출력을 확인하기 위해서는 2kΩ정도의 풀업 저항을 달아 측정해야 된다.

C Example

```
// 0 축의 트리거 기능의 트리거 출력시간, 레벨, 출력방법,출력시 인터럽트 설정
AxmTriggerSetTimeLevel(0, 4000, HIGH, 0, 1);

double dTrigTime;
DWORD dwTriggerLevel;
DWORD dwSelect;
DWORD dwInterrupt;

// 0 축의 트리거 기능의 트리거 출력시간, 레벨, 출력방법,출력시 인터럽트 설정여부를 반환한
      다.
AxmTriggerGetTimeLevel(0, &dTrigTime, &dwTriggerLevel, &dwSelect, &
      dwInterrupt);
```

VB Example

```
'0 축의 트리거 기능의 트리거 출력시간, 레벨, 출력방법,출력시 인터럽트 설정
AxmTriggerSetTimeLevel 0, 4000, HIGH, 0, 1

Dim dTrigTime As Double
Dim dwTriggerLevel, dwSelect, dwInterrupt As Long
'0 축의 트리거 기능의 트리거 출력시간, 레벨, 출력방법,출력시 인터럽트 설정여부를 반환한다.
AxmTriggerGetTimeLevel 0, dTrigTime, dwTriggerLevel, dwSelect, dwInterrupt
```

Delphi Example

```
var
dwTriggerLevel, dwSelect, dwInterrupt : Dword;
```

```
dTrigTime : Double;

Begin

{ 0 축의 트리거 기능의 트리거 출력시간, 레벨, 출력방법,출력시 인터럽트 설정 }
AxmTriggerSetTimeLevel(0, 4000, HIGH, 0, 1);

{ 0 축의 트리거 기능의 트리거 출력시간, 레벨, 출력방법,출력시 인터럽트 설정여부를 반환한
       다.}
AxmTriggerGetTimeLevel(0, @dTrigTime, @dwTriggerLevel, @dwSelect,
       @dwInterrupt);
End;
```

See Also

AxmTriggerSetTimeLevel, AxmTriggerSetAbsPeriod, AxmTriggerGetAbsPeriod, AxmTriggerSetBlock, AxmTriggerGetBlock, AxmTriggerOneShot, AxmTriggerOnlyAbs, AxmTriggerSetReset

AxmTriggerSetAbsPeriod

Purpose

지정 축에 트리거 기능을 주기, 절대 위치 모드로 설정한다.

Format

C

```
DWORD AxmTriggerSetAbsPeriod (long lAxisNo, DWORD uMethod, double dPos));
```

Visual Basic

```
Function AxmTriggerSetAbsPeriod(ByVal lAxisNo As Long, ByVal uMethod As Long, ByVal dPos As Double) As Long
```

Delphi

```
function AxmTriggerSetAbsPeriod (lAxisNo : LongInt; uMethod : DWord; dPos : Double) : DWord; stdcall;
```

Input / Output

Name	in/out	Format	Init Value	Explanation
AxisNo	in	long	-	채널(축) 번호(0 ~ (최대축수 - 1))
Method	In	DWORD	-	트리거 출력 모드
Pos	In	double	-	트리거 신호 출력할 위치값 (주기사용시: 위치마다 트리거출력)

Method

#define	Value	Explanation
PERIOD_MODE	00h	트리거 위치값을 사용한 주기 트리거 방식
ABS_POS_MODE	01h	트리거 절대위치에서 트리거 발생

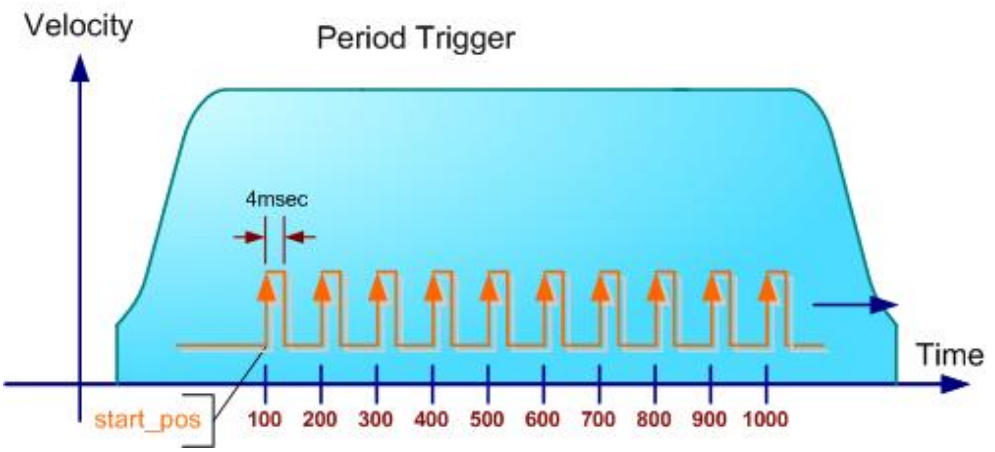
Return AXT_RT_SUCCESS(0000) : Successful execution of API

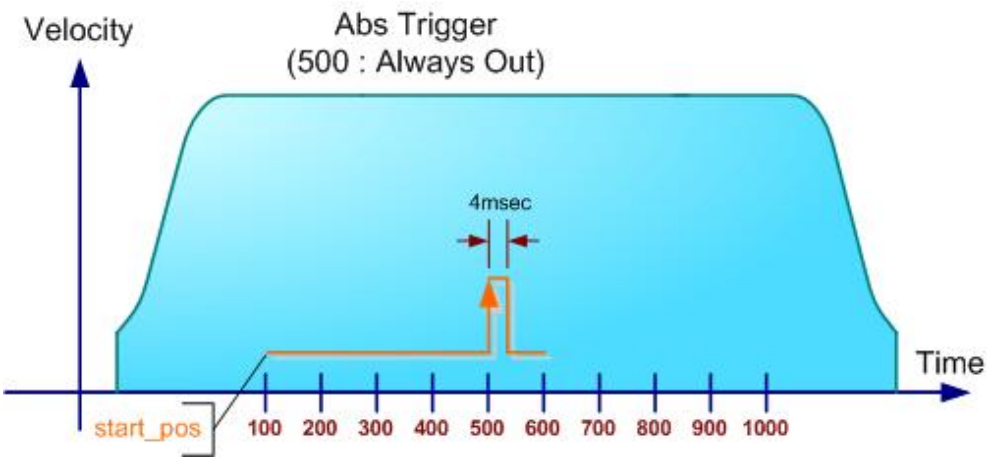
* See error code Table more information on status error codes

Description

‘ AxmTriggerSetTimeLevel’ 트리거 기능의 트리거 출력시간, 레벨, 출력방법,출력시 인터럽트 설정 되어 있다. 이함수랑 반드시 같이 사용하여야 한다. 트리거 출력 레벨 에따라 아래와 같이 달라진다.

주의점: AxmTriggerSetBlock함수와 다르게 시작한 위치부터 카운터 변화량을 기준으로 트리거 출력된다. 트리거모드를 주기모드로 설정시 설정할 때 트리거가 출력된다.
Low(B접점) : 평상시 Close -> 감지되면 Open
High(A점점): 평상시 Open -> 감지되면 Close
Falling Edge : 평상시 High 상태를 유지하다가 트리거 시점에서 Low로 떨어졌다가 다시 High로 올라간다.
Riging Edge : 평상시 Low 상태를 유지하다가 트리거 시점에서 High로 올라갔다가 다시 Low로 떨어진다..
주기모드 : Position이 100로 했을경우 100 펄스 마다 트리거 출력이 나온다.





절대모드 : Position이 500로 했을경우 500위치에오면 무조건 트리거가 출력된다.
100 펄스 마다 트리거 출력이 나온다.

C Example

```
// 0 축의 트리거 기능의 트리거 출력시간, 레벨, 출력방법,출력시 인터럽트 설정
AxmTriggerSetTimeLevel(0, 4000, HIGH, 0, 1);
// 0 축의 트리거 기능을 주기로 설정
AxmTriggerSetAbsPeriod (0, PERIOD_MODE, 100);

// 0 축의 트리거 기능의 사용 여부를 확인
double uPos;
DWORD uMode;

AxmTriggerGetAbsPeriod (0, &uMode, &uPos);
```

VB Example

```
‘0 축의 트리거 기능의 트리거 출력시간, 레벨, 출력방법,출력시 인터럽트 설정
AxmTriggerSetTimeLevel 0, 4000, HIGH, 0, 1
‘ 0 축의 트리거 기능을 주기로 설정
AxmTriggerSetAbsPeriod 0, PERIOD_MODE, 100

‘ 0 축의 트리거 기능의 사용 여부를 확인
Dim uMode As Long
Dim uPos As Double

AxmTriggerGetAbsPeriod 0, uMode, uPos
```

Delphi Example

```
{ 0 축의 트리거 기능의 사용 여부를 확인 }
var
uMode: DWord;
uPos: Double;

Begin

{ 0 축의 트리거 기능의 트리거 출력시간, 레벨, 출력방법,출력시 인터럽트 설정 }
AxmTriggerSetTimeLevel(0, 4000, HIGH, 0, 1);
{ 0 축의 트리거 기능을 주기로 설정 }
AxmTriggerSetAbsPeriod (0, PERIOD_MODE, 100);

AxmTriggerGetAbsPeriod (0, @uMode, @uPos);
end;
```

See Also

AxmTriggerSetTimeLevel, AxmTriggerGetTimeLevel, AxmTriggerGetAbsPeriod, AxmTriggerSetBlock, AxmTriggerGetBlock, AxmTriggerOneShot, AxmTriggerOnlyAbs, AxmTriggerSetReset

AxmTriggerGetAbsPeriod

Purpose

지정 축에 트리거 기능을 주기, 절대 위치 모드로 반환한다.

Format

C

```
DWORD AxmTriggerGetAbsPeriod (long lAxisNo, DWORD *upMethod, double *dpPos);
```

Visual Basic

```
Function AxmTriggerGetAbsPeriod (ByVal lAxisNo As Long, ByRef upMethod As Long, ByRef
dpPos As Double) As Long
```

Delphi

```
function AxmTriggerGetAbsPeriod (lAxisNo : LongInt; upMethod : PDWord; dpPos : PDouble) :
DWord; stdcall;
```

Input / Output

Name	in/out	Format	Init Value	Explanation
AxisNo	in	long	-	채널(축) 번호(0 ~ (최대축수 - 1))
Method	out	DWORD*	-	트리거 출력 모드
Pos	out	double*	-	트리거 신호 출력할 위치값 (주기사용시: 위치마다 트리거출력)

Method

#define	Value	Explanation
PERIOD_MODE	00h	트리거 위치값을 사용한 주기 트리거 방식
ABS_POS_MODE	01h	트리거 절대위치에서 트리거 발생

Return AXT_RT_SUCCESS(0000) : Successful execution of API

* See error code Table more information on status error codes

Description

참고 : 필요한 데이터만 가지고 올 경우, 필요하지 않는 변수에 NULL값을 넣어 사용 할 수 있다.

'AxmTriggerSetAbsPeriod' 함수의 설정된 지정 축에 트리거 기능을 주기, 절대 위치 모드로 반환한다.

C Example

```
// 0 축의 트리거 기능의 트리거 출력시간, 레벨, 출력방법,출력시 인터럽트 설정
AxmTriggerSetTimeLevel(0, 4000, HIGH, 0, 1);
// 0 축의 트리거 기능을 주기로 설정
AxmTriggerSetAbsPeriod (0, PERIOD_MODE, 100);

// 0 축의 트리거 기능의 사용 여부를 확인
double uPos;
DWORD uMode;

AxmTriggerGetAbsPeriod (0, &uMode, &uPos);
```

VB Example

```
‘0 축의 트리거 기능의 트리거 출력시간, 레벨, 출력방법,출력시 인터럽트 설정
AxmTriggerSetTimeLevel 0, 4000, HIGH, 0, 1
‘ 0 축의 트리거 기능을 주기로 설정
AxmTriggerSetAbsPeriod 0, PERIOD_MODE, 100

‘ 0 축의 트리거 기능의 사용 여부를 확인
Dim uMode As Long
Dim uPos As Double

AxmTriggerGetAbsPeriod 0, uMode, uPos
```

Delphi Example

```
{ 0 축의 트리거 기능의 사용 여부를 확인 }
var
uMode: DWord;
uPos: Double;

Begin

{ 0 축의 트리거 기능의 트리거 출력시간, 레벨, 출력방법,출력시 인터럽트 설정 }
AxmTriggerSetTimeLevel(0, 4000, HIGH, 0, 1);
{ 0 축의 트리거 기능을 주기로 설정 }
AxmTriggerSetAbsPeriod (0, PERIOD_MODE, 100);

AxmTriggerGetAbsPeriod (0, @uMode, @uPos);
end;
```

See Also

AxmTriggerSetTimeLevel, AxmTriggerGetTimeLevel, AxmTriggerSetAbsPeriod, AxmTriggerSetBlock, AxmTriggerGetBlock, AxmTriggerOneShot, AxmTriggerOnlyAbs, AxmTriggerSetReset

AxmTriggerSetBlock

Purpose

사용자가 지정한 시작위치부터 종료위치까지 일정구간마다 트리거를 출력을 설정한다.

Format

C

```
DWORD AxmTriggerSetBlock (long lAxisNo, double dStartPos, double dEndPos, double
dPeriodPos);
```

Visual Basic

```
Function AxmTriggerSetBlock (ByVal lAxisNo As Long, ByVal dStartPos As Double, ByVal dEndPos
As Double, ByVal dPeriodPos As Double) As Long
```

Delphi

```
function AxmTriggerSetBlock (lAxisNo : LongInt; dStartPos : Double; dEndPos : Double;
dPeriodPos : Double) : DWord; stdcall;
```

Input / Output

Name	in/out	Format	Init Value	Explanation
AxisNo	in	long	-	채널(축) 번호(0 ~ (최대축수 - 1))
StartPos	In	double	-	트리거 신호를 시작할 위치값
EndPos	In	double	-	트리거 신호를 끝내는 위치값
PeriodPos	In	double	-	트리거 출력 주기 위치 설정

Return AXT_RT_SUCCESS(0000) : Successful execution of API

* See error code Table more information on status error codes

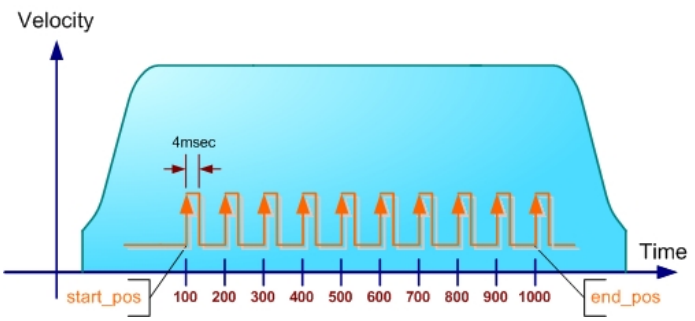
Description

‘ AxmTriggerSetTimeLevel’ 트리거 기능의 트리거 출력시간, 레벨, 출력방법,출력시 인터럽트 설정 되어 있다. 이함수랑 반드시 같이 사용하여야 한다. 트리거 출력 레벨 에따라 아래와 같이 달라진다.

주의사항: 트리거 위치를 설정 할 경우 반드시 UNIT/PULSE를 맞추어서 설정한다. 위치를 UNIT/PULSE 보다 작게 할 경우 최소단위가 UNIT/PULSE로 맞추어지기 때문에 그위치에 출력 할 수 없다.

Low(B접점) : 평상시 Close -> 감지되면 Open

High(A점점): 평상시 Open -> 감지되면 Close
Falling Edge : 평상시 High 상태를 유지하다가 트리거 시점에서 Low로 떨어졌다가 다시 High로 올라간다.
Riging Edge : 평상시 Low 상태를 유지하다가 트리거 시점에서 High로 올라갔다가 다시 Low로 떨어진다.



C Example

```
// 0 축의 트리거 기능의 트리거 출력시간, 레벨, 출력방법,출력시 인터럽트 설정
AxmTriggerSetTimeLevel(0, 4000, HIGH, 0, 1);
// 100 ~ 1050 사이에 100펄스마다 트리거 출력을 4msec, 레벨 low설정,인터럽트 Enable.
AxmTriggerSetBlock (0, 100 , 1050, 100);

// 0 축의 'AxmSetBlockTrigger ' 함수의 트리거 설정한 값을 읽는다
double dpStartPos, dpEndPos, dpPeriodPos;

AxmTriggerGetBlock (0, &dpStartPos, &dpEndPos, &dpPeriodPos);
```

VB Example

```
'0 축의 트리거 기능의 트리거 출력시간, 레벨, 출력방법,출력시 인터럽트 설정
AxmTriggerSetTimeLevel 0, 4000, HIGH, 0, 1
'100 ~ 1050 사이에 100펄스마다 트리거 출력을 4msec, 레벨 low설정,인터럽트.
AxmTriggerSetBlock 0, 100, 1050, 100

'0 축의 'AxmSetBlockTrigger ' 함수의 트리거 설정한 값을 읽는다
Dim dpStartPos, dpEndPos, dpPeriodPos As Double

AxmTriggerGetBlock 0, dpStartPos, dpEndPos, dpPeriodPos
```

Delphi Example

```
{ 0 축의 'AxmSetBlockTrigger ' 함수의 트리거 설정한 값을 읽는다}
var
dpStartPos, dpEndPos, dpPeriodPos : Double;
```

```
Begin
{ 0 축의 트리거 기능의 트리거 출력시간, 레벨, 출력방법,출력시 인터럽트 설정 }
AxmTriggerSetTimeLevel(0, 4000, HIGH, 0, 1);
{100 ~ 1050 사이에 100펄스마다 트리거 출력을 4msec, 레벨 low설정,인터럽트.}
AxmTriggerSetBlock (0, 100, 1050, 100);

AxmTriggerGetBlock (0, @dpStartPos, @dpEndPos, @dpPeriodPos);
end;
```

See Also

AxmTriggerSetTimeLevel, AxmTriggerGetTimeLevel, AxmTriggerSetAbsPeriod, AxmTriggerGetAbsPeriod, AxmTriggerGetBlock, AxmTriggerOneShot, AxmTriggerOnlyAbs, AxmTriggerSetReset

AxmTriggerGetBlock

Purpose

사용자가 지정한 시작위치부터 종료위치까지 일정구간마다 트리거를 출력을 설정을 반환한다.

Format

C

DWORD AxmTriggerGetBlock (long lAxisNo, double *dpStartPos, double *dpEndPos, double *dpPeriodPos);

Visual Basic

Function AxmTriggerGetBlock (ByVal lAxisNo As Long, ByRef dpStartPos As Double, ByRef dpEndPos As Double, ByRef dpPeriodPos As Double) As Long

Delphi

function AxmTriggerGetBlock (lAxisNo : LongInt; dpStartPos : PDouble; dpEndPos : PDouble; dpPeriodPos : PDouble) : DWord; stdcall;

Input / Output

Name	in/out	Format	Init Value	Explanation
AxisNo	in	long	-	채널(축) 번호(0 ~ (최대축수 - 1))
StartPos	out	double*	-	트리거 신호를 시작할 위치값
EndPos	out	double*	-	트리거 신호를 끝내는 위치값
PeriodPos	out	double*	-	트리거 출력 주기 위치 설정

Return AXT_RT_SUCCESS(0000) : Successful execution of API

* See error code Table more information on status error codes

Description

참고 : 필요한 데이터만 가지고 올 경우, 필요하지 않는 변수에 NULL값을 넣어 사용 할 수 있다.
'AxmTriggerSetBlock' 성정된 트리거 설정한 값을 반환한다.

C Example

```
// 0 축의 트리거 기능의 트리거 출력시간, 레벨, 출력방법,출력시 인터럽트 설정
AxmTriggerSetTimeLevel(0, 4000, HIGH, 0, 1);
// 100 ~ 1050 사이에 100펄스마다 트리거 출력을 4msec, 레벨 low설정,인터럽트 Enable.
```

```
AxmTriggerSetBlock (0, 100 , 1050, 100);

// 0 축의 'AxmSetBlockTrigger ' 함수의 트리거 설정한 값을 읽는다
double dpStartPos, dpEndPos, dpPeriodPos;

AxmTriggerGetBlock (0, &dpStartPos, &dpEndPos, &dpPeriodPos);
```

VB Example

```
'0 축의 트리거 기능의 트리거 출력시간, 레벨, 출력방법,출력시 인터럽트 설정
AxmTriggerSetTimeLevel 0, 4000, HIGH, 0, 1
'100 ~ 1050 사이에 100펄스마다 트리거 출력을 4msec, 레벨 low설정,인터럽트.
AxmTriggerSetBlock 0, 100, 1050, 100

'0 축의 'AxmSetBlockTrigger ' 함수의 트리거 설정한 값을 읽는다
Dim dpStartPos, dpEndPos, dpPeriodPos As Double

AxmTriggerGetBlock 0, dpStartPos, dpEndPos, dpPeriodPos
```

Delphi Example

```
{ 0 축의 'AxmSetBlockTrigger ' 함수의 트리거 설정한 값을 읽는다}
var
dpStartPos, dpEndPos, dpPeriodPos : Double;

Begin
{ 0 축의 트리거 기능의 트리거 출력시간, 레벨, 출력방법,출력시 인터럽트 설정 }
AxmTriggerSetTimeLevel(0, 4000, HIGH, 0, 1);
{100 ~ 1050 사이에 100펄스마다 트리거 출력을 4msec, 레벨 low설정,인터럽트.}
AxmTriggerSetBlock (0, 100, 1050, 100);

AxmTriggerGetBlock (0, @dpStartPos, @dpEndPos, @dpPeriodPos);
end;
```

See Also

AxmTriggerSetTimeLevel, AxmTriggerGetTimeLevel, AxmTriggerSetAbsPeriod, AxmTriggerGetAbsPeriod, AxmTriggerSetBlock, AxmTriggerOneShot, AxmTriggerOnlyAbs, AxmTriggerSetReset

AxmTriggerOneShot

Purpose

사용자가 지정한 위치에 한 개의 트리거 펄스를 출력한다.

Format

C

```
DWORD AxmTriggerOneShot (long lAxisNo);
```

Visual Basic

```
Function AxmTriggerOneShot (ByVal lAxisNo As Long) As Long
```

Delphi

```
function AxmTriggerOneShot (lAxisNo : LongInt) : DWord; stdcall;
```

Input / Output

Name	in/out	Format	Init Value	Explanation
AxisNo	in	long	-	채널(축) 번호(0 ~ (최대축수 - 1))

Return AXT_RT_SUCCESS(0000) : Successful execution of API

* See error code Table more information on status error codes

Description

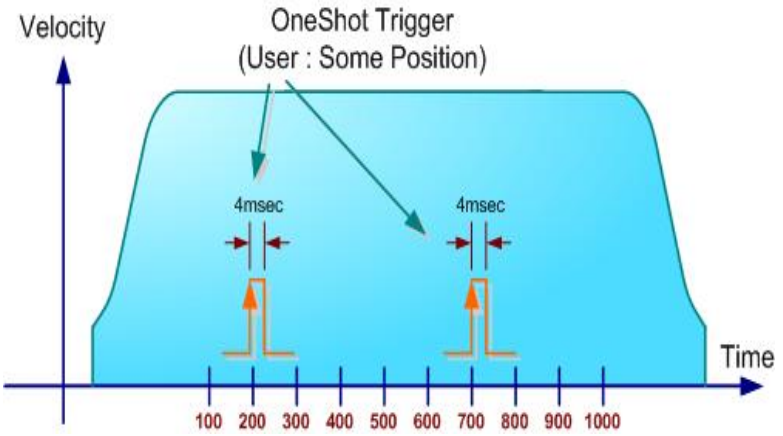
' AxmTriggerSetTimeLevel' 트리거 기능의 트리거 출력시간, 레벨, 출력방법,출력시 인터럽트 설정 되어 있다. 이함수랑 반드시 같이 사용하여야 한다. 트리거 출력 레벨 에따라 아래와 같이 달라진다.

Low(B접점) : 평상시 Close -> 감지되면 Open

High(A점점): 평상시 Open -> 감지되면 Close

Falling Edge : 평상시 High 상태를 유지하다가 트리거 시점에서 Low로 떨어졌다가 다시 High로 올라간다.

Riging Edge : 평상시 Low 상태를 유지하다가 트리거 시점에서 High로 올라갔다가 다시 Low로 떨어진다.



C Example

```
// 0 축의 트리거 기능의 트리거 출력시간, 레벨, 출력방법,출력시 인터럽트 설정
AxmTriggerSetTimeLevel(0, 4000, HIGH, 0, 1);
// 0 축의 한번의 트리거를 강제로 출력시킨다.
AxmTriggerOneShot (0);
```

VB Example

```
‘0 축의 트리거 기능의 트리거 출력시간, 레벨, 출력방법,출력시 인터럽트 설정
AxmTriggerSetTimeLevel 0, 4000, HIGH, 0, 1
‘0 축의 한번의 트리거를 강제로 출력시킨다.
AxmTriggerOneShot 0
```

Delphi Example

```
Begin
{ 0 축의 트리거 기능의 트리거 출력시간, 레벨, 출력방법,출력시 인터럽트 설정 }
AxmTriggerSetTimeLevel(0, 4000, HIGH, 0, 1);
{ 0 축의 한번의 트리거를 강제로 출력시킨다.}
AxmTriggerOneShot (0);
End;
```

See Also

AxmTriggerSetTimeLevel, AxmTriggerGetTimeLevel, AxmTriggerSetAbsPeriod, AxmTriggerGetAbsPeriod, AxmTriggerSetBlock, AxmTriggerGetBlock, AxmTriggerOnlyAbs, AxmTriggerSetReset

AxmTriggerOnlyAbs

Purpose

위치갯수와 상관없이 사용자가 지정한 절대위치에 트리거 펄스를 출력한다.

Format

C

```
DWORD AxmTriggerOnlyAbs (long lAxisNo, long lTrigNum, double* dpTrigPos);
```

Visual Basic

```
Function AxmTriggerOnlyAbs(ByVal lAxisNo As Long, ByVal lTrigNum As Long, ByRef dpTrigPos As
Double) As Long
```

Delphi

```
function AxmTriggerOnlyAbs (lAxisNo : LongInt; lTrigNum : LongInt; dpTrigPos : PDouble) :
DWord; stdcall;
```

Input / Output

Name	in/out	Format	Init Value	Explanation
AxisNo	in	long	-	채널(축) 번호(0 ~ (최대축수 - 1))
TrigNum	in	long	-	트리거 출력 위치 개수(1000개)
TrigPos	in	double*	-	트리거 위치 배열

Return AXT_RT_SUCCESS(0000) : Successful execution of API

* See error code Table more information on status error codes

Description

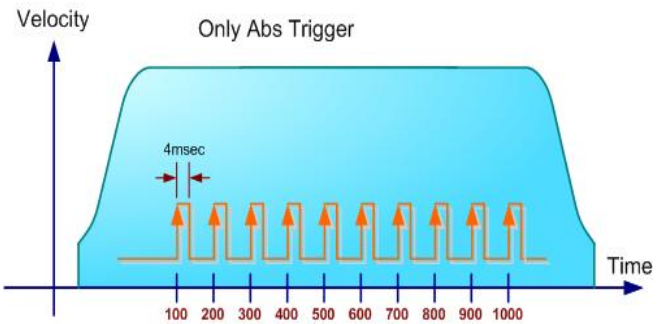
‘AxmTriggerSetTimeLevel’ 트리거 기능의 트리거 출력시간, 레벨, 출력방법,출력시 인터럽트 설정 되어 있다. 이함수랑 반드시 같이 사용하여야 한다. 트리거 출력 레벨 에따라 아래와 같이 달라진다.

Low(B접점) : 평상시 Close -> 감지되면 Open

High(A점점): 평상시 Open -> 감지되면 Close

Falling Edge : 평상시 High 상태를 유지하다가 트리거 시점에서 Low로 떨어졌다가 다시 High로 올라간다.

Riging Edge : 평상시 Low 상태를 유지하다가 트리거 시점에서 High로 올라갔다가 다시 Low로 떨어진다.



C Example

```
double trig_pos[10];
trig_pos[0] = 100;  trig_pos[1] = 200;  trig_pos[2] = 300;
trig_pos[3] = 400;  trig_pos[4] = 500;  trig_pos[5] = 600;
trig_pos[6] = 700;  trig_pos[7] = 800;  trig_pos[8] = 900;
trig_pos[9] = 1000;
// 0 축의 트리거 기능의 트리거 출력시간, 레벨, 출력방법,출력시 인터럽트 설정
AxmTriggerSetTimeLevel(0, 4000, HIGH, 0, 1);
// 0 축의 사용자의 절대위치를 설정한다.
AxmTriggerOnlyAbs (0, 10, trig_pos);// trig_time: 1usec ~ 4000 usec
```

VB Example

```
Dim trig_pos(0 To 9) As Double
trig_pos(0) = 100: trig_pos(1) = 200: trig_pos(2) = 300
trig_pos(3) = 400: trig_pos(4) = 500: trig_pos(5) = 600
trig_pos(6) = 700: trig_pos(7) = 800: trig_pos(8) = 900
trig_pos(9) = 1000
‘0 축의 트리거 기능의 트리거 출력시간, 레벨, 출력방법,출력시 인터럽트 설정
AxmTriggerSetTimeLevel 0, 4000, HIGH, 0, 1
‘0 축의 사용자의 절대위치를 설정한다.
AxmTriggerOnlyAbs 0, 10, trig_pos(0)    ' trig_time: 1usec ~ 4000 usec
```

Delphi Example

```
var
trig_pos : array [0..9] of Double;
Begin
trig_pos[0] := 100;  trig_pos[1] := 200;  trig_pos[2] := 300;
trig_pos[3] := 400;  trig_pos[4] := 500;  trig_pos[5] := 600;
trig_pos[6] := 700;  trig_pos[7] := 800;  trig_pos[8] := 900;
trig_pos[9] := 1000;
{ 0 축의 트리거 기능의 트리거 출력시간, 레벨, 출력방법,출력시 인터럽트 설정 }
AxmTriggerSetTimeLevel(0, 4000, HIGH, 0, 1);
{0 축의 사용자의 절대위치를 설정한다.}
AxmTriggerOnlyAbs (0, 10, @trig_pos);   { trig_time: 1usec ~ 4000 usec}
End;
```

See Also

AxmTriggerSetTimeLevel, AxmTriggerGetTimeLevel, AxmTriggerSetAbsPeriod, AxmTriggerGetAbsPeriod, AxmTriggerSetBlock, AxmTriggerGetBlock, AxmTriggerOneShot, AxmTriggerSetReset

AxmTriggerSetReset

Purpose

사용자가 설정한 트리거출력을 해제한다.

Format

C

DWORD AxmTriggerSetReset (long lAxisNo);

Visual Basic

Function AxmTriggerSetReset (ByVal lAxisNo As Long) As Long

Delphi

function AxmTriggerSetReset (lAxisNo : LongInt) : DWord; stdcall;

Input / Output

Name	in/out	Format	Init Value	Explanation
AxisNo	in	long	-	채널(축) 번호(0부터 시작함)

Return AXT_RT_SUCCESS(0000) : Successful execution of API

* See error code Table more information on status error codes

Description

‘ AxmTriggerSetTimeLevel’ 로 설정된 사용자가 지정한 축의 트리거의 설정 여부를 해제한다.

C Example

```
// 0축의 트리거 설정을 해제한다.
AxmTriggerSetReset(0);
```

VB Example

```
// 0축의 트리거 설정을 해제한다.
AxmTriggerSetReset 0
```

Delphi Example

```
begin
// 0축의 트리거 설정을 해제한다.
AxmTriggerSetReset(0);
End;
```

See Also

AxmTriggerSetTimeLevel, AxmTriggerGetTimeLevel, AxmTriggerSetAbsPeriod, AxmTriggerGetAbsPeriod, AxmTriggerSetBlock, AxmTriggerGetBlock, AxmTriggerOneShot, AxmTriggerOnlyAbs

겐트리 구동 함수

겐트리(Gantry)는 두 개의 모터 축이 기구물에 의해 연결되어 하나의 축처럼 제어되는 시스템을 말한다. 일반적으로 리니어 서보모터 2축을 사용하는 겐트리 시스템에서 두 겐트리간의 기구적인 편차가 일정값 이상 존재하면 서보모터에 과부하가 발생하여 알람이 발생할 여지가 있기 때문에 두 축에 전달되는 Command를 완전히 동기시키는 방법이 사용된다.

Function	Description
AxmGantrySetEnable	모션모듈은 두 축이 기구적으로 Link되어있는 겐트리 구동시스템 제어를 지원한다. 이 함수를 이용해 Master축을 겐트리 제어로 설정하면 해당 Slave축은 Master축과 동기되어 구동된다. 만약 겐트리 설정 이후 Slave축에 구동명령이나 정지 명령등을 내려도 모두 무시된다.
AxmGantryGetEnable	이 함수를 이용해 Master축을 겐트리 제어로 설정할 때 설정한 파라메타를 반환한다.
AxmGantrySetDisable	겐트리 제어로 설정후 원점검색을 수행하면 Master축이 원점검색을 수행한 후 Slave축이 원점검색을 수행 한다. 이 함수는 원점검색 이후 Master축 원점과 Slave축 원점간의 오차 값을 확인 할 때 사용한다.

두 축이 아래의 그림에서와 같이 기구적으로 Link되어있는 겐트리 구동시스템을 제어하기 위해 갠트리 구동 함수를 지원한다. 이 함수를 이용해 Master축을(짝수인 채널(축) 번호(0부터 시작함)) 겐트리 제어로 설정하면 해당 Slave축은 Master축과 동기되어 구동된다. 만약 겐트리 설정이후 Slave축에 구동명령이나 정지 명령등을 내려도 모두 무시된다.

※ 알림: 겐트리 시스템의 경우 각각의 로봇을 평행으로 놓고 기구적으로 Link하는 구조로 조립 후 Master축과 Slave축의 원점센서가 일정 값의 오차를 가지게 된다. 원점검색을 하게되면 두축의 원점센서가 가지는 오차를 마지막에 보상해 주게된다. 기구조립시 발생하는 센서간의 오차를 dSlOffset에 설정해 주면된다.

겐트리 제어에서 중요한 것은 원점검색을 수행하는데 있어 Master축과 Slave축을 고려해서 원점검색을 해야된다. 일반적인 순서는 master축의 홈센서를 먼저 찾은후 slave축의 홈센서을 찾는다는 겁니다. 제일 처음에 AxmGantrySetEnable 로 겐트리 제어로 묶어논다음에 겐트리를 홈 구동을시작하면 구동이 되고 일반적으로 원점검색을 사용하는 함수처럼 사용하면 된다

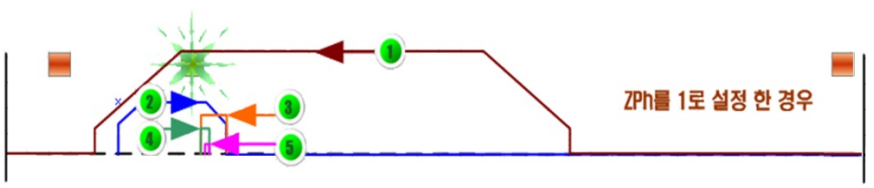
.

원점검색관련 설정변수	변수 설명
HmSig	원점센서로 사용할 신호를 설정 예) 원점센서(4), +Limit센서(0) 등
HmLev	원점센서의 Active Level을 설정 예) 0: A접점, 1: B접점
HmDir	원점센서 검출시 초기 검색진행 방향을 설정 예) 1: (+)방향, 0: (-)방향
Zphas	원점센서 검출 후 엔코더 Z상을 검출 여부 설정
VelFirst	원점검색 시 초기 고속검출 속도 (초기 원점센서가 감지되지 않았을 경우)
VelSecond	원점검색 시 1차 센서 검출 후 반대 방향으로 빠져나오는 속도
VelThird	원점검색 시 1차 센서 검출 후 재 검색 진행하는 속도
VelLast	원점검색 시 최종 검출 속도 설정 [원점검색의 정밀도 결정]

AccFirst	원점검색 시 초기 고속검출 가속도(가속도의 단위는 Unit/pulse를 1/1로 한경우에 PPS[Pulses/sec])
AccSecond	원점검색 시 1차 센서 검출 후 반대 방향으로 빠져나오는 가속도(가속도의 단위는 Unit/pulse를 1/1로 한경우에 PPS[Pulses/sec])
HClrTim	원점검색 완료 후 Cmd, Act(Encoder)위치를 Clear전 대기시간
HOffset	원점검색 완료 후 기구원점 재 설정 시 이동 Offset 값
NSWLimit	원점검색 완료 후 적용할 (-) Software Limit 값 설정
PSWLimit	원점검색 완료 후 적용할 (+) Software Limit 값 설정 (NSWLimit값이 PSWLimit값보다 같거나 클 경우 S/W Limit 설정되지 않음)

1. Master 축의 홈센서를 먼저 찾는다.

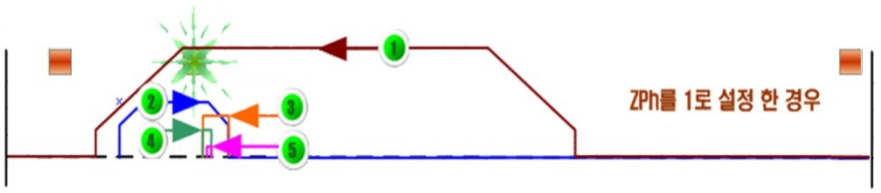
Zph가 1인 경우



4번 Step에서 Z상의 Down Edge를 검출하고 5번 Step에서 Z상의 Up Edge를 재 검출함으로 원점검색 속도와 정밀도를 높임

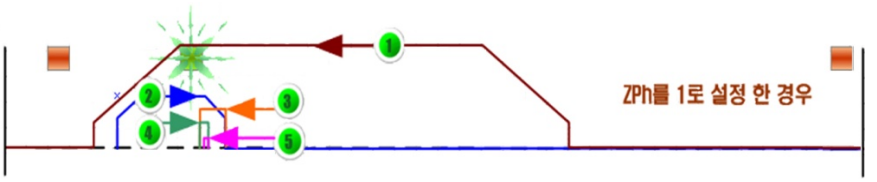
2. Slave 축의 홈센서를 찾은 다음 위치를 clear 한다.

Zph가 1인 경우



4번 Step에서 Z상의 Down Edge를 검출하고 5번 Step에서 Z상의 Up Edge를 재검출함으로 원점검색 속도와 정밀도를 높임

Step별 원점검출 속도설정의 이해 (Master, Slave 똑같은 속도사용)



1.

VelFirst, AccFirst사용

HmDir 방향으로 원점센서를 고속 검출하고 HAccF 감속도로 감속정지

2.

VelSecond, AccSecond사용

HmDir 반대 방향으로 원점센서의 Down Edge를 검출하고 HAccF감속도로 감속정지

3.

VelThird사용

HmDir 방향으로 원점센서의 Up Edge를 검출하고 급정지

4.

VelThird 사용

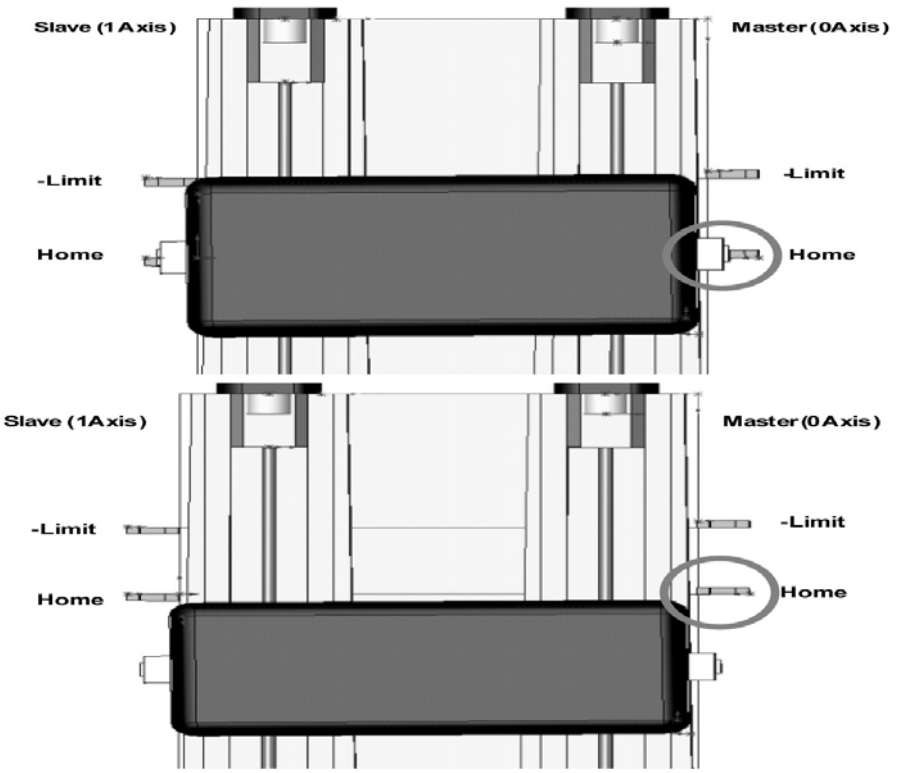
HmDir 반대 방향으로 Z상의 Down Edge를 검출하고 급정지

5.

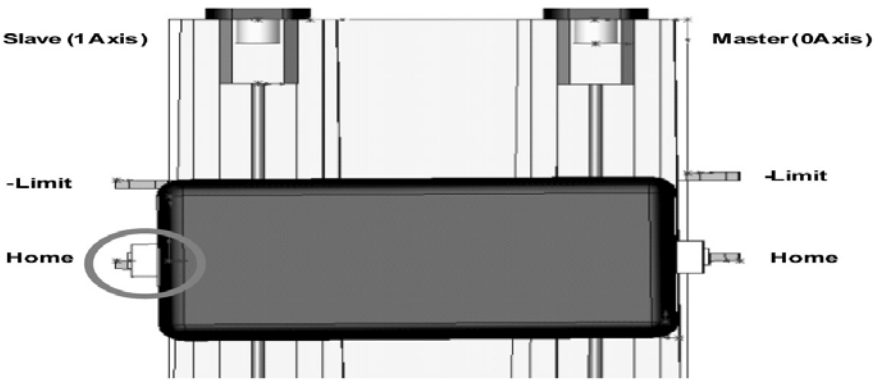
VelLast 사용

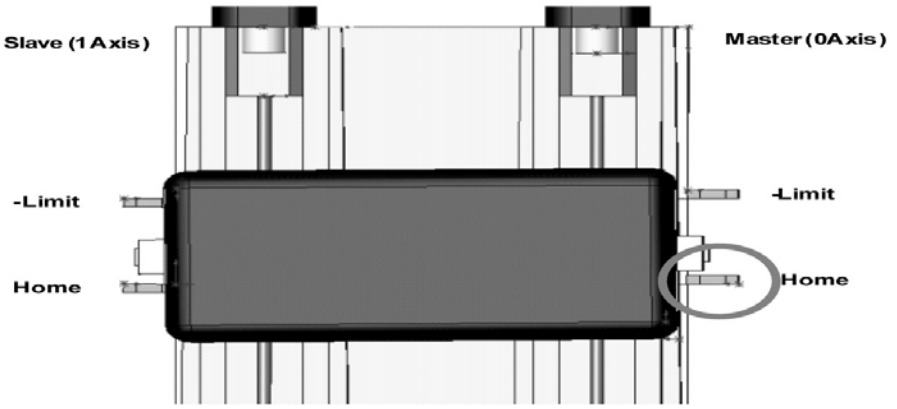
HmDir 방향으로 Z상의 Up Edge를 검출하고 급정지

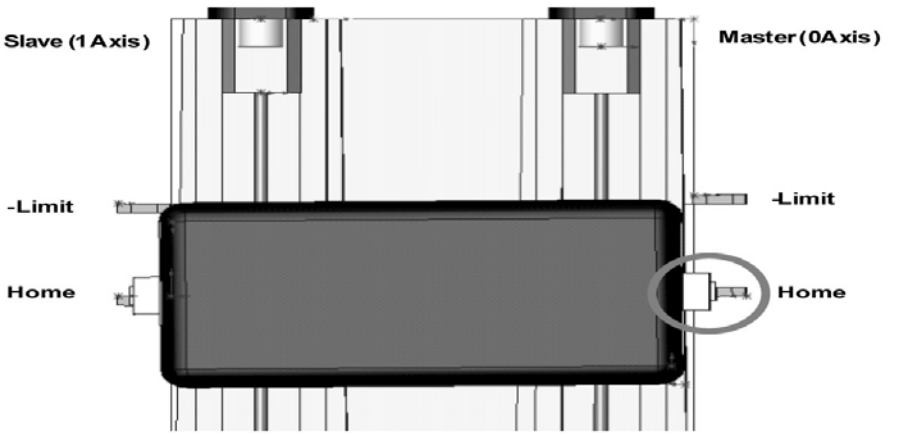
1. Master축의 홈센서를 먼저 찾는다. Maste축만 위치 0으로 만든다.

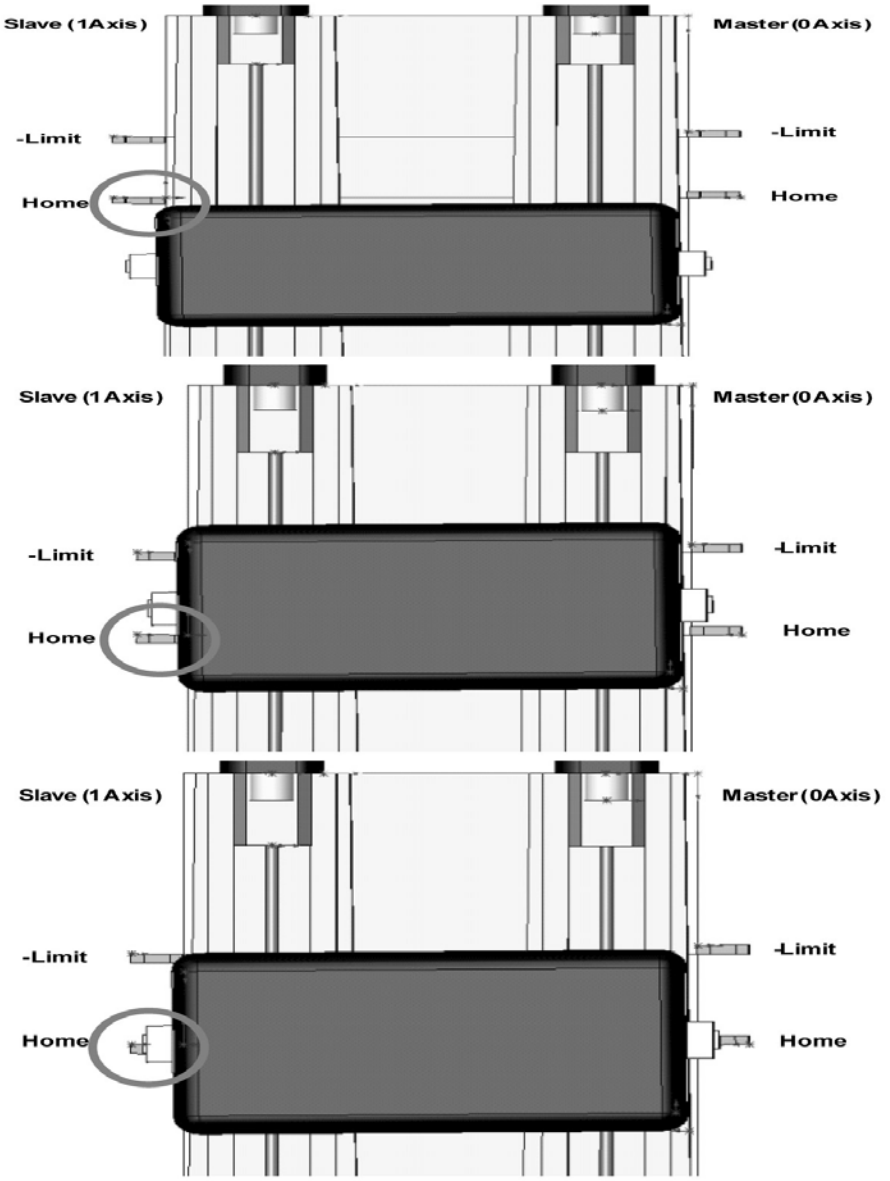


2. Slave축의 홈센서를 찾는다. 슬레이브축만 위치 0로 만든다. 그럼 마스터에 대한 슬레이브의 Offset 값을 얻을수있다.

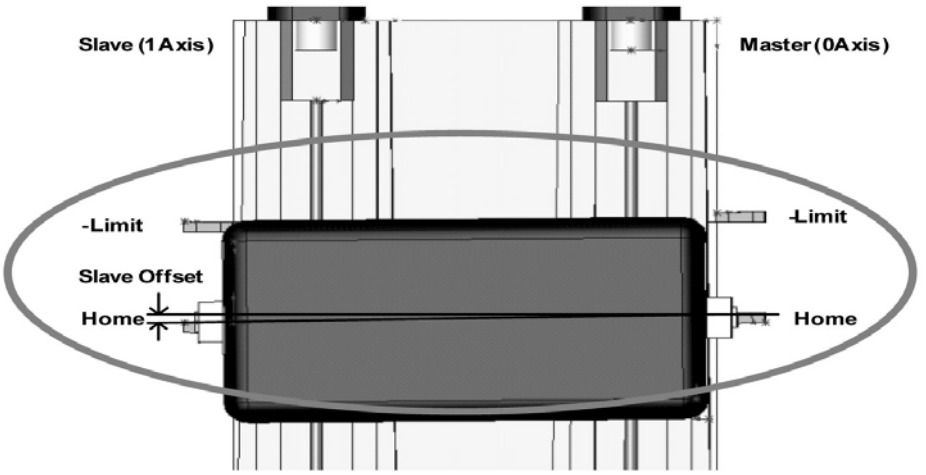



Slave (1Axis)
Master(0Axis)
-Limit
+Limit
Home
Home

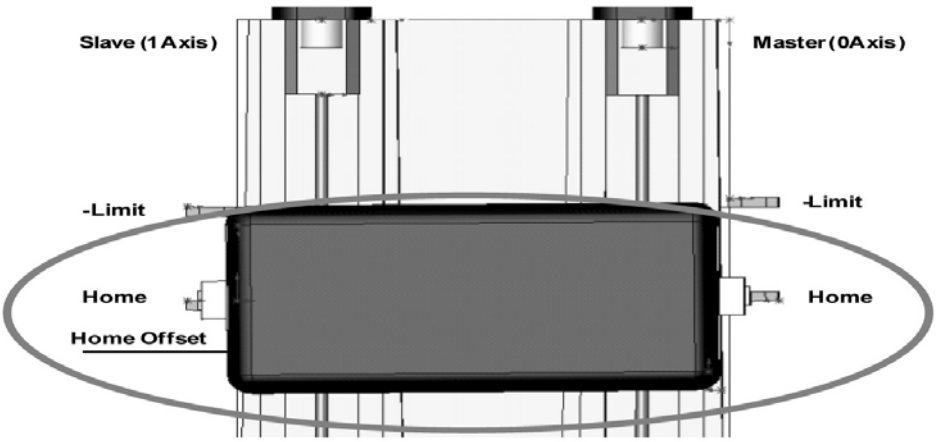

Slave (1Axis)
Master(0Axis)
-Limit
+Limit
Home
Home


Slave (1Axis)
Master(0Axis)
-Limit
-Limit
Home
Home
Slave (1Axis)
Master(0Axis)
-Limit
-Limit
Home
Home
Slave (1Axis)
Master(0Axis)
-Limit
-Limit
Home
Home

3. Master와 Slave의 Link를 풀고 Slave축을 Slave offset만큼 이동해준다. 그다음 Master축을 0으로 보낸다. 다음 다시 Link시킨다.



4. 홈 Offset 만큼 이동해준 다음 위치를 0로 만든다.
홈 Offset이 없을경우 이동하지 않고 위치를 0로 만든다.



5. Software Limit 설정 및 겐트리 구동하기

※ 주의 : 원점검색이 정상적으로 완료된 후 Soft Limit을 설정할 수 있다. Soft Limit은 기구적으로 Limit센서가 장착되어 있지 않거나 상황에 따라서 Limit을 변경해야 될 경우 사용하면 된다. 이때 한 가지 주의할 것은 NLimit(-)보다 PLimit(+)값이 커야 정상적으로 Soft Limit이 설정된다. 만약 NSwLimit값이 PSwlimit 값보다 같거나 클 경우 Soft Limit는 설정되지 않는다.

AxmGantrySetEnable

Purpose

이 함수를 이용해 Master축을 겐트리 제어로 설정하면 해당 Slave축은 Master축과 동기 시킨다. 겐트리 제어 기능을 활성화 시키고 이후 Slave축에 구동명령이나 정지 명령등을 내려도 모두 무시된다.

Format

C

```
DWORD AxmGantrySetEnable(long lMasterAxisNo, long lSlaveAxisNo, DWORD uSlHomeUse,
double dSlOffset, double dSlOffsetRange);
```

Visual Basic

```
Function AxmGantrySetEnable (ByVal lMasterAxisNo As Long, ByVal lSlaveAxisNo As Long, ByVal
uSlHomeUse As Long, ByVal dSlOffset As Double, ByVal dSlOffsetRange As Double) As Long
```

Delphi

```
function AxmGantrySetEnable (lMasterAxisNo : LongInt; lSlaveAxisNo : LongInt; uSlHomeUse :
DWord; dSlOffset : Double; dSlOffsetRange : Double) : DWord; stdcall;
```

Input/Output

Name	in/out	Format	Init Value	Explanation
MasterAxisNo	in	long	0	Master채널(축) 번호(0 ~ (최대축수 – 1))
SlaveAxisNo	in	long	0	Slave채널(축) 번호(0 ~ (최대축수 – 1))
SlHomeUse	In	DWORD	0	Master와 같이 Slave축도 원점검색을 같이 찾을것인지 선택인자
SlOffset	In	double	0	Master축의 원점센서와 Slave축 원점센서간의 기구적인 오차 값
SlOffsetRange	In	double	0	원점검색시 Master축의 원점센서와 Slave축 원점센서간에 허용할 최대 오차값 지정

uSlHomeUse

Value	Explanation
00h	Master 축만에 원점검색 선택
01h	Master축 과 Slave축에 원점검색 후, Slave 축 SlOffset위치 만큼 이동
02h	Master축 과 Slave축의 센서의 오차 값 확인

Return AXT_RT_SUCCESS(0000) : Successful execution of API
* See error code Table more information on status error codes

Description

이 함수를 이용해 Master축을 겐트리 제어로 설정하면 해당 Slave축은 Master축과 동기되어 구동된다. 만약 겐트리 설정 이후 Slave축에 구동명령이나 정지 명령등을 내려도 모두 무시된다

EIP-M2/4Q사용시 주의사항: 갠트리 ENABLE시 슬레이브축은 모션중 AxmStatusReadMotion 함수로 확인하면 InMotion 중이므로 True로 확인되어야 한다. 만약 InMotion 되지 않았으므로 알람 발생 여부 혹은 Limit 신호 발생 여부를 확인한다

겐트리 제어로 설정후 원점검색을 수행하면 Master축이 원점검색을 수행한 후 Slave축이 원점검색을 수행 한다. 이 함수는 원점검색 이후 Master축 원점과 Slave축 원점간의 오차 값을 확인 할 때 사용한다

uSlHomeUse : 슬레이축 홈사용 우뮤 (0 – 2)
(0 : 슬레이브축 홈을 사용안하고 마스터축을 홈을 찾는다.)
(1 : 마스터축 , 슬레이브축 홈을 찾는다. 슬레이브 dSlOffset 값 적용해서 보정함.)
(2 : 마스터축 , 슬레이브축 홈을 찾는다. 슬레이브 dSlOffset 값 적용해서 보정안함.)

Slave축의 Offset값을 알아내는방법.
A. 마스터, 슬레이브를 두개다 서보온을 시킨다.
B. AxmGantrySetEnable함수에서 uSlHomeUse = 1로 설정후 AxmHomeSetStart함수를 이용해서 홈을 찾는다.
C. 홈을 찾고 나면 마스터축의 Command값을 읽어보면 마스터축과 슬레이브축의 틀어진 Offset값을 볼수있다.
D. Offset값을 읽어서 AxmGantrySetEnable함수의 dSlOffset인자에 넣어준다.
E. dSlOffset값을 넣어줄때 마스터축에 대한 슬레이브 축 값이기 때문에 부호를 반대로 – dSlOffset 넣어준다.
F.dSlOffsetRange 는 Slave Offset의 Range 범위를 말하는데 Range의 한계를 지정하여 한계를 벗어나면 에러를 발생시킬때 사용한다.
G. AxmGantrySetEnable함수에 Offset값을 넣어줬으면 AxmGantrySetEnable함수에서 uSlHomeUse = 1로 설정후 AxmHomeSetStart함수를 이용해서 홈을 찾는다.

알림: 겐트리 시스템의 경우 각각의 로봇을 평행으로 놓고 기구적으로 Link하는 구조로 조립 후 Master축과 Slave축의 원점센서가 일정 값의 오차를 가지게 된다. 초기 기구 셋업시 원점 검색완료 후 이 함수를 이용해 오차값을 읽어 dSlOffset에 설정해 주면된다. (초기 적절한 Offset값을 찾을시만 사용됨)

▶ AxmGantrySetEnable 함수를 이용해 마스터, 슬레이브 축을 설정하면 레브 축은 모션 중이 설정된다. AxmStatusReadInMotion 함수의 Status값은 반드시 1로 읽여야 된다. 1이 아니면 겐트리 설정이

안 된 것이다.

▶주의점: 8축 이상을 구성하였을 때

8축 모션이상 보드는 Axis 0 – 3의 4개의 축이 축그룹 1로 구분되고 Axis4 – 7 4개의 축이 축그룹 2로 구분되고 Axis8 – 11 4개의 축이 축그룹 3으로 구분된다. 동일 그룹에 속하는 축들간의 직선보간구동, 원호보간구동을 사용할 수 있지만, 서로 다른 축그룹에 속한 축들간의 직선보간구동과 원호보간구동은 사용할 수 없다.

C Example

```
if(AxmGantrySetEnable (0, 1, TRUE, 0, 0.0) ==  AXT_RT_SUCCESS )
     printf("0축, 1축을 겐트리 제어로 설정했습니다.\n");
else
     printf("0축, 1축을 겐트리 제어 설정을 실패했습니다.\n");

long    lHmDir = 0;
DWORD   lHmsig =  4;
DWORD   dwZphas = 0;
double  dwHClrTim = 2000.0, dwHOffset = 0.0;

AxmHomeSetMethod(0, lHmDir, lHmsig, dwZphas, dwHClrTim, dwHOffset);
AxmHomeSetVel(0,10000, 5000, 500, 50, 40000, 10000);
AxmHomeSetStart(0);
```

VB Example

```
If (AxmGantrySetEnable(0, 1, 1, 0#, 5#) = AXT_RT_SUCCESS) Then
     MsgBox "0축, 1축을 겐트리 제어로 설정했습니다", vbOKCancel
Else
     MsgBox "0축, 1축을 겐트리 제어 설정을 실패했습니다", vbOKCancel
End If

Dim dwHOffset As Double
Dim dwHClrTim As Double
Dim lHmDir, lHmsig, dwZphas As Long

dwHClrTim = 2000#
dwHOffset = 0#

lHmDir = 0
lHmsig = 4
dwZphas = 0
'(-)방향으로 원점검색, Home Sensor이용, Home Sensor Type A접점, Z상 검출 사용안함,
       Offset이동안함으로 설정

AxmHomeSetMethod 0, lHmDir, lHmsig, dwZphas, dwHClrTim, dwHOffset
AxmHomeSetVel 0, 10000, 5000, 500, 50, 40000, 10000
AxmHomeSetStart 0
```

Delphi Example

```
var
   dwZphas, lHmsig : DWORD;
   dwHOffset, dwHClrTim : Double;
   lHmDir : LongInt;
   velocity : Double;

begin

lHmDir := 0;
lHmsig :=  4;
dwZphas := 0;
dwHClrTim := 2000.0;
dwHOffset := 0.0;

if (AxmGantrySetEnable (0,1,1, 0.0, 5.0) =  AXT_RT_SUCCESS)  then
 Application.MessageBox('0축, 1축을 겐트리 제어로 설정했습니다.', 'Ajinextek',
       MB_OK)
else
 Application.MessageBox('0축, 1축을 겐트리 제어 설정을 실패했습니다.',
       'Ajinextek', MB_OK);

 AxmHomeSetMethod(0, lHmDir, lHmsig, dwZphas, dwHClrTim,dwHOffset);
 AxmHomeSetVel(0,10000, 5000, 500, 50, 40000, 10000);
AxmHomeSetStart(0);

End;
```

See Also

AxmGantrySetDisable, AxmGantryGetEnable, AxmHomeSetMethod, AxmHomeGetMethod, AxmHomeSetVel, AxmHomeGetVel, AxmHomeSetStart, AxmHomeSetResult, AxmHomeGetResult, AxmHomeGetRate

AxmGantryGetEnable

Purpose

이 함수를 이용해 Master축을 겐트리 제어로 설정할 때 설정한 파라메타를 반환한다.

Format

C

```
DWORD AxmGantryGetEnable (long lMasterAxisNo, DWORD *upSlHomeUse, double *dpSlOffset,
double *dpSlORange, DWORD *upGatryOn);
```

Visual Basic

```
Function AxmGantryGetEnable (ByVal lMasterAxisNo As Long, ByRef upSlHomeUse As Long,
ByRef dpSlOffset As Double, ByRef dpSlORange As Double, ByRef upGatryOn As Long) As Long
```

Delphi

```
function AxmGantryGetEnable (lMasterAxisNo : LongInt; upSlHomeUse : PDWord; dpSlOffset :
PDouble; dpSlORange : PDouble; upGatryOn : PDWord) : DWord; stdcall;
```

Input/Output

Name	in/out	Format	Init Value	Explanation
MasterAxisNo	in	long	0	Master채널(축) 번호(0 ~ (최대축수 - 1))
SlHomeUse	out	DWORD*	0	Slave축 원점 검색 여부를 선택 인자
SlOffset	out	double*	0	Master축의 원점센서와 Slave축 원점센서간의 기구적인 오차 값
SlOffsetRange	out	double*	0	원점검색시 Master축의 원점센서와 Slave축 원점센서간에 허용할 최대 오차값 지정
GatryOn	out	DWORD*	0	겐트리 ON/OFF 인자

uSlHomeUse

#define	Value	Explanation
DISABLE	00h	Master 축만 원점검색 선택
ENABLE	01h	Maser축 과 Slave축에 원점검색 같이 선택 확인

uGatryOn

#define	Value	Explanation

DISABLE	00h	겐트리 OFF
ENABLE	01h	겐트리 ON

Return AXT_RT_SUCCESS(0000) : Successful execution of API
* See error code Table more information on status error codes

Description

참고 : 필요한 데이터만 가지고 올 경우, 필요하지 않는 변수에 NULL 값을 넣어 사용 할 수 있다.

AxmGantrySetEnable 에서 설정한 Gantry 관련 설정 파라미터를 확인한다.
파라미터에 관한 자세한 설명은 AxmGantrySetEnable함수 설명 참조

C Example

```
double   dOffset;
DWORD   upHomeResult0;
double     dpSlOffset,dSlORange;
DWORD   bSlHomeUse, bGatryOn;
if(AxmGantrySetEnable (0, 1, TRUE, 0, 0.0) ==  AXT_RT_SUCCESS )
     printf("0축, 1축을 겐트리 제어로 설정했습니다.\n");
else
     printf("0축, 1축을 겐트리 제어 설정을 실패했습니다.\n");

long   lHmDir = 0;
DWORD   lHmsig =  4;
DWORD   dwZphas = 0;
double  dwHClrTim = 2000.0, dwHOffset = 0.0;

AxmHomeSetMethod(0, lHmDir, lHmsig, dwZphas, dwHClrTim, dwHOffset);
AxmHomeSetVel(0,10000, 5000, 500, 50, 40000, 10000);
AxmHomeSetStart(0);

AxmGantryGetEnable(0, &bSlHomeUse, &dpSlOffset, &dSlORange, &bGatryOn);
```

VB Example

```
Dim  dOffset , dSlORange, dpSlOffset As Double
Dim upHomeResult0, uID  As Long
Dim bSlHomeUse, bGatryOn As Long
Dim strData As String
If (AxmGantrySetEnable(0, 1, 1, 0#, 5#) = AXT_RT_SUCCESS) Then
     MsgBox "0축, 1축을 겐트리 제어로 설정했습니다", vbOKCancel
Else
     MsgBox "0축, 1축을 겐트리 제어 설정을 실패했습니다", vbOKCancel
End If

Dim dwHOffset As Double
Dim dwHClrTim As Double
```

```
Dim lHmDir, lHmsig, dwZphas As Long

dwHClrTim = 2000#
dwHOffset = 0#

lHmDir = 0
lHmsig = 4
dwZphas = 0
'(-)방향으로 원점검색, Home Sensor이용, Home Sensor Type A접점, Z상 검출 사용안함,
       Offset이동안함으로 설정

AxmHomeSetMethod 0, lHmDir, lHmsig, dwZphas, dwHClrTim, dwHOffset
AxmHomeSetVel 0, 10000, 5000, 500, 50, 40000, 10000
AxmHomeSetStart 0

AxmGantryGetEnable 0, bSlHomeUse, dpSlOffset, dSlORange, bGatryOn
```

Delphi Example

```
var
   dwZphas, lHmsig : DWORD;
   dwHOffset, dwHClrTim : Double;
   lHmDir : LongInt;
   velocity : Double;
   dOffset: Double;
   upHomeResult0, bMaSlChange, bSlHomeUse , bGatryOn :DWORD;
   dSlORange, dpSlOffset : Double;

begin

lHmDir := 0;
lHmsig :=  4;
dwZphas := 0;
dwHClrTim := 2000.0;
dwHOffset := 0.0;

if (AxmGantrySetEnable (0,1,1, 0.0, 5.0) =  AXT_RT_SUCCESS)  then
Application.MessageBox('0축, 1축을 겐트리 제어로 설정했습니다.', 'Ajinextek',
       MB_OK)
else
Application.MessageBox('0축, 1축을 겐트리 제어 설정을 실패했습니다.',
       'Ajinextek', MB_OK);

AxmHomeSetMethod(0, lHmDir, lHmsig, dwZphas, dwHClrTim,dwHOffset);
AxmHomeSetVel(0,10000, 5000, 500, 50, 40000, 10000);
AxmHomeSetStart(0);

AxmGantryGetEnable(0, @bSlHomeUse, @dpSlOffset, @dSlORange, @bGatryOn);

End;
```

See Also

AxmGantrySetEnable, AxmGantrySetDisable, AxmHomeSetMethod, AxmHomeGetMethod, AxmHomeSetVel, AxmHomeGetVel, AxmHomeSetStart, AxmHomeSetResult, AxmHomeGetResult, AxmHomeGetRate

AxmGantrySetDisable

Purpose

두 축이 기구적으로 Link되어있는 겐트리 구동시스템 제어를 해제한다.

Format

C

DWORD AxmGantrySetDisable (long lMasterAxisNo, long lSlaveAxisNo);

Visual Basic

Function AxmGantrySetDisable (ByVal lMasterAxisNo As Long, ByVal lSlaveAxisNo As Long) As Long

Delphi

function AxmGantrySetDisable (lMasterAxisNo : LongInt; lSlaveAxisNo : LongInt) : DWord; stdcall;

Input/Output

Name	in/out	Format	Init Value	Explanation
MasterAxisNo	in	long	0	Master채널(축) 번호(0 ~ (최대축수 – 1))
SlaveAxisNo	in	long	0	Slave채널(축) 번호(0 ~ (최대축수 – 1))

Return AXT_RT_SUCCESS(0000) : Successful execution of API

* See error code Table more information on status error codes

Description

'AxmGantrySetEnable' 설정된 Link되어있는 겐트리 구동시스템 제어를 해제한다

C Example

```
if(AxmGantrySetEnable (0, 1, TRUE, 0, 0.0) ==  AXT_RT_SUCCESS )
     printf("0축, 1축을 겐트리 제어로 설정했습니다.\n");
else
     printf("0축, 1축을 겐트리 제어 설정을 실패했습니다.\n");

// 겐트리 구동시스템을 해제시킨다..
AxmGantrySetDisable (0, 1);
```

VB Example

```
If (AxmGantrySetEnable(0, 1, 1, 0#, 5#) = AXT_RT_SUCCESS) Then
     MsgBox “0축, 1축을 겐트리 제어로 설정했습니다”, vbOKCancel
Else
     MsgBox “0축, 1축을 겐트리 제어 설정을 실패했습니다”, vbOKCancel
End If

AxmGantrySetDisable 0, 1
```

Delphi Example

```
begin

if (AxmGantrySetEnable (0,1,1, 0.0, 5.0) =  AXT_RT_SUCCESS)  then
    Application.MessageBox(‘0축, 1축을 겐트리 제어로 설정했습니다.’,
       ‘Ajinextek’, MB_OK)
else
    Application.MessageBox(‘0축, 1축을 겐트리 제어 설정을 실패했습니다.’,
       ‘Ajinextek’, MB_OK);

AxmGantrySetDisable (0, 1);

End;
```

See Also

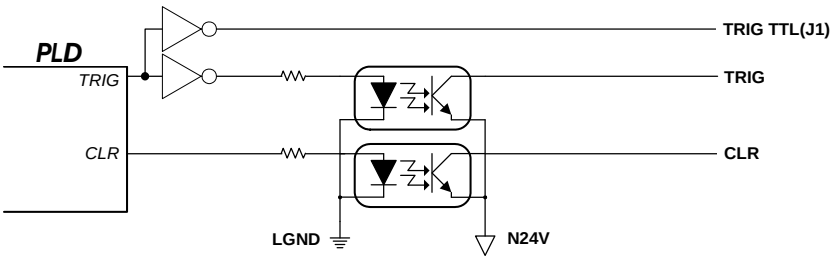
AxmGantrySetEnable, AxmGantryGetEnable, AxmHomeSetMethod, AxmHomeGetMethod, AxmHomeSetVel, AxmHomeGetVel, AxmHomeSetStart, AxmHomeSetResult, AxmHomeGetResult, AxmHomeGetRate

CRC(잔여 펄스 클리어) 함수

Function	Description
AxmCrcSetMaskLevel	지정 축에 CRC 신호 사용 여부 및 출력 레벨을 설정한다.
AxmCrcGetMaskLevel	지정 축의 CRC 신호 사용 여부 및 출력 레벨을 반환한다.
AxmCrcSetOutput	지정 축에 CRC 신호를 강제로 발생 시킨다.
AxmCrcGetOutput	지정 축의 CRC 신호를 강제로 발생 여부를 반환한다.
AxmCrcSetEndLimit	지정 축에 리미트에 대한 CRC 신호의 사용 여부 및 출력 레벨을 설정한다.
AxmCrcGetEndLimit	지정 축의 리미트에 대한 CRC 신호의 사용 여부 및 출력 레벨을 반환한다.

CRC 신호 설정

특정 서보팩의 경우 구동이 완료된 시점에서 또는 리미트센서를 감지한 경우에 외부에서 CRC(Current Remaining Clear)신호를 받아 서보팩이 가지고 있는 잔여 펄스를 지워주어야 하는 경우가 발생한다. 이를 위해 신호핀을 할당하여 서보팩에 직접 연결할 수 있도록 제공하고 있으며, CRC신호의 사용여부, 출력레벨, 리미트 신호 사용여부 등을 설정할 수 있도록 함수로 제공하고 있다. 또한 임의로 사용자가 CRC신호를 발생시킬 수도 있도록 함수를 제공한다.
출력 레벨은 칩 출력의 반전된 레벨이 나오게 된다. 즉, 칩 출력 레벨이 논리 '1'일 때, CLR출력은 N24V로, 논리 '0'일 때의 출력은 Open Collector이다. 이 신호 모두 범용출력과 마찬가지로 Photo-Coupler로 외부 신호와 전기적으로 절연되어 있다. 이 신호들은 외부의 특정 위치 또는 외부의 특정 조건에 의해 출력을 내게 되는데, 시간 지연은 Photo-Coupler를 거치는 시간과, PLD내부 회로의 시간 지연에 의한 것이다. PELM, NELM을 입력으로 받는 CLR신호의 경우 약 6㎲~8시간지연이 발생된다. Photo Coupler 출력은 Normally Open 형태의 출력이므로, 24V레벨의 전압을 얻기 위해서는 사용자는 Pull-up저항을 붙여야 한다. Pull-up저항은 약 10㎏의 저항을 사용한다.



AxmCrcSetMaskLevel

Purpose

지정 축에 CRC 신호 사용 여부를 설정한다.

Format

C

```
DWORD AxmCrcSetMaskLevel (long lAxisNo, DWORD uLevel, DWORD uMethod);
```

Visual Basic

```
Function AxmCrcSetMaskLevel (ByVal lAxisNo As Long, ByVal uLevel As Long, ByVal lMethod As Long) As Long
```

Delphi

```
function AxmCrcSetMaskLevel  (lAxisNo : LongInt; uLevel : Dword; lMethod : Dword) : DWord; stdcall;
```

Input/Output

Name	in/out	Format	Init Value	Explanation
AxisNo	in	long	-	채널(축) 번호(0 ~ (최대축수 - 1)
Level	In	DWORD	-	CRC 신호 사용 여부
Method	In	DWORD	0	잔여펄스 제거 출력 신호 펄스 폭

Level

#define	Value	Explanation
LOW	00h	B 접점(NORMAL CLOSE), Active Low Level CRC 신호 사용
HIGH	01h	A 접점(NORMAL OPEN), Active High Level CRC 신호 사용
UNUSED	02h	CRC 신호 사용안함
USED	03h	CRC 신호 현상태유지

Method

Value	Explanation
00h	사용 하지 않음
01h	사용 하지 않음
02h	500 uSec

03h	1 mSec
04h	10 mSec
05h	50 mSec
06h	100 mSec

Return AXT_RT_SUCCESS(0000) : Successful execution of API
* See error code Table more information on status error codes

Description

CRC Enable을 시키기위해서는 AxmCrcSetEndLimit 함수를 사용한다.

* 이 신호는 센서가 아님
Low(B접점) : 평상시 Close
High(A접점): 평상시 Open

C Example

```
// 0축에 CRC 신호 사용 여부를 설정
AxmCrcSetMaskLevel (0, LOW, 0);

// 0축에 CRC 신호 사용 여부를 확인
DWORD uLevel, uMethod;
AxmCrcGetMaskLevel (0, &uLevel, &uMethod);
```

VB Example

```
' 0축에 CRC 신호 사용 여부를 설정
AxmCrcSetMaskLevel 0, LOW, 0

' 0축에 CRC 신호 사용 여부를 확인
Dim uLevel As Long
Dim uMethod As Long
AxmCrcGetMaskLevel 0, uLevel, uMethod
```

Delphi Example

```
{ 0축에 CRC 신호 사용 여부를 확인 }
var
uLevel: DWord;
uMethod: DWord;

Begin
{ 0축에 CRC 신호 사용 여부를 설정 }
AxmCrcSetMaskLevel (0, LOW, 0);
```

```
AxmCrcGetMaskLevel (0, @uLevel, @uMethod);
end;
```

See Also

AxmCrcGetMaskLevel, AxmCrcSetOutput, AxmCrcGetOutput, AxmCrcSetEndLimit, AxmCrcGetEndLimit

AxmCrcGetMaskLevel

Purpose

지정 축에 CRC 신호 사용 여부 설정을 반환 한다.

Format

C

```
DWORD AxmCrcGetMaskLevel (long lAxisNo, DWORD *upLevel, DWORD *upMethod);
```

Visual Basic

```
Function AxmCrcGetMaskLevel (ByVal lAxisNo As Long, ByRef upLevel As Long, ByVal lMethod As Long) As Long
```

Delphi

```
function AxmCrcGetMaskLevel (lAxisNo : LongInt; upLevel : PDWord; upMethod : PDword) : DWord; stdcall;
```

Input/Output

Name	in/out	Format	Init Value	Explanation
AxisNo	in	long	-	채널(축) 번호(0 ~ (최대축수 - 1))
Level	Out	DWORD*	-	CRC 신호 사용 여부
Method	Out	DWORD*	0	잔여펄스 제거 출력 신호 펄스 폭

Level

#define	Value	Explanation
LOW	00h	B 접점(NORMAL CLOSE) Active Low Level CRC 신호 사용
HIGH	01h	A 접점(NORMAL OPEN) Active High Level CRC 신호 사용
UNUSED	02h	CRC 신호 사용안함
USED	03h	CRC 신호 현상태유지

Method

Value	Explanation
00h	사용 하지 않음

01h	사용 하지 않음
02h	500 uSec
03h	1 mSec
04h	10 mSec
05h	50 mSec
06h	100 mSec

Return AXT_RT_SUCCESS(0000) : Successful execution of API
* See error code Table more information on status error codes

Description

'AxmCrcSetMaskLevel' 로 설정한 지정한 축의 CRC 신호 사용 여부 또는 신호 Active Level을 설정을 반환 한다.

* 이 신호는 센서가 아님
Low(B접점) : 평상시 Close
High(A점점): 평상시 Open

C Example

```
// 0축에 CRC 신호 사용 여부를 설정
AxmCrcSetMaskLevel (0, LOW, 0);

// 0축에 CRC 신호 사용 여부를 확인
DWORD uLevel, uMethod;
AxmCrcGetMaskLevel (0, &uLevel, &uMethod);
```

VB Example

```
' 0축에 CRC 신호 사용 여부를 설정
AxmCrcSetMaskLevel 0, LOW, 0

' 0축에 CRC 신호 사용 여부를 확인
Dim uLevel As Long
Dim uMethod As Long
AxmCrcGetMaskLevel 0, uLevel, uMethod
```

Delphi Example

```
{ 0축에 CRC 신호 사용 여부를 확인 }
var
uLevel: DWord;
```

```
uMethod: DWord;

Begin
{ 0축에 CRC 신호 사용 여부를 설정 }
AxmCrcSetMaskLevel (0, LOW, 0);

AxmCrcGetMaskLevel (0, @uLevel, @uMethod);
end;
```

See Also

AxmCrcSetMaskLevel, AxmCrcSetOutput, AxmCrcGetOutput, AxmCrcSetEndLimit, AxmCrcGetEndLimit

AxmCrcSetOutput

Purpose

지정 축에 CRC 신호를 강제적으로 출력을 발생시킨다.

Format

C

```
DWORD AxmCrcSetOutput (long lAxisNo, DWORD uOnOff);
```

Visual Basic

```
Function AxmCrcSetOutput (ByVal lAxisNo As Long, ByVal uOnOff As Long) As Long
```

Delphi

```
function AxmCrcSetOutput (lAxisNo : LongInt; uOnOff : DWord) : DWord; stdcall;
```

Input/Output

Name	in/out	Format	Init Value	Explanation
AxisNo	in	long	-	채널(축) 번호(0 ~ (최대축수 - 1))
OnOff	In	DWORD	-	CRC 신호를 Program으로 발생 여부

OnOff

#define	Value	Explanation
FALSE	00h	CRC 신호 Off
TRUE	01h	CRC 신호 On

Return AXT_RT_SUCCESS(0000) : Successful execution of API

* See error code Table more information on status error codes

Description

참고로 EIP-M2/4Q에서 CRC Enable을 시키기위해서는 AxmCrcSetEndLimit 함수를 사용한다.

사용자가 원할 때 지정한 axis축에 CRC 신호를 발생 시킨다.

* 이 신호는 센서가 아님
Low(B접점) : 평상시 Close
High(A점점): 평상시 Open

C Example

```
// 0축에 CRC 신호를 Program으로 발생 시킨다
AxmCrcSetOutput (0, ENABLE);
```

VB Example

```
‘ 0축에 CRC 신호를 Program으로 발생 시킨다
AxmCrcSetOutput 0, ENABLE
```

Delphi Example

```
begin
{ 0축에 CRC 신호를 Program으로 발생 시킨다 }
AxmCrcSetOutput (0, ENABLE);
End;
```

See Also

AxmCrcSetMaskLevel, AxmCrcGetMaskLevel, AxmCrcSetEndLimit, AxmCrcGetEndLimit

AxmCrcGetOutput

Purpose

지정 축에 CRC 신호를 강제적으로 출력 발생 여부를 반환한다.

Format

C

```
DWORD AxmCrcGetOutput (long lAxisNo, DWORD *upOnOff);
```

Visual Basic

```
Function AxmCrcGetOutput (ByVal lAxisNo As Long, ByRef upOnOff As Long) As Long
```

Delphi

```
function AxmCrcGetOutput (lAxisNo : LongInt; upOnOff : PDWord) : DWord; stdcall;
```

Input/Output

Name	in/out	Format	Init Value	Explanation
AxisNo	in	Long	-	채널(축) 번호(0 ~ (최대축수 - 1))
OnOff	out	DWORD*	-	CRC 신호를 Program으로 발생 여부

OnOff

#define	Value	Explanation
FALSE	00h	CRC 신호 Off
TRUE	01h	CRC 신호 On

Return AXT_RT_SUCCESS(0000) : Successful execution of API

* See error code Table more information on status error codes

Description

'AxmCrcSetOutput' 로 설정한 지정 축에 CRC 신호를 강제적으로 출력 발생 여부를 반환한다.

*이 신호는 센서가 아님
Low(B접점) : 평상시 Close
High(A점점): 평상시 Open

C Example

```
// 0축에 CRC 신호를 Program으로 발생 시킨다
AxmCrcSetOutput (0, ENABLE);

// 0축의 CRC 신호를 Program으로 발생 여부를 확인
DWORD uUse;
AxmCrcGetOutput (0, &uUse);
```

VB Example

```
' 0축에 CRC 신호를 Program으로 발생 시킨다
AxmCrcSetOutput 0, ENABLE

' 0축의 CRC 신호를 Program으로 발생 여부를 확인
Dim uUse As Long
AxmCrcGetOutput 0, uUse
```

Delphi Example

```
var
uUse: DWord;

begin
{ 0축에 CRC 신호를 Program으로 발생 시킨다 }
AxmCrcSetOutput (0, ENABLE);
AxmCrcGetOutput (0, @uUse);
end;
```

See Also

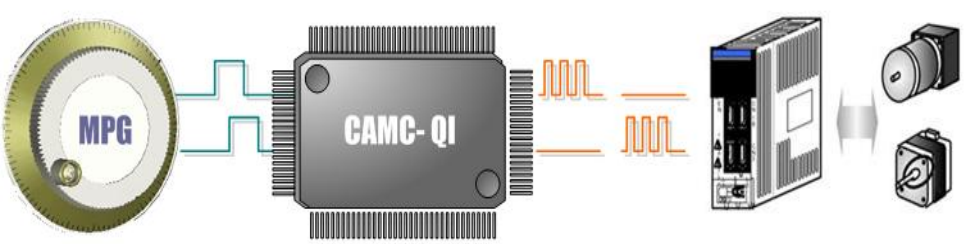
AxmCrcSetMaskLevel, AxmCrcGetMaskLevel, AxmCrcSetOutput, AxmCrcSetEndLimit, AxmCrcGetEndLimit

MPG 구동 함수

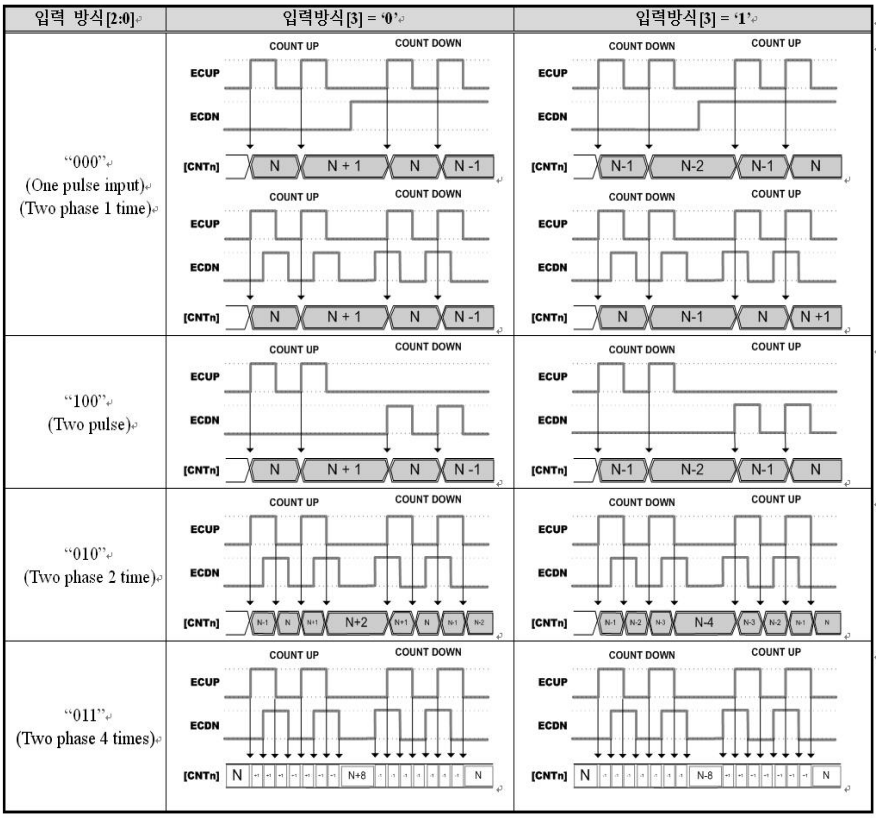
Function	Description.
AxmMPGSetEnable	지정 축에 MPG 입력방식, 드라이브 구동 모드, 이동 거리 등을 설정한다.
AxmMPGGetEnable	지정 축에 MPG 입력방식, 드라이브 구동 모드, 이동 거리 등을 반환한다.
AxmMPGSetRatio	지정 축에 MPG 드라이브 구동 모드에서 한펄스당 이동할 펄스 비율을 설정한다.
AxmMPGGetRatio	지정 축에 MPG 드라이브 구동 모드에서 한펄스당 이동할 펄스 비율을 반환한다.
AxmMPGReset	지정 축에 MPG 드라이브 설정을 해지한다.

EXPP, EXMP 신호의 입력에 따라 방향 신호가 결정되고, 입력 사양 설정 레지스터에 따라 펄스 수가 결정된다. 출력되는 펄스의 속도는 설정된 Object Speed에 의해 출력된다

EIP-M2/4Q

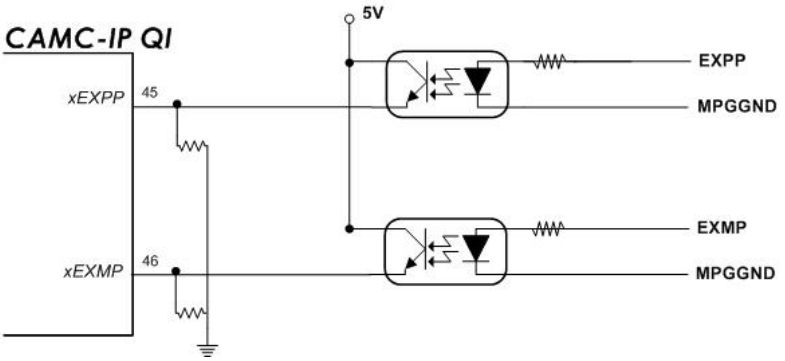


입력 펄스형태

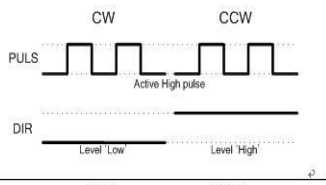
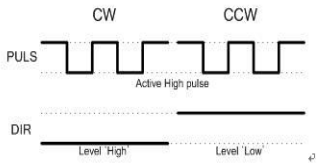
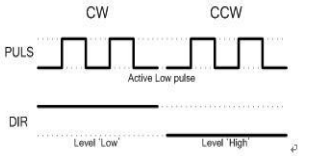
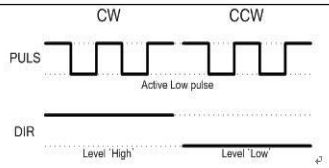
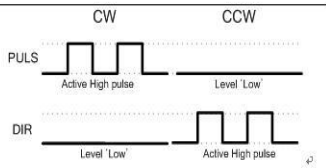
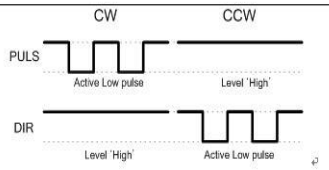
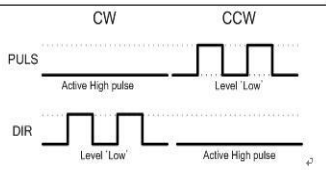
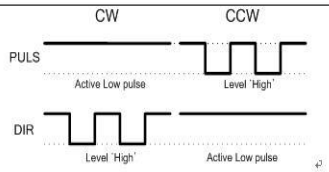
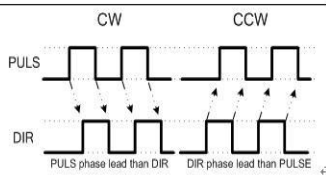
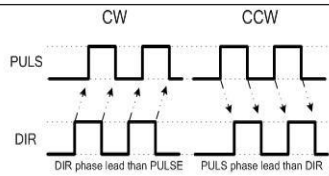


● *MPG 외부와의 인터페이스*

EXPP, EXMP 두 신호는 EIP-M2/4Q의 외부 로타리 인코더의 입력 신호로 이용되며, 최대 1.2MHz 고속 포토커플러를 통하여 EIP-M2/4Q칩과 연결되어 있다.



MPG 입력 신호는 각 축에 독립적으로 입력되는 것이 아니고 두축이 공동으로 사용하는데 0축에 입력되는 MPG신호는 2축, 1축에 입력되는 MPG신호는 3축, 4축에 입력되는 MPG신호는 6축, 5축에 입력되는 MPG신호는 7축에서 공통으로 사용한다. 즉 2/3/6/7축과 연결되는 외부 컨넥트 J2/J4의 MPG신호 입력핀은 사용되지 않는다.

[UCFG3][2~0]	[UCFG3][3] = '0'	[UCFG3][3] = '1'
"000" (One pulse mode)		
"001" (One pulse mode)		
"010" (Two pulse mode)		
"011" (Two pulse mode)		
"100" (Two phase mode)		

AxmMPGSetEnable

Purpose

지정 축에 MPG 입력방식, 드라이브 구동 모드, 이동 거리 , 이동속도 등을 설정한다.

Format

C

```
DWORD AxmMPGSetEnable (long lAxisNo, long lInputMethod, long lDriveMode, double dMPGPos ,
double dVel, double dAccel);
```

Visual Basic

```
Function AxmMPGSetEnable Lib "AxI.dll" (ByVal lAxisNo As Long, ByVal lInputMethod As Long,
ByVal lDriveMode As Long, ByVal dMPGPos As Double, ByVal dVel As Double, ByVal dAccel As
Double) As Long
```

Delphi

```
function AxmMPGSetEnable (lAxisNo : LongInt; lInputMethod : LongInt; lDriveMode : LongInt;
dMPGPos : Double; dVel: Double; dAccel: Double) : DWord; stdcall;
```

Input/ Output

Name	in/out	Format	Init Value	Explanation
AxisNo	in	long	-	채널(축) 번호(0 ~ (최대축수 - 1))
InputMethod	In	long	-	입력 신호 방식
DriveMode	In	long	-	Mpg 동작 모드
MPGPos	In	double	-	MPG_PRESET_MODE 모드일경우 이동거리
dVel	In	double	-	Mpg 속도
dAccel	In	double	-	Mpg 가속도(EIP-M2/4Q는 지원안함)

InputMethod

#define	Value	Explanation
MPG_DIFF_ONE_PHASE	00h	MPG 입력 방식 One Phase
MPG_DIFF_TWO_PHASE_1X	01h	EIP-M2/4Q는 지원안함
MPG_DIFF_TWO_PHASE_2X	02h	MPG 입력 방식 TwoPhase2
MPG_DIFF_TWO_PHASE_4X	03h	MPG 입력 방식 TwoPhase4

DriveMode

Value	Explanation
00h	MPG 연속모드

Return AXT_RT_SUCCESS(0000) : Successful execution of API
* See error code Table more information on status error codes

Description

지정 축에 MPG 드라이브 지정 펄스 구동 모드에서 사용할 이동 거리, 구동모드 및 구동 방향 등을 설정한다.

여기서 dMPGdenominator = 4095, MPGnumerator=0 가 의미하는 것은 MPG 한바퀴에 200펄스면 그대로 1:1로 1펄스씩 출력을 의미한다. 만약 dMPGdenominator = 4095, MPGnumerator=1 로 했을 경우는 1:2로 2펄스씩 출력을 내보낸다는의미이다.

여기에 MPG PULSE = ((Numerator) * (Denominator)/ 4096) 칩내부에 출력나가는 계산식이다.

제품에 따라 AxmMPGSetEnable 수행 후 AxmStatusReadInMotion 함수 실행 결과가 다르다.

AxmMPGReset 하기 전까지 정상 상태에서는 모션 구동 중 상태.

C Example

```
// 0축의 Mpg 신호를 설정한다.
AxmMPGSetEnable (0, MPG_DIFF_ONE_PHASE, 1, 10, 50);
AxmMPGSetRatio(0, 1, 4096 );

// 0축의 Mpg 신호를 설정을 반환한다.
long lpInputMethod, lpDriveMode;
double dpVel, dpAccel, dpDecel, dpMPGPos;
double dMPGdenominator, dMPGnumerator, dVel;
AxmMPGGetEnable (0, &lpInputMethod, & lpDriveMode, &dpMPGPos, &dVel);
AxmMPGGetRatio(0, &dMPGnumerator, &dMPGdenominator);
```

VB Example

```
Dim dVelocity As Double
Dim dAccel As Double
Dim dDecel As Double
Dim dMPGdenominator As Double
Dim dMPGnumerator, dMPGPosition, dVel As Double
Dim lInputMethod, lDriveMode, lDirMode, lUserDir As Long

' 0축의 Mpg 신호를 설정한다.
AxmMPGSetEnable 0, MPG_DIFF_ONE_PHASE, 1, 10, 50
AxmMPGSetRatio 0, 1, 4096

' 0축의 Mpg 신호를 설정을 반환한다.
AxmMPGGetEnable 0, lInputMethod, lDriveMode, dMPGPosition, dVel
AxmMPGGetRatio 0, dMPGnumerator, dMPGdenominator
```

Delphi Example

```
{ 0축의 Mpg 신호를 설정을 반환한다. }
var
dVelocity : Double;
dAccel  : Double;
dDecel  : Double;
dMPGdenominator        : Double;
dMPGnumerator          : Double;
lpInputMethod,lpDriveMode : longInt;
dMPGPosition, dVel : Double;

begin

// 0축의 Mpg 신호를 설정한다.
AxmMPGSetEnable (0, MPG_DIFF_ONE_PHASE, 1, 10, 50);
AxmMPGSetRatio(0, 1, 4096  );

// 0축의 Mpg 신호를 설정을 반환한다.
AxmMPGGetEnable (0, @lpInputMethod, @lpDriveMode, @dMPGPosition);
AxmMPGGetRatio(0, @dMPGnumerator, @dMPGdenominator, @dVel);
end;
```

See Also

AxmMPGGetEnable, AxmMPGSetRatio, AxmMPGGetRatio, AxmMPGReset

AxmMPGGetEnable

Purpose

지정 축에 MPG 입력방식, 드라이브 구동 모드, 이동 거리 등을 반환한다.

Format

C

```
DWORD AxmMPGGetEnable (long lAxisNo, long *lpInputMethod, long *lpDriveMode, double
*dpMPGPos, double *dAccel);
```

Visual Basic

```
Function AxmMPGGetEnable Lib "Axl.dll" (ByVal lAxisNo As Long, ByRef lpInputMethod As Long,
ByRef lpDriveMode As Long, ByRef dpMPGPos As Double, ByRef dAccel As Double) As Long
```

Delphi

```
function AxmMPGGetEnable (lAxisNo : LongInt; lpInputMethod : PLongInt; lpDriveMode : PLongInt;
dpMPGPos : PDouble; dAccel: PDouble) : DWord; stdcall;
```

Input/ Output

Name	in/out	Format	Init Value	Explanation
AxisNo	in	long	-	채널(축) 번호(0 ~ (최대축수 - 1))
InputMethod	In	long	-	입력 신호 방식 (MPG_DIFF_TWO_PHASE_1X지원안함)
DriveMode	In	long	-	MPG 동작 모드
MPGPos	In	double	-	MPG_PRESET_MODE 모드일경우 이동거리
dpVel	In	double	-	속도

Return AXT_RT_SUCCESS(0000) : Successful execution of API

* See error code Table more information on status error codes

Description

'AxmMPGSetEnable ' 로 설정된 지정 축에 MPG 입력방식, 드라이브 구동모드, 이동거리 등을 반환한다.

C Example

```
// 0축의 Mpg 신호를 설정한다.
AxmMPGSetEnable (0, MPG_DIFF_ONE_PHASE, 1, 10, 50);
AxmMPGSetRatio(0, 1, 4096  );

// 0축의 Mpg 신호를 설정을 반환한다.
long lpInputMethod, lpDriveMode;
double dpVel, dpAccel, dpDecel, dpMPGPos;
double dMPGdenominator, dMPGnumerator, dVel;

AxmMPGGetEnable (0, &lpInputMethod, & lpDriveMode, &dpMPGPos, &dVel);
AxmMPGGetRatio(0, &dMPGnumerator, &dMPGdenominator);
```

VB Example

```
Dim dVelocity As Double
Dim dAccel As Double
Dim dDecel As Double
Dim dMPGdenominator As Double
Dim dMPGnumerator, dMPGPosition, dVel As Double
Dim lInputMethod, lDriveMode, lDirMode, lUserDir As Long

' 0축의 Mpg 신호를 설정한다.
AxmMPGSetEnable 0, MPG_DIFF_ONE_PHASE, 1, 10, 50
AxmMPGSetRatio 0, 1, 4096

' 0축의 Mpg 신호를 설정을 반환한다.
AxmMPGGetEnable 0, lInputMethod, lDriveMode, dMPGPosition, dVel
AxmMPGGetRatio 0, dMPGnumerator, dMPGdenominator
```

Delphi Example

```
{ 0축의 Mpg 신호를 설정을 반환한다. }
var
dVelocity : Double;
dAccel  : Double;
dDecel  : Double;
dMPGdenominator        : Double;
dMPGnumerator          : Double;
lpInputMethod,lpDriveMode : longInt;
dMPGPosition, dVel : Double;

begin

// 0축의 Mpg 신호를 설정한다.
AxmMPGSetEnable (0, MPG_DIFF_ONE_PHASE, 1, 10, 50);
AxmMPGSetRatio(0, 1, 4096  );

// 0축의 Mpg 신호를 설정을 반환한다.
AxmMPGGetEnable (0, @lpInputMethod, @lpDriveMode, @dMPGPosition, @dVel);
AxmMPGGetRatio(0, @dMPGnumerator, @dMPGdenominator);
```

```
end;
```

See Also

AxmMPGSetEnable, AxmMPGSetRatio, AxmMPGGetRatio, AxmMPGReset

AxmMPGSetRatio

Purpose

지정 축에 MPG 드라이브 구동 모드에서 한펄스당 이동할 펄스 비율을 설정한다.

Format

C

```
DWORD AxmMPGSetRatio(long lAxisNo, double dMPGnumerator, double dMPGdenominator);
```

Visual Basic

```
Function AxmMPGSetRatio Lib "Axl.dll" (ByVal lAxisNo As Long, ByVal dMPGnumerator As Double,
ByVal dMPGdenominator As Double) As Long
```

Delphi

```
function AxmMPGSetRatio (lAxisNo : LongInt; dMPGnumerator : Double; dMPGdenominator :
Double) : DWord; stdcall;
```

Input/ Output

Name	in/out	Format	Init Value	Explanation
AxisNo	in	long	-	채널(축) 번호(0 ~ (최대축수 - 1))
MPGnumerator	In	double	-	수동 펄스 발생 장치 입력)구동 시 곱하기 값 (입력 값 계산됨) 최대(1 ~ 64) 까지 설정 가능 MPG PULSE = (dwNumerator)* (dwDenominator)/ 4096
MPGdenominator	In	double	-	(수동 펄스 발생 장치 입력)구동 시 나누기 값(입력 값 계산됨) 최대(1 ~ 4096) 까지 설정 가능

Return AXT_RT_SUCCESS(0000) : Successful execution of API

* See error code Table more information on status error codes

Description

지정 축에 MPG 드라이브 지정 펄스 구동 모드에서 사용할 이동 거리, 구동모드 및 구동 방향 등을 설정한다.

여기서 dMPGdenominator = 4096, MPGnumerator=1 의미하는 것은 MPG 한바퀴에 200펄스면 그대로 1:1로 1펄스씩 출력을 의미한다. 만약 dMPGdenominator = 4096, MPGnumerator=2 로 했을경우는 1:2로 2펄스씩 출력을 내보낸다는의미이다.

여기에 MPG PULSE = ((Numerator) * (Denominator)/ 4096) 칩내부에 출력나가는 계산식이다.

C Example

```
// 0축의 Mpg 신호를 설정한다.
AxmMPGSetEnable (0, MPG_DIFF_ONE_PHASE, 1, 10, 50);
AxmMPGSetRatio(0, 1, 4096  );

// 0축의 Mpg 신호를 설정을 반환한다.
long lpInputMethod, lpDriveMode;
double dpVel, dpAccel, dpDecel, dpMPGPos;
double dMPGdenominator, dMPGnumerator, dVel;

AxmMPGGetEnable (0, &lpInputMethod, & lpDriveMode, &dpMPGPos, &dVel);
AxmMPGGetRatio(0, &dMPGnumerator, &dMPGdenominator);
```

VB Example

```
Dim dVelocity As Double
Dim dAccel As Double
Dim dDecel As Double
Dim dMPGdenominator As Double
Dim dMPGnumerator, dMPGPosition, dVel As Double
Dim lInputMethod, lDriveMode, lDirMode, lUserDir As Long

' 0축의 Mpg 신호를 설정한다.
AxmMPGSetEnable 0, MPG_DIFF_ONE_PHASE, 1, 10, 50
AxmMPGSetRatio 0, 1, 4096

' 0축의 Mpg 신호를 설정을 반환한다.
AxmMPGGetEnable 0, lInputMethod, lDriveMode, dMPGPosition, dVel
AxmMPGGetRatio 0, dMPGnumerator, dMPGdenominator
```

Delphi Example

```
{ 0축의 Mpg 신호를 설정을 반환한다. }
var
dVelocity : Double;
dAccel   : Double;
dDecel   : Double;
dMPGdenominator         : Double;
dMPGnumerator           : Double;
lpInputMethod,lpDriveMode : longInt;
dMPGPosition, dVel : Double;

begin
```

```
// 0축의 Mpg 신호를 설정한다.
AxmMPGSetEnable (0, MPG_DIFF_ONE_PHASE, 1, 10, 50);
AxmMPGSetRatio(0, 1, 4096  );

// 0축의 Mpg 신호를 설정을 반환한다.
AxmMPGGetEnable (0, @lpInputMethod, @lpDriveMode, @dMPGPosition, @dVel);
AxmMPGGetRatio(0, @dMPGnumerator, @dMPGdenominator);

end;
```

See Also

AxmMPGSetEnable, AxmMPGGetEnable, AxmMPGGetRatio, AxmMPGReset

AxmMPGGetRatio

Purpose

지정 축에 MPG 드라이브 구동 모드에서 한펄스당 이동할 펄스 비율을 반환한다.

Format

C

```
DWORD AxmMPGGetRatio(long lAxisNo, DWORD *upMPGnumerator, DWORD
*upMPGdenominator);
```

Visual Basic

```
Function AxmMPGGetRatio Lib "Axl.dll" (ByVal lAxisNo As Long, ByRef dpMPGnumerator As
Double, ByRef dpMPGdenominator As Double) As Long
```

Delphi

```
function AxmMPGGetRatio (lAxisNo : LongInt; dpMPGnumerator : PDouble; dpMPGdenominator :
PDouble) : DWord; stdcall;
```

Input/ Output

Name	in/out	Format	Init Value	Explanation
AxisNo	in	long	-	채널(축) 번호(0 ~ (최대축수 - 1))
MPGnumerator	In	double	-	수동 펄스 발생 장치 입력)구동 시 곱하기 값 (입력 값 계산됨) 최대(0 ~ 64) 까지 설정 가능 MPG PULSE = (dwNumerator)* (dwDenominator)/ 4096
MPGdenominator	In	double	-	(수동 펄스 발생 장치 입력)구동 시 나누기 값(입력 값로 계산됨) 최대(0 ~ 4096) 까지 설정 가능 단

Return AXT_RT_SUCCESS(0000) : Successful execution of API

* See error code Table more information on status error codes

Description

'AxmMPGSetRatio ' 로 설정된 지정 축에 MPG 구동 모드에서 한펄스당 이동할 펄스 비율을 반환한

다.

C Example

```
// 0축의 Mpg 신호를 설정한다.
AxmMPGSetEnable (0, MPG_DIFF_ONE_PHASE, 1, 10, 50);
AxmMPGSetRatio(0, 1, 4096  );

// 0축의 Mpg 신호를 설정을 반환한다.
long lpInputMethod, lpDriveMode;
double dpVel, dpAccel, dpDecel, dpMPGPos;
double dMPGdenominator, dMPGnumerator;
AxmMPGGetEnable (0, &lpInputMethod, & lpDriveMode, &dpMPGPos, &dVel);
AxmMPGGetRatio(0, &dMPGnumerator, &dMPGdenominator);
```

VB Example

```
Dim dVelocity As Double
Dim dAccel As Double
Dim dDecel As Double
Dim dMPGdenominator As Double
Dim dMPGnumerator, dMPGPosition, dVel As Double
Dim lInputMethod, lDriveMode, lDirMode, lUserDir As Long

' 0축의 Mpg 신호를 설정한다.
AxmMPGSetEnable 0, MPG_DIFF_ONE_PHASE, 1, 10, 50
AxmMPGSetRatio 0, 1, 4096

' 0축의 Mpg 신호를 설정을 반환한다.
AxmMPGGetEnable 0, lInputMethod, lDriveMode, dMPGPosition, dVel
AxmMPGGetRatio 0, dMPGnumerator, dMPGdenominator
```

Delphi Example

```
{ 0축의 Mpg 신호를 설정을 반환한다. }
var
dVelocity : Double;
dAccel  : Double;
dDecel  : Double;
dMPGdenominator        : Double;
dMPGnumerator          : Double;
lpInputMethod,lpDriveMode : longInt;
dMPGPosition, dVel : Double;

begin

// 0축의 Mpg 신호를 설정한다.
AxmMPGSetEnable (0, MPG_DIFF_ONE_PHASE, 1, 10, 50);
AxmMPGSetRatio(0, 1, 4096  );

// 0축의 Mpg 신호를 설정을 반환한다.
```

```
AxmMPGGetEnable (0, @lpInputMethod, @lpDriveMode, @dMPGPosition, @dVel);
AxmMPGGetRatio(0, @dMPGnumerator, @dMPGdenominator);

end;
```

See Also

AxmMPGSetEnable, AxmMPGGetEnable, AxmMPGGetRatio, AxmMPGReset

AxmMPGReset

Purpose

지정 축에 MPG 드라이브 설정을 해지한다.

Format

C

DWORD AxmMPGReset (long lAxisNo);

Visual Basic

Function AxmMPGReset (ByVal lAxisNo As Long) As Long

Delphi

function AxmMPGReset (lAxisNo : LongInt) : DWord; stdcall;

Input/ Output

Name	in/out	Format	Init Value	Explanation
AxisNo	in	long	-	채널(축) 번호(0 ~ (최대축수 - 1)))

Return AXT_RT_SUCCESS(0000) : Successful execution of API

* See error code Table more information on status error codes

Description

'AxmMPGSetEnable' 로 설정된 지정 축에 MPG 드라이브 지정 펄스 구동 모드에서 사용할 이동 거리, 구동모드 및 구동 방향 등 MPG 드라이브 설정을 해지한다.

C Example

```
// 0축의 MPG 드라이브 설정을 해지한다.
AxmMPGReset(0);
```

VB Example

```
' 0축의 MPG 드라이브 설정을 해지한다.
AxmMPGReset 0
```

Delphi Example

```
begin
{0축의 MPG 드라이브 설정을 해지한다.}
AxmMPGReset(0);
End;
```

See Also

AxmMPGSetEnable, AxmMPGGetEnable, AxmMPGSetRatio, AxmMPGGetRatio,

네트웍크 상태에 대한 모션 구동 에러 처리

Function	Description	비고
AxmNetWorkErrorSetAction	지정 축이 연결된 노드의 통신이 끊어 졌을때 현재 모션 동작 상태에 대한 예외처리 방법 설정한다	
AxmNetWorkErrorGetAction	지정 축이 연결된 노드의 통신이 끊어 졌을때 현재 모션 동작 상태에 대한 예외처리 방법 확인한다	

모션 구동 중 네트웍이 끊어 졌을 때 모션 구동 상태에 대한 예외처리 방법을 설정하고 확인한다.
dwNetErrorAct 인자로 0으로 설정 할 경우 현재 모션이 계속 진행이 되고, 1로 설정을 할 경우 감속 정지, 2로 설정 할 경우 급정지가 된다
이를 AxmNetWorkErrorSetAction 함수에서 설정 할 수 있으며, AxmNetWorkErrorGetAction함수에서 확인 할 수 있다.

AxmNetWorkErrorSetAction

Purpose

지정 축이 연결된 노드의 통신이 끊어 졌을때 현재 모션 동작 상태에 대한 예외처리 방법 설정한다.

Format

C

```
DWORD AxmNetWorkErrorSetAction   (long lAxisNo, DWORD dwNetErrorAct);
```

Visual Basic

```
Function AxmNetWorkErrorSetAction   (ByVal lAxisNo As Long, ByVal dwNetErrorAct As Long) As Long
```

Delphi

```
function AxmNetWorkErrorSetAction   (lAxisNo : LongInt; dwNetErrorAct: Dword) : Dword; stdcall;
```

Input / Output

Name	in/out	Format	Init Value	Explanation
AxisNo	in	long	-	채널(축) 번호(0 ~ (최대축수 - 1))
dwNetErrorAct	In	DWORD	0	접점의 상태

Return AXT_RT_SUCCESS(0000) : Successful execution of API

* See error code Table more information on status error codes

dwNetErrorAct

Value	Explanation
00	모션 구동 계속 진행
01	감속 정지
02	급정지

Description

지정 축이 연결된 노드의 통신이 끊어 졌을때 현재 모션 동작 상태에 대한 예외처리 방법 설정한다. 이 설정은 AxmNetWorkErrorGetAction 로 확인할 수 있다.

dwNetErrorAct 가 0으로 설정 시 모션 구동은 계속 진행, 1로 설정시 감속 정지, 2로 설정시 급정지로 설정 된다.

C Example

```
// 0축아 연결된 노드가 끊어 졌을 때 현재 모션 구동은 계속 진행된다는 것을 설정
AxmNetWorkErrorSetAction  (0, 0);

// 0축아 연결된 노드가 끊어 졌을 때 현재 모션 구동은 계속 진행된다는 것을 확인
DWORD dwpNetErrorAct;
AxmNetWorkErrorGetAction (0, &dwpNetErrorAct);
```

VB Example

```
' 0축의 정지 신호 사용 유무를 설정
AxmNetWorkErrorSetAction  0, 0_

' 0축의 비상 정지 신호 사용 유무 설정 값을 확인
Dim dwpNetErrorAct As Long
AxmNetWorkErrorGetAction 0 dwpNetErrorAct
```

Delphi Example

```
var
dwpNetErrorAct: DWord;

Begin
{ 0축의 정지 신호 사용 유무를 설정 }
AxmNetWorkErrorSetAction  (0, 0);
{ 0축의 비상 정지 신호 사용 유무 설정 값을 확인 }
AxmNetWorkErrorGetAction (0, @dwpNetErrorAct);
end;
```

See Also

AxmNetWorkErrorGetAction

AxmNetWorkErrorGetAction

Purpose

지정 축이 연결된 노드의 통신이 끊어 졌을때 현재 모션 동작 상태에 대한 예외처리 방법 확인한다

Format

C

```
DWORD AxmNetWorkErrorGetAction (long lAxisNo, DWORD *dwpNetErrorAct);
```

Visual Basic

```
Function AxmNetWorkErrorGetAction (ByVal lAxisNo As Long, ByRef dwpNetErrorActAs Long) As Long
```

Delphi

```
function AxmNetWorkErrorGetAction (lAxisNo : LongInt; dwpNetErrorAct: PDword) : Dword; stdcall;
```

Input / Output

Name	in/out	Format	Init Value	Explanation
AxisNo	in	long	-	채널(축) 번호(0 ~ (최대축수 - 1))
dwpNetErrorAct	out	DWORD*	0	접점의 상태

Return AXT_RT_SUCCESS(0000) : Successful execution of API

* See error code Table more information on status error codes

dwNetErrorAct

Value	Explanation
00	모션 구동 계속 진행
01	감속 정지
02	급정지

Description

' AxmNetWorkErrorSetAction ' 로 설정한 네트웍이 끊어 졌을 때 모션 상태를 반환한다.
dwpNetErrorAct 가 0이면 모션 구동은 계속 진행, 1이면 감속 정지, 2이면 설정시 급정지로 설정된 것을 확인 할 수 있다.

C Example

```
// 0축이 연결된 노드의 통신이 끊어 졌을 때 모션 구동은 계속 진행 되도록 설정한다.
AxmNetWorkErrorSetAction  (0, 0);

// 0측이 연결된 노드의 통신이 끊어 졌을 때 모션 구동은 계속 되는 설정을 확인한다.
DWORD    dwpNetErrorAct;

AxmNetWorkErrorGetAction (0, & dwpNetErrorAct);
```

VB Example

```
'0축이 연결된 노드의 통신이 끊어 졌을 때 모션 구동은 계속 진행 되도록 설정한다.
AxmNetWorkErrorSetAction 0, 0

'0측이 연결된 노드의 통신이 끊어 졌을 때 모션 구동은 계속 되는 설정을 확인한다.
Dim dwpNetErrorAct As Long

AxmNetWorkErrorGetAction0, dwpNetErrorAct
```

Delphi Example

```
var
dwpNetErrorAct: DWord;

Begin
{ 0축이 연결된 노드의 통신이 끊어 졌을 때 모션 구동은 계속 진행 되도록 설정한다.}
AxmNetWorkErrorSetAction  (0, 0);
{ 0측이 연결된 노드의 통신이 끊어 졌을 때 모션 구동은 계속 되는 설정을 확인한다.}
AxmNetWorkErrorGetAction (0, @dwpNetErrorAct);
end;
```

See Also

AxmNetWorkErrorSetAction

에러코드 테이블(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 하세요.

에러코드 (Error Code)	Description
AXT_RT_SUCCESS(0000)	API 함수 수행 성공
AXT_RT_OPEN_ERROR (1001)	라이브러리 오픈 되지않음
AXT_RT_OPEN_ALREADY (1002)	라이브러리 오픈 되어있거나 사용중 임
AXT_RT_NOT_OPEN (1053)	라이브러리 초기화 실패
AXT_RT_NOT_SUPPORT_VERSION(1054)	지원하지않는 하드웨어
AXT_RT_INVALID_NODE_NO (1101)	유효하지않는 노드 번호
AXT_RT_INVALID_MODULE_POS (1102)	유효하지않는 모듈 위치
AXT_RT_INVALID_LEVEL (1103)	유효하지않는 레벨
AXT_RT_FLASH_BUSY(1150)	플래시 상태가 Busy 상태 임
AXT_RT_ERROR_VERSION_READ(1151)	라이브러리 버전을 읽을수 없음
AXT_RT_ERROR_NETWORK(1152)	네트웍크 에러
AXT_RT_ERROR_HW_ACCESS(1153)	하드웨어 에러
AXT_RT_ERROR_NETWORK_CHEKSUM(1154)	네트웍크 CheckSum 에러
AXT_RT_1ST_BELOW_MIN_VALUE (1160)	첫번째 인자값이 최소값보다 더 작음
AXT_RT_1ST_ABOVE_MAX_VALUE (1161)	첫번째 인자값이 최대값보다 더 큼
AXT_RT_2ND_BELOW_MIN_VALUE (1170)	두번째 인자값이 최소값보다 더 작음
AXT_RT_2ND_ABOVE_MAX_VALUE (1171)	두번째 인자값이 최대값보다 더 큼
AXT_RT_3RD_BELOW_MIN_VALUE (1180)	세번째 인자값이 최소값보다 더 작음
AXT_RT_3RD_ABOVE_MAX_VALUE (1181)	세번째 인자값이 최대값보다 더 큼
AXT_RT_4TH_BELOW_MIN_VALUE (1190)	네번째 인자값이 최소값보다 더 작음
AXT_RT_4TH_ABOVE_MAX_VALUE (1191)	네번째 인자값이 최대값보다 더 큼

AXT_RT_5TH_BELOW_MIN_VALUE (1200)	다섯번째 인자값이 최소값보다 더 작음
AXT_RT_5TH_ABOVE_MAX_VALUE (1201)	디섯번째 인자값이 최대값보다 더 큼
AXT_RT_6TH_BELOW_MIN_VALUE (1210)	여섯번째 인자값이 최소값보다 더 작음
AXT_RT_6TH_ABOVE_MAX_VALUE (1211)	여섯번째 인자값이 최대값보다 더 큼
AXT_RT_7TH_BELOW_MIN_VALUE (1220)	일곱번째 인자값이 최소값보다 더 작음
AXT_RT_7TH_ABOVE_MAX_VALUE (1221)	일곱번째 인자값이 최대값보다 더 큼
AXT_RT_8TH_BELOW_MIN_VALUE (1230)	여덟번째 인자값이 최소값보다 더 작음
AXT_RT_8TH_ABOVE_MAX_VALUE (1231)	여덟번째 인자값이 최대값보다 더 큼
AXT_RT_9TH_BELOW_MIN_VALUE (1240)	아홉번째 인자값이 최소값보다 더 작음
AXT_RT_9TH_ABOVE_MAX_VALUE (1241)	아홉번째 인자값이 최대값보다 더 큼
AXT_RT_10TH_BELOW_MIN_VALUE (1250)	열번째 인자값이 최소값보다 더 작음
AXT_RT_10TH_ABOVE_MAX_VALUE (1251)	열번째 인자값이 최대값보다 더 큼
AXT_RT_AIO_OPEN_ERROR (2001)	AIO 모듈 오픈실패
AXT_RT_AIO_NOT_MODULE (2051)	AIO 모듈 없음
AXT_RT_AIO_INVALID_MODULE_NO (2101)	유효하지않은 AIO모듈
AXT_RT_AIO_INVALID_CHANNEL_NO (2102)	유효하지않은 AIO채널번호
AXT_RT_AIO_INVALID_USE (2106)	AIO 함수 사용못함
AXT_RT_AIO_INVALID_TRIGGER_MODE (2107)	유효하지않는 트리거 모드
AXT_RT_DIO_OPEN_ERROR (3001)	DIO 모듈 오픈실패
AXT_RT_DIO_NOT_MODULE (3051)	DIO 모듈 없음
AXT_RT_DIO_INVALID_MODULE_NO (3101)	유효하지않는 DIO 모듈 번호
AXT_RT_DIO_INVALID_OFFSET_NO (3102)	유효하지않는 DIO OFFSET 번호
AXT_RT_DIO_INVALID_LEVEL (3103)	유효하지않는 DIO 레벨
AXT_RT_DIO_INVALID_MODE (3104)	유효하지않는 DIO 모드
AXT_RT_DIO_INVALID_VALUE (3105)	유효하지않는 값 설정
AXT_RT_DIO_INVALID_USE (3106)	DIO 함수 사용못함
AXT_RT_MOTION_OPEN_ERROR(4001)	모션 라이브러리 Open 실패
AXT_RT_MOTION_NOT_MODULE(4051)	시스템에 장착된 모션 모듈이 없음
AXT_RT_MOTION_NOT_INITIAL_AXIS_NO(4053)	해당 축 모션 초기화 실패

AXT_RT_MOTION_NOT_PARA_READ(4055)	원점 구동 설정 파라미터 로드 실패
AXT_RT_MOTION_INVALID_AXIS_NO(4101)	해당 축이 존재하지 않음
AXT_RT_MOTION_INVALID_METHOD(4102)	해당 축 구동에 필요한 설정이 잘못됨
AXT_RT_MOTION_INVALID_USE(4103)	'uUse' 인자값이 잘못 설정됨
AXT_RT_MOTION_INVALID_LEVEL(4104)	'uLevel' 인자값이 잘못 설정됨
AXT_RT_MOTION_INVALID_BIT_NO(4105)	범용 입출력 해당 비트가 잘못 설정됨
AXT_RT_MOTION_INVALID_STOP_MODE(4106)	모션 정지 모드 설정값이 잘못됨
AXT_RT_MOTION_INVALID_TRIGGER_MODE(4107)	트리거 설정 모드가 잘못 설정됨
AXT_RT_MOTION_INVALID_TRIGGER_LEVEL(4108)	트리거 출력 레벨 설정이 잘못됨
AXT_RT_MOTION_INVALID_SELECTION(4109)	'uSelection' 인자가 COMMAND 또는 ACTUAL 이외의 값으로 설정되어 있음
AXT_RT_MOTION_INVALID_TIME(4110)	Trigger 출력 시간값이 잘못 설정되어 있음
AXT_RT_MOTION_INVALID_FILE_LOAD(4111)	모션 설정값이 저장된 파일이 로드가 안됨
AXT_RT_MOTION_INVALID_FILE_SAVE(4112)	모션 설정값을 저장하는 파일 저장에 실패함
AXT_RT_MOTION_INVALID_VELOCITY(4113)	모션 구동 속도값이 0으로 설정되어 모션 에러 발생
AXT_RT_MOTION_INVALID_ACCELTIME(4114)	모션 구동 가속 시간값이 0으로 설정되어 모션 에러 발생
AXT_RT_MOTION_INVALID_PULSE_VALUE(4115)	모션 단위 설정 시 입력 펄스값이 0보다 작은값으로 설정됨
AXT_RT_MOTION_INVALID_NODE_NUMBER (4116)	위치나 속도 오버라이드 함수가 모션 정지 중에 실행됨
AXT_RT_MOTION_INVALID_TARGET(4117)	다축 모션 정지 원인에 관한 플래그를 반환한다.
AXT_RT_MOTION_ERROR_IN_NONMOTION(4151)	모션 구동중이어야 되는데 모션 구동중이 아닐 때
AXT_RT_MOTION_ERROR_IN_MOTION(4152)	모션 구동 중에 다른 모션 구동 함수를 실행함
AXT_RT_MOTION_ERROR(4153)	다축 구동 정지 함수 실행 중 에러 발생함

AXT_RT_MOTION_ERROR_GANTRY_ENABLE(4154)	겐트리 enable이 되어있어 모션중일 때 또 겐트리 enable을 눌렀을 때
AXT_RT_MOTION_ERROR_GANTRY_AXIS(4155)	겐트리 축이 마스터채널(축) 번호(0 ~ (최대축수 - 1))가 잘못 들어갔을 때
AXT_RT_MOTION_ERROR_MASTER_SERVOON(4156)	마스터 축 서보온이 안되어있을 때
AXT_RT_MOTION_ERROR_SLAVE_SERVOON(4157)	슬레이브 축 서보온이 안되어있을 때
AXT_RT_MOTION_INVALID_POSITION(4158)	유효한 위치에 없을 때
AXT_RT_ERROR_NOT_SAME_MODULE (4159)	똑 같은 모듈내에 있지 않을경우
AXT_RT_ERROR_NOT_SAME_PRODUCT (4161)	제품이 서로 다를경우
AXT_RT_NOT_CAPTURED(4162)	위치가 저장되지 않을 때
AXT_RT_ERROR_NOT_SAME_IC(4163)	같은 칩내에 존재하지않을 때
AXT_RT_ERROR_NOT_GEARMODE(4164)	기어모드로 변환이 안될 때
AXT_ERROR_CONTI_INVALID_AXIS_NO(4165)	연속보간 축맵핑할 때 유효한축이 아닐 때
AXT_ERROR_CONTI_INVALID_MAP_NO(4166)	유효한 맵핑 번호가 아닐 때
AXT_ERROR_CONTI_EMPTY_MAP_NO(4167)	맵핑 번호가 비워있을 때
AXT_RT_MOTION_ERROR_CACULATION(4168)	계산상의 오차가 발생했을 때
AXT_RT_ERROR_NOT_SAME_NODE(4169)	노드 번호가 틀릴 때
AXT_ERROR_HELICAL_INVALID_AXIS_NO(4170)	헬리컬 축 번호가 존재하지 않을 때
AXT_ERROR_HELICAL_INVALID_MAP_NO(4171)	헬리컬 맵이 존재하지 않을 때
AXT_ERROR_HELICAL_EMPTY_MAP_NO(4172)	헬리컬 맵 번호가 비어 있을 때
AXT_RT_MOTION_INTERPOL_VALUE(4184)	보간할 때 입력 값이 잘못넣어졌을 때
AXT_RT_MOTION_HOME_SEARCHING(4201)	홈을 찾고 있는 중일 때 다른 모션 함수들을 사용할 때
AXT_RT_MOTION_HOME_ERROR_SEARCHING(4202)	홈을 찾고 있는 중일 때 외부에서 사용자나 혹은 어떤것에 의한 강제로 정지당할 때
AXT_RT_MOTION_HOME_ERROR_START(4203)	초기화 문제로 홈시작 불가할 때
AXT_RT_MOTION_HOME_ERROR_GANTRY(4204)	홈을 찾고 있는 중일 때 겐트리 enable 불가할 때

AXT_RT_MOTION_POS_OUTOFBOUND(4251)	설정한 위치값이 설정 최대값보다 크거나 최소값보다 작은값임
AXT_RT_MOTION_PROFILE_INVALID(4252)	구동 속도 프로파일 설정이 잘못됨
AXT_RT_MOTION_VELOCITY_OUTOFBOUND(4253)	구동 속도값이 최대값보다 크게 설정됨
AXT_RT_MOTION_MOVE_UNIT_IS_ZERO(4254)	구동 단위값이 0으로 설정됨
AXT_RT_MOTION_SETTING_ERROR(4255)	속도, 가속도, 저크, 프로파일 설정이 잘못됨
AXT_RT_MOTION_DISABLE_TRIGGER(4257)	트리거 출력이 Disable 상태임
AXT_RT_MOTION_INVALID_CONT_INDEX(4258)	연속 보간 Index값 설정이 잘못됨
AXT_RT_INIT_DOSE_NOT_EXIST_LAN_CARD(4301)	랜카드가 존재하지 않을 때
AXT_RT_INIT_DOES_NOT_RESPONSE_SLAVE(4302)	슬레이브에서의 응답이 없을 때
AXT_RT_INIT_INVALID_HOST_IP_ADDRESS(4303)	마스터의 IP 주소가 틀릴 때
AXT_RT_INIT_INVALID_HOST_COUNT(4304)	랜카드 개수가 틀릴 때
AXT_RT_INIT_EIPSTART_FAIL(4305)	Ethernet/IP 통신이 초기화 되지 않을 때
AXT_RT_INIT_ALREADY_INITIALIZED(4306)	초기화가 이미 되어 있을 때
AXT_RT_INIT_INVALID_NET_TYPE(4307)	Network 타입이 틀릴 때
AXT_RT_INIT_PRODUCTID(4308)	모듈 ID가 존재하지 않을 때
AXT_RT_INIT_NOT_ENOUGH_MEMORY(4309)	메모리가 충분하지 않을 때
AXT_RT_INIT_DOES_NOT_RESPONSE_SLAVE_AT_RING(4310)	링 오픈시 응답이 없을 때
AXT_RT_INIT_NETWORK_ERROR(4311)	네트워크 초기화 에러
AXT_RT_INIT_BACKGROUND_NOT_STARTED(4312)	BackGround Process 에러
AXT_RT_INIT_BACKGROUND_START_FAIL(4313)	BackGround Process 초기화 실패

이 설명서의 내용은 예고 없이 변경될 수 있습니다. 용례에 사용된 회사, 기관, 제품, 인물 및 사건 등은 실제 데이터가 아닙니다. 어떠한 실제 회사, 기관, 제품, 인물 또는 사건과도 연관시킬 의도가 없으며 그렇게 유추해서도 안됩니다. 해당 저작권법을 준수하는 것은 사용자의 책임입니다. 저작권에서의 권리와는 별도로, 이 설명서의 어떠한 부분도(주)아진엑스텍의 명시적인 서면 승인 없이는 어떠한 형식이나 수단(전기적, 기계적, 복사기에 의한 복사, 디스크 복사 또는 다른 방법) 또는 다른 목적으로도 복제되거나, 검색 시스템에 저장 또는 도입되거나, 전송될 수 없습니다.

(주)아진엑스텍은 이 설명서 본 안에 관련된 특허권, 상표권, 저작권 또는 기타 지적 소유권 등을 보유할 수 있습니다. 서면 사용권 계약에 따라 (주)아진엑스텍으로부터 귀하에게 명시적으로 제공된 권리 이외에, 이 설명서의 제공은 귀하에게 이러한 특허권, 저작권 또는 기타 지적 소유권 등에 대한 어떠한 사용권도 허용하지 않습니다.