

AXL library family
AXM

Product Information

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

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

Useful Contact Information

Customer Support Seoul
Tel : 82-031-436-2180~2 Fax: 82-031-436-2183

Customer Support Cheonan
Tel : 82-041-555-9771~2 Fax: 82-041-555-9773

Customer Support Deagu
Tel : 82-053-593-3700~2 Fax: 82-053-593-3703



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

Contents

CONTENTS.....	3
HOW TO USE MOTION LIBRARY	9
Introduction of Motion Library	10
Introduction of Motion Control	10
Introduction of Motion Library	11
Initialization and Exiting	13
Initialization of Library	13
Exiting Library	15
Motion Parameter	17
Motion Parameter Setting	17
Motion Signal	27
Motion Signal Setting	28
Universal Input/Output Signal Reading and Output Writing.....	34
Saving and Opening Setting Information	36
Motion Drive.....	40
Introduction of Motion Drive	41
1) One Axis	41
2) Multi Axis	41
Glossary of Motion Drive	42
1) Servo On	42
2) Velocity Profile.....	43
3) Exit API at Start Point and Exit API at End Point	44
4) Relative Motion Drive and Absolute Motion Drive.....	44
One-Axis Drive.....	45
1) Position Drive	45
2) Velocity Drive	49
3) Signal Search Drive	52
4) Home Search Drive	56
5) Stop of Motion Drive	62
6) Verification of Motion Drive State	65
Multi-axis Drive	75
1) Multi-axis Position Drive	75
2) Interpolation Drive	80
3) Continuous Interpolation Drive.....	90
Expanded Motion Drive.....	101
CRC Signal Setting.....	102
Override Drive	106
MPG Drive (Manual Pulse Generation)	110
Sync Drive	121
Gantry Drive	124
1) Setting and Cancellation of Gantry Drive	124

2) Home Search in Gantry Drive	125
Virtual Axis Mapping	134
Interrupt Setting.....	137
1) Interrupt Management Method Setting	144
2) Interrupt Use Setting.....	146
Use of Trigger Signal	151
Advanced Motion Drive(PCI-N804/404 Board Exclusive Use API)	159
Helical Interpolation Motion.....	159
1) Continuous Helical Interpolation Motion	164
Spline Interpolation Motion	168
MOTION COMMAND MANUAL INFORMATION.....	175
Header File.....	175
Name of API in This Manual	175
MOTION COMMAND QUICK LIST	177
API List	178
Board and Module Verification API (Info) – Information	178
Virtual Axis API (Virtual)	178
Interrupt API (Interrupt)	179
Motion Parameter Setting API (Mot) -> Motor Parameter	179
Input and Output Setting API (Signal).....	181
State Verification API after Motion Drive (Status)	183
Home Search API(Home)	183
Position Drive API (Move)	185
Position and Velocity Override Drive API (Override)	186
MASTER, SLAVE Drive API (Link).....	186
Gantry Drive API (Gantry)	186
Interpolation Drive API (Line, Circle)	188
Continuous Interpolation Setting and Drive API (Conti) -> Continue	189
Trigger API (Trigger)	190
CRC(Remaining Pulse Clear) Drive API.....	190
MPG(Manul pulse Generator) Drive API.....	191
Helical Interpolation Setting and Drive API.....	191
Spline Interpolation Setting and Drive API.....	192
Define Sentence.....	193
MOTION COMMAND FUNCTION LIST.....	211
Board and Module Initialization	212
AxmInfoGetAxis	213
AxmInfoIsMotionModule	215
AxmInfoIsInvalidAxisNo.....	217
AxmInfoGetAxisCount	218
AxmInfoGetFirstAxisNo	219

Virtual Axis API	222
AxmVirtualSetAxisNoMap	223
AxmVirtualGetAxisNoMap	225
AxmVirtualSetMultiAxisNoMap	227
AxmVirtualGetMultiAxisNoMap	229
AxmVirtualResetAxisMap	231
 Interrupt API.....	 232
AxIInterruptEnable	233
AxIInterruptDisable	235
AxmInterruptSetAxis.....	236
AxmInterruptSetAxisEnable	238
AxmInterruptGetAxisEnable	240
AxmInterruptRead	242
AxmInterruptReadAxisFlag	244
AxmInterruptSetUserEnable.....	251
AxmInterruptGetUserEnable	259
 Motion Parameter Setting API.....	 267
AxmMotLoadParaAll	269
AxmMotSaveParaAll	273
AxmMotSetParaLoad	277
AxmMotGetParaLoad	281
AxmMotSetPulseOutMethod	285
AxmMotGetPulseOutMethod.....	289
AxmMotSetEncInputMethod	291
AxmMotGetEncInputMethod	294
AxmMotSetMoveUnitPerPulse	296
AxmMotGetMoveUnitPerPulse	299
AxmMotSetDecelMode	301
AxmMotGetDecelMode	303
AxmMotSetRemainPulse	305
AxmMotGetRemainPulse	307
AxmMotSetMaxVel	309
AxmMotGetMaxVel	311
AxmMotSetAbsRelMode	313
AxmMotGetAbsRelMode	315
AxmMotSetProfileMode	317
AxmMotGetProfileMode	320
AxmMotSetAccelUnit	322
AxmMotGetAccelUnit	324
AxmMotSetMinVel	326
AxmMotGetMinVel	328
AxmMotSetAccelJerk	330
AxmMotGetAccelJerk	332
AxmMotSetDecelJerk	334
AxmMotGetDecelJerk	336
 Input and Output Level Setting API	 338
AxmSignalSetZphaseLevel	340
AxmSignalGetZphaseLevel	342
AxmSignalSetServoOnLevel	344

AxmSignalGetServoOnLevel	346
AxmSignalSetServoAlarmResetLevel	348
AxmSignalGetServoAlarmResetLevel	350
AxmSignalSetInpos	352
AxmSignalGetInpos	354
AxmSignalReadInpos	356
AxmSignalSetServoAlarm	358
AxmSignalGetServoAlarm	360
AxmSignalReadServoAlarm	362
AxmSignalSetLimit	364
AxmSignalGetLimit	367
AxmSignalReadLimit	369
AxmSignalSetSoftLimit	371
AxmSignalGetSoftLimit	374
AxmSignalSetStop	376
AxmSignalGetStop	378
AxmSignalReadStop	380
AxmSignalServoOn	382
AxmSignalServoOn	385
AxmSignalServoAlarmReset	387
AxmSignalWriteOutput	390
AxmSignalReadOutput	392
AxmSignalWriteOutputBit	394
AxmSignalReadOutputBit	396
AxmSignalReadInput	398
AxmSignalReadInputBit	400
Status Checking APIs After Motion Move	402
AxmStatusReadInMotion	403
AxmStatusReadDrivePulseCount	405
AxmStatusReadMotion	407
AxmStatusReadStop	414
AxmStatusReadMechanical	423
AxmStatusReadVel	427
AxmStatusReadPosError	429
AxmStatusReadDriveDistance	431
AxmStatusSetActPos	433
AxmStatusGetActPos	435
AxmStatusSetCmdPos	437
AxmStatusGetCmdPos	439
Home Search API	441
AxmHomeSetSignalLevel	445
AxmHomeGetSignalLevel	447
AxmHomeReadSignal	449
AxmHomeSetMethod	451
AxmHomeGetMethod	454
AxmHomeSetVel	457
AxmHomeGetVel	460
AxmHomeSetStart	463
AxmHomeSetResult	465
AxmHomeGetResult	469

AxmHomeGetRate	473
Position Move API.....	477
AxmMoveStartPos	479
AxmMovePos	482
AxmMoveVel	484
AxmMoveStartMultiVel	486
AxmMoveSignalSearch	489
AxmMoveSignalCapture	492
AxmMoveGetCapturePos.....	495
AxmMoveStartMultiPos	497
AxmMoveMultiPos	500
AxmMoveStop	503
AxmMoveEStop	505
AxmMoveSStop	507
Position / Velocity Override Move API.....	509
AxmOverridePos	510
AxmOverrideSetMaxVel	512
AxmOverrideVel	514
AxmOverrideAccelVelDecel	516
AxmOverrideVelAtPos.....	518
MASTER, SLAVE Move API	520
AxmLinkSetMode	521
AxmLinkGetMode	523
AxmLinkResetMode	525
Interpolation Move API (Line, Circle)	526
AxmLineMove	528
AxmCircleCenterMove	535
AxmCirclePointMove	543
AxmCircleRadiusMove	551
AxmCircleAngleMove	559
Continuous interpolation setting and move API	567
AxmContiSetAxisMap	570
AxmContiGetAxisMap.....	573
AxmContiSetAbsRelMode	576
AxmContiGetAbsRelMode	579
AxmContiBeginNode	582
AxmContiEndNode	586
AxmContiReadFree	590
AxmContiReadIndex	594
AxmContiWriteClear.....	598
AxmContiStart	600
AxmContiIsMotion	603
AxmContiGetNodeNum	607
AxmContiGetTotalNodeNum.....	611
Trigger API	615
Trigger signal use	615

AxmTriggerSetTimeLevel	616
AxmTriggerGetTimeLevel	619
AxmTriggerSetAbsPeriod	622
AxmTriggerGetAbsPeriod	625
AxmTriggerSetBlock	627
AxmTriggerGetBlock	629
AxmTriggerOneShot	631
AxmTriggerSetTimerOneshot	633
AxmTriggerOnlyAbs	635
AxmTriggerSetReset	637
Gantry move function	638
AxmGantrySetEnable	645
AxmGantryGetEnable	649
AxmGantrySetDisable	652
CRC(Remaining pulse clear) API	654
AxmCrcSetMaskLevel	655
AxmCrcGetMaskLevel	657
AxmCrcSetOutput	659
AxmCrcGetOutput	661
AxmCrcSetEndLimit	663
AxmCrcGetEndLimit	666
MPG move API	668
AxmMPGSetEnable	671
AxmMPGGetEnable	674
AxmMPGSetRatio	676
AxmMPGGetRatio	678
AxmMPGReset	680
Helical interpolation setting and move API	681
AxmHelixCenterMove	683
AxmHelixPointMove	690
AxmHelixRadiusMove	696
AxmHelixAngleMove	703
Spline interpolation setting and move API	710
AxmSplineWrite	711
ERROR CODE TABLE CHECK	716

How to Use Motion Library

[Introduction of Motion Library](#)

[Introduction of Motion Control](#)

[Introduction of Motion Library](#)

[Initialization and Exiting](#)

[Initialization of Library](#)

[Exiting Library](#)

[Motion Parameter](#)

[Motion Parameter Setting](#)

[Motion Signal](#)

[Motion Signal Setting](#)

[Universal Input/Output Signal Reading and Output Writing](#)

[Saving and Opening Setting Information](#)

[Motion Drive](#)

[Introduction of Motion Drive](#)

[1\) One Axis](#)

[2\) Multi Axis](#)

[Glossary of Motion Drive](#)

[1\) Servo On](#)

[2\) Velocity Profile](#)

[3\) Exit API at Start Point and Exit API at End Point](#)

[4\) Relative Motion Drive and Absolute Motion Drive](#)

[One-Axis Drive](#)

[1\) Position Drive](#)

[2\) Velocity Drive](#)

[3\) Signal Search Drive](#)

[4\) Home Search Drive](#)

[5\) Stop of Motion Drive](#)

[6\) Verification of Motion Drive State](#)

[Multi-axis Drive](#)

[1\) Multi-axis Position Drive](#)

[2\) Interpolation Drive](#)

[3\) Continuous Interpolation Drive](#)

[Expanded Motion Drive](#)

[CRC Signal Setting](#)

[Override Drive](#)

[MPG Drive \(Manual Pulse Generation\)](#)

[Sync Drive](#)

[Gantry Drive](#)

[1\) Setting and Cancellation of Gantry Drive](#)

[2\) Home Search in Gantry Drive](#)

[Virtual Axis Mapping](#)

[Interrupt Setting](#)

[1\) Interrupt Management Method Setting](#)

[2\) Interrupt Use Setting](#)

[Use of Trigger Signal](#)

[Advanced Motion Drive\(Functions dedicated for PCI-N804/404 Boards\)](#)

[Helical Interpolation Motion](#)

[1\) Continuous Helical Interpolation Motion](#)

[Spline Interpolation Motion](#)

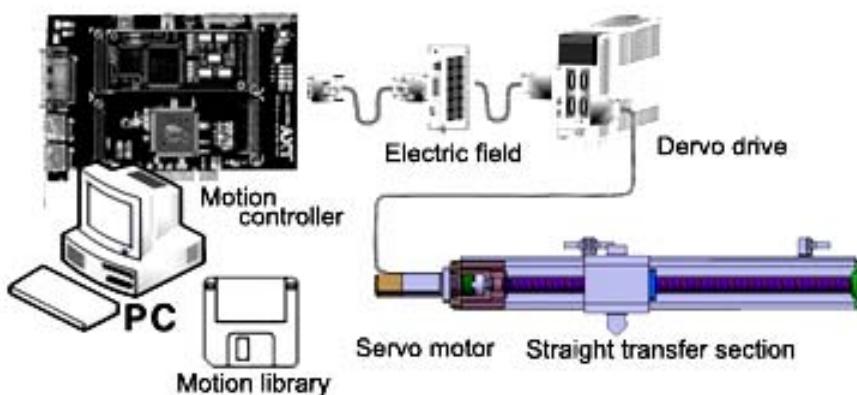
Introduction of Motion Library

[Introduction of Motion Control](#)[Introduction of Motion Library](#)

Introduction of Motion Control

If you take a look at the devices that assemble or inspect accurate parts such as semiconductor, or the machine tools that process structures, what you can see obviously is the motion controller (transfer system) where actual movement occurs. Motion means movement. Motion can be explained as moves of certain object from one place to another. And, control is a concept of handling. In other words, it means that we can move an object to the direction and position that we want. Ultimately, motion control is a combination of motion and control, and can be defined as a movement of an object to the direction that we want. Following parts stated below are required for the motion control.

- Straight transfer section : mechanical part that rectilineal movement occurs
- Servo motor : device that generates power
- Servo drive : device that flows electricity to servo motor
- Motion controller : device that generates various signals to control motion
- Electric field : other connecting devices to connect motion controller and servo drive
- PC : primary device that controls motion controller by PC programming
- Motion library : API that moves motion controller



Introduction of Motion Library

Window programming which programs on MS Windows is prevailing recently. Variety of tools are provided for the programming tools including Visual C++, C++ Builder, Delphi, Visual Basic, and so on, and AXL provides API functions that can be utilized on all those tools. Especially, since the library functions for the motion drive provide the very subdivided functions from the specific control for the mechanical signal and universal input/output signal of general device part to the function for the one-axis and multi-axis interpolation drive function, they can be utilized to all motion controls required by users.

This user guide explains the usage of library functions for driving PCI-N804/404 motion controllers including four axes motion control LSI and SMC-2V03 motion controllers including single chip two axes motion control LSI among various motion control modules, both of which support single axis and multi axes drive as well as interpolation function. Precision control of stepping motors and servo motors is possible by using SMC-2V03 and PCI-804/404 motion libraries. Therefore, continuous motion control system can be realized without increasing the CPU load by using these motion control products, and home search, velocity and position override, linear interpolation, circular interpolation can be performed by the combination of linear and S-curve acceleration and deceleration profiles.

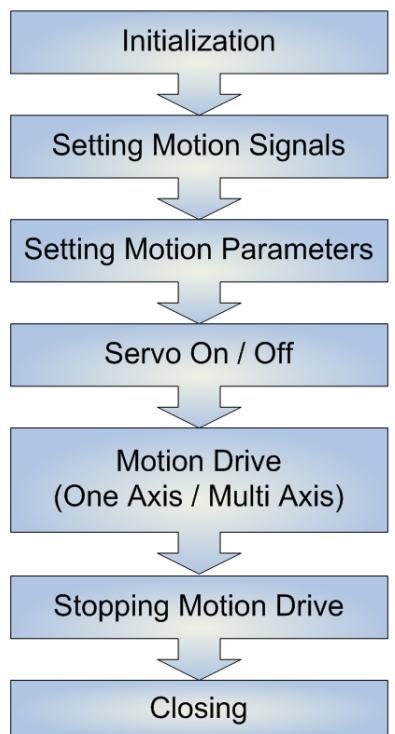
Basic Motion Drive

- Library initialization and exiting
- Motion parameter setting
- Motion signal setting
- Utilization of Universal Input/Output Signal
- One-axis Drive
 - Position Drive
 - Velocity Drive
 - Signal Search Drive
 - Home Search Drive
- Multi-axis drive
 - Multi-axis Position Drive
 - Multi-axis Interpolation Drive
 - Multi-axis Continuous Interpolation Drive

Advanced Motion Drive

- CRC Signal Setting
- Sensor Position Drive
- Override Drive
- MPG Drive
- Sync Drive
- Dynamic Link
- Gantry Drive
- Virtual Axis Mapping
- Interrupt Setting
- Use of Trigger Signal

Basic motion drive is divided into initialization and exiting, input/output signal, parameter setting and verification, servo on/off, signal setting drive, one-axis position drive, one-axis velocity drive, monitoring of drive state and stop state, and stopping one-axis drive and is done by the following order. On each chapter, it is explained on the basis of concept of hardware and basic API, and the sample of basic API that is necessary to the chapter.



Initialization and Exiting

The initialization work of AXL library is done by running AxIOpen API. When AxIOpen API runs, it initializes all library as well as base board and module that are installed internally. Hence, user must verify existence of motion module before using motion library.

[Initialization of Library](#)

[Exiting Library](#)

Initialization of Library

Initialization of Library

AxIOpen : AxIOpen API is an API that initializes library. When a user uses interrupt, it registers IRQ number that manage interrupt. Particularly, inputted IRQ number during the library initialization is ignored since IRQ number is generated automatically on the PCI type base board.

This API returns error code to the return value, so following can be used

```
// Initialize library.  
// 7 means IRQ. IRQ is set automatically in PCI.  
if (AxIOpen(7) == AXT_RT_SUCCESS)  
    printf("Library is initialized.");
```

Verification whether library is initialized or not

AxIsOpened : AxIsOpened API is an API that verifies whether library is initialized or not. This API returns BOOL value which is not an error code. If the library is initialized, it returns TRUE, otherwise it returns FALSE.

```
// Verify if library is initialized.  
if (AxIsOpened())  
    printf("Library is initialized.");  
else  
    printf("Library is not initialized.");
```

Verification whether motion module exists or not

AxmlInfoIsMotionModule : [AxmlInfoIsMotionModule](#) API is an API that verifies whether actual motion module exists or not. This is used to verify the existence of motion module before using motion related API. To verify the existence of module, following is done.

```
// Verify if motion module exists.  
DWORD dwStatus;  
AxmlInfoIsMotionModule (&dwStatus);  
if(dwStatus == STATUS_EXIST)  
    printf("Motion module exists.");  
else  
    printf ("motion module does not exists.");
```

Verification of the number of axis installed in the system

[AxmlInfoGetAxisCount](#) : [AxmlInfoGetAxisCount](#) API is an API that verifies the number of axis installed in the whole system.

```
//Verify the number of axis installed in the system  
long lAxisCount;  
AxmlInfoGetAxisCount (&lAxisCount);  
printf("number of axis installed in the system : %d",lAxisCount);
```

[AxmlInfoGetFirstAxisNo](#) : [AxmlInfoGetFirstAxisNo](#) API is an API that verifies the first axis number on the specific module of specified base board.

```
// Verify the starting axis number on number 0 module of number 0 board.  
long lBoardNo = 0;  
long lModulePos = 0;  
long lFirstAxisNo;  
  
AxmlInfoGetFirstAxisNo (lBoardNo, lModulePos, &lFirstAxisNo);  
print f("starting axis number on the 1st module of number 0 base board : %d", lFirstAxisNo);
```

Exiting Library

AxIClose : AxIClose API is an API that ends library. Library must be ended lastly and return the assigned memory after using. If API runs normally and library ends, it returns TRUE, otherwise it returns FALSE.

```
// End library.
if (AxIClose())
    printf("Library is ended.");
```

```
// Ex1_AXM_InitAndClose.cpp : Defines the entry point for the console application.
// Initialize library, verify information of axis and end.

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

void main(void)
{
    // Initialize library.
    // 7 means IRQ. IRQ is set automatically in PCI.
    DWORD Code = AxIOpen(7);
    if (Code == AXT_RT_SUCCESS)
    {
        printf("Library is initialized.\n");

        // Inspect whether motion module exists or not
        DWORD dwStatus;
        Code = AxmlInfoIsMotionModule (&dwStatus);
        if(Code == AXT_RT_SUCCESS)
        {
            if(dwStatus == STATUS_EXIST)
            {
                printf("Motion module exists.\n");

                // Verify the number of axis installed in the system
                long lpAxisCount;
                AxmlInfoGetAxisCount (&lpAxisCount);
                printf("number of axis installed in the system : %d \n",lpAxisCount);

                //Verify board number, module position, and module ID on number axis 0.
                long lBoardNo, lModulePos;
                long lAxisNo = 0;
                DWORD dwModuleID;
                AxmlInfoGetAxis(lAxisNo, &lBoardNo, &lModulePos, &dwModuleID);
                printf("on number axis 0, board number : %x, module position
                      : %x, module ID : %x\n", lBoardNo,lModulePos,dwModuleID);
        }
    }
}
```

```
//Verify the starting axis number on number 1 module of number 0 board.  
long lFirstAxisNo;  
AxmlInfoGetFirstAxisNo(lBoardNo, lModulePos, &lFirstAxisNo);  
printf("on the %d th number base board, starting axis number on the module of  
the %d th : %d\n", lBoardNo,lModulePos,lFirstAxisNo);  
  
}  
else  
    printf ("AxmlInfoIsMotionModule () : ERROR ( NOT STATUS_EXIST )  
           code 0x%x\n",Code);  
}  
else  
    printf ("AxmlInfoIsMotionMode () : ERROR ( Return FALSE )   code 0x%x\n",Code);  
  
}  
else  
    printf ("AxlOpen() : ERROR   code 0x%x\n",Code);  
  
  
// End library.  
if (AxlClose())  
    printf("Library is ended.\n");  
else  
    printf("Library is not ended normally.\n");  
}
```

Motion Parameter

Motion parameter is a setting value for the right setting of each part that constitutes motion system. Motion system, as it is mentioned before, is combined with several devices such as instrumental part, servo driver, servo motor, electric field, motion controller, and so on. Obviously, each component has its own setting parameter. Therefore, even though just one has a problem for setting, the whole system malfunctions or commits an error.

Motion parameter that is set by motion controller is as following, and each meaning shows below..

Unit per Pulse

- it is a value that sets a unit of ordered value from the command.

Start / Stop Speed

- it sets velocity when drive part starts and ends.

Pulse Out Method

- it sets type of pulse that outputs from motion controller to servo drive.

Encoder Input Method

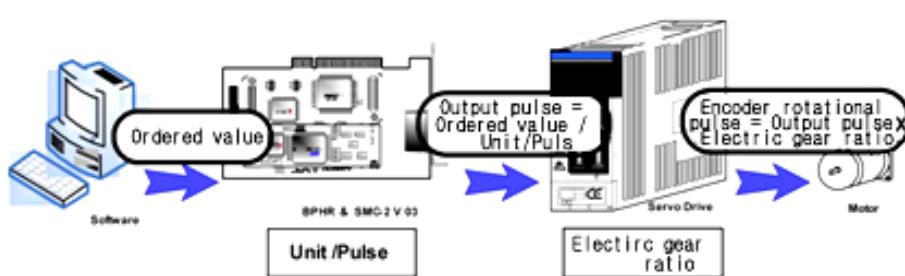
- it sets method that receives encoder signal sent from servo drive to motion controller.

Motion Parameter Setting

Parameter of motion controller is set related to setting and characteristics that each element has in the whole system. Hence, preconditions to set each parameter value should be read carefully, and set adjusting correctly compared to the system that user has

1) Unit per Pulse

Basically, ordered value of moving distance that user assigns on the commands of library takes two steps of unit conversion as the picture below.



If a movement volume is inputted on the software, number of output pulse of motion controller by an already set parameter called unit/pulse is determined. And this output pulse is converted again by parameter called electric gear ration set on the servo drive, and the drive pulse that drives actual motor is determined.

Unit/pulse is a parameter that is to assign unit of values given by running the motion drive API,

and the API which sets this value is [AxmMotSetMoveUnitPerPulse](#). In other words, if it is set 10 on unit and 100 on pulse, 100 pulses would be outputted to the servo drive when 10 is commanded on the motion drive API. For example, let us assume that a motion controller outputs 10,000 pulse for 1 rotation of servo motor on the instrumental part that moves 10mm per rotation of servo motor, and if you want to set the ordered value that is given on the program by mm unit, then do below.

```
//Make an ordered value of axis 0 to be mm unit.
long lAxisNo = 0;
long IPulse = 10000;      // number of pulse for 1 rotation
double dUnit = 10;        // movement distance during 1 rotation
AxmMotSetMoveUnitPerPulse (lAxisNo, dUnit, IPulse);
```

If you want to set the ordered value given on the program by pulse unit, then do below.

```
//Make an ordered value of axis 0 to be pulse unit.
long lAxisNo = 0;
long IPulse = 1;
double dUnit = 1;
AxmMotSetMoveUnitPerPulse (lAxisNo, dUnit, IPulse);
```

* *Electronic gear Ratio (전자기어비)*

Unit/Pulse is a parameter that is set on the motion controller, while electric gear ratio is set on the servo drive.

$$\begin{aligned} [\text{Electric gear ratio}] &= \\ & [\text{Number of encoder pulse that is generated during 1 rotation of servo motor}] / \\ & [\text{Number of output ordered pulse of motion controller during 1 rotation of servo motor}] \end{aligned}$$

If 17-bit encoder is used, 131,072 of encoder pulse can be reduced to 10,000 on the motion controller based on above. That is, what a rotational accuracy of servo motor encoder lowers by an accuracy of mechanical part is an electric gear ratio..

For example, in case of 20mm ball screw, 1/2 of electric gear ratio, and 10000 of pulse number is required for 1 rotation of motor, if it is set as IPulse=10000 and dUnit=10, pulse number that outputs per moving distance of 1mm is set to 1000.



Unit / Pulse = 10 / 10000 Set

1 mm Control Use

2) Start Speed

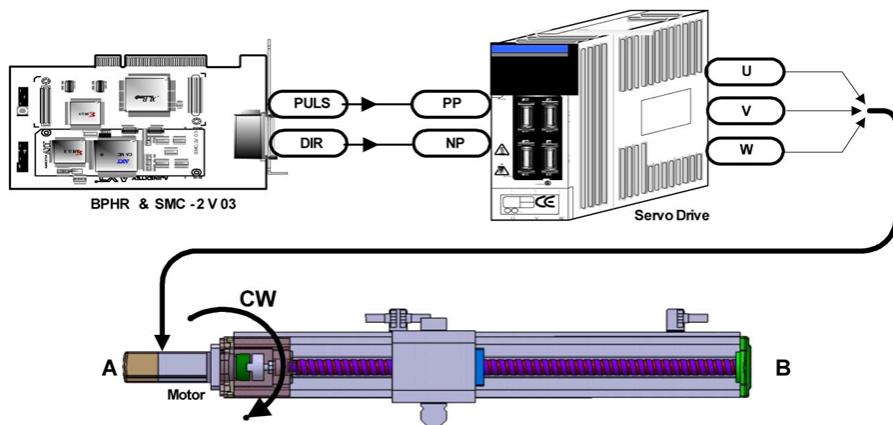
Assign an initial velocity of when motion is started. The initial velocity must be set because it is to be the reference point of motion drive, and a default value is set by 1. Using API of [AxmMotSetMinVel](#), set as following.

```
// Set an initial velocity of axis 0 as 1. Default : 1
long lAxisNo = 0;
double dMinVelocity = 1;
AxmMotSetMinVel (lAxisNo, dMinVelocity);
```

3) Pulse Out Method

Drive command which is ordered by user has a structure that transfers a pulse to the servo drive based on the ordered value by the motion controller showing on the picture below, and that drives the servo drive again by the servo motor.

Let us assume that the motion controller outputs a pulse to the servo drive through PULS pin and DIR pin. Then PP pin and NP pin of the servo drive would receive the pulse and it would drive the servo motor as much as the concerned pulse according to the pulse signal received from the two pins. Now it is required to determine to rotate the servo motor with how much pulse from which pin and with which direction. And it will run normally if the servo drive is set identically with what the motion controller is set.

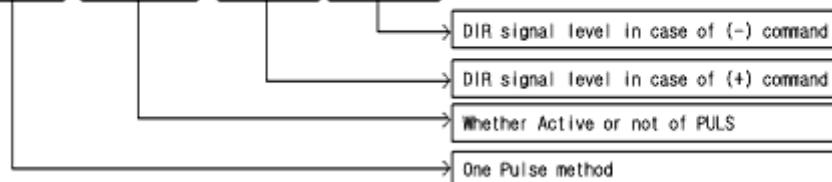


PULS and DIR pulse output method of the motion controller that outputs to the servo drive is classified into one pulse method and two pulse method..

One Pulse Method

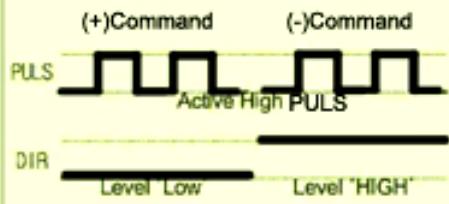
In one pulse method, PULS pin of the motion controller outputs a pulse of the number of rotational amount and DIR pin outputs a signal level of the rotational direction. As shown on the picture below, 1 pulse output method consists of whether PULS signal and DIR are active or not and the signal level when (+) direction command is given and (-)direction command is given. Since there is a little difference on the basis of the servo drive and the servo motor, the pulse output method of the motion controller is to be set carefully based on the setting method on the servo drive.

One High Low High

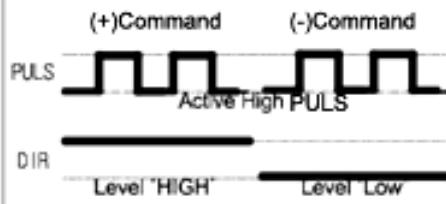


OneHighLowHigh

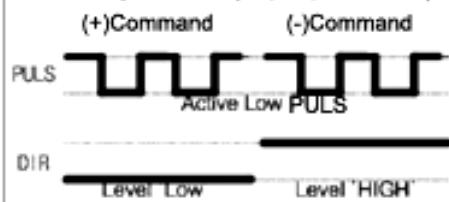
Pulse output method(D6, D5, D4->'000')

**OneHighHighLow**

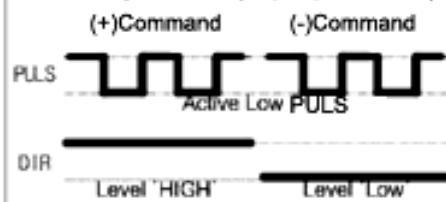
Pulse output method(D6, D5, D4->'001')

**OneLowLowHigh**

Pulse output method(D6, D5, D4->'010')

**OneLowHighLow**

Pulse output method(D6, D5, D4->'010')



Type	Description
OneHighLowHigh	1 pulse method, PULSE(Active High) + direction(DIR=Low) / - direction(DIR=High)
OneHighHighLow	1 pulse method, PULSE(Active High) + direction (DIR=High) / - direction (DIR=Low)
OneLowLowHigh	1 pulse method, PULSE(Active Low) + direction (DIR=Low) / - direction (DIR=High)
OneLowHighLow	1 pulse method, PULSE(Active Low) + direction (DIR=High)/ - direction (DIR=Low)

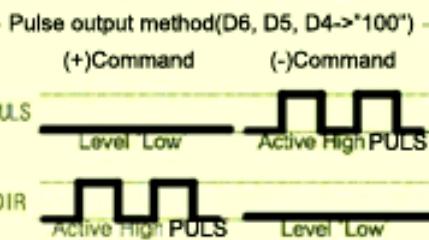
Two Pulse Method

Two pulse method is an output method that is structured to show each rotational amount and rotational direction simultaneously by the pulse signal from PULS pin and DIR pin. Showing on the picture below, 2 pulse output method is structured with a part that sets a pin that a signal would be outputted when a command of (+) direction direction is given, a pin for (-) direction, and whether signal is active or not. Here, Cw means PULS pin and Ccw means DIR pin..

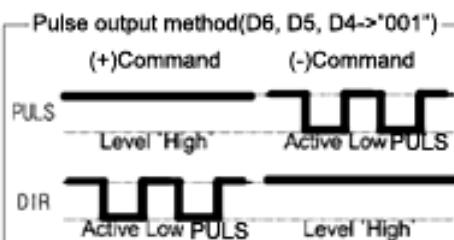
Two Ccw Cw High



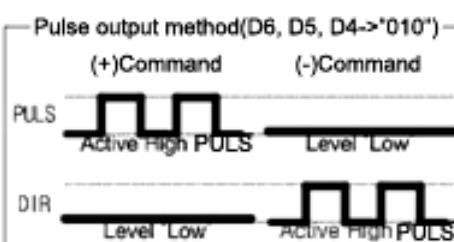
TwoCcxCwHigh



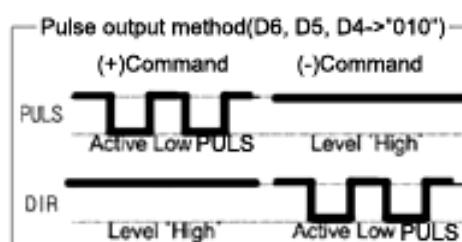
TwoCcxCwLow



TwoCwCcxCwHigh



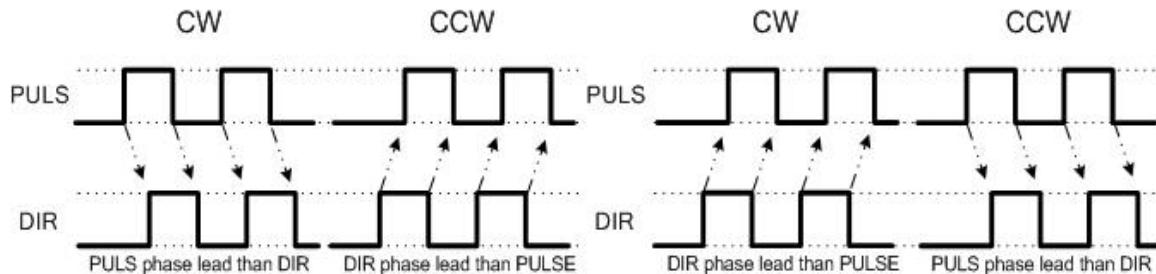
TwoCwCcxCwLow



종 류	설 명
TwoCcxCwHigh	2 pulse method, PULSE(CW:-direction) DIR(CCW:+ direction), Active High
TwoCcxCwLow	2 pulse method, PULSE(CW:- direction) DIR(CCW:+ direction), Active Low
TwoCwCcxCwHigh	2 pulse method, PULSE(CW:+ direction) DIR(CCW:- direction), Active High
TwoCwCcxCwLow	2 pulse method, PULSE(CW:+ direction) DIR(CCW:- direction), Active Low

Two Phase 방식

Two phase mode



An output pulse can be set as 2 phase 4 multiplication output method, and maximum frequency of control pulse can be outputted quarterly..

펄스 출력방식의 설정

User can choose a pulse output method that meets the character of his/her mechanical part and servo drive, and set a pulse output method on an assigned axis using [AxmMotSetPulseOutMethod](#) API.

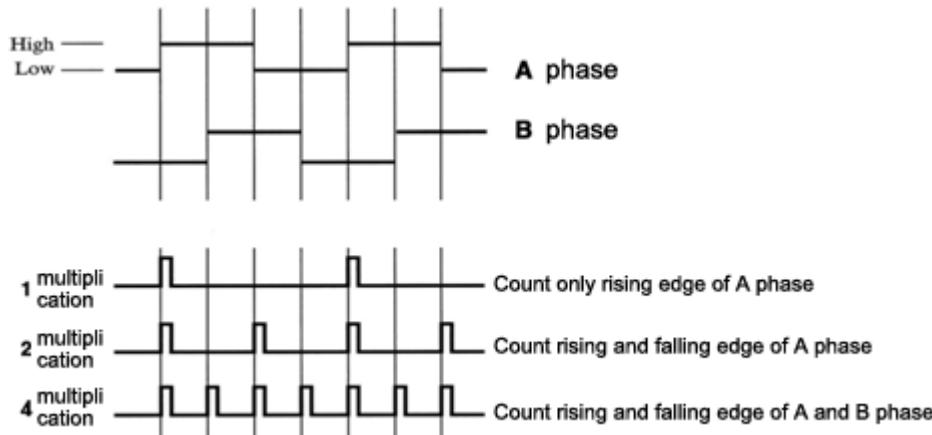
For example, let us assume that servo drive rotates servo motor to forward direction if a pulse is inputted on the PP pin and mechanical part moves a table with (-) direction, and vice versa if a pulse is inputted on the NP. When a command is given on the program, it has to be set that a pulse is outputted from DIR pin of motion controller and inputted to NP pin of servo drive with (+) command, and vice versa with (-) command. It is 2 pulse method because it runs based on a pulse that is inputted to each pin, and it requires to be set as CcwCw because a pulse is outputted on DIR pin with (+) command and outputted on PULS pin with (-) command. And it requires to be set as TwoCcwcwHigh because of setting with High Active.

```
//Pulse output method of axis 0 is to be TwoCcwcwHigh. Default :
long lAxisNo = 0;
DWORD uMethod = TwoCcwcwHigh;
AxmMotSetPulseOutMethod (lAxisNo, uMethod);
```

4) Encoder Input Method

When servo motor rotates, pulse is outputted from encoder according to the rotational amount connected with drive axis. Generally, encoder outputs 3 types of pulses, A, B, and Z phase, and A and B phase has 90 degrees of phase difference. That is, if a motor rotates in forward direction, A phase is showed first and then a pulse on B phase is showed with 90 degrees delay. Z phase is a pulse that is generated whenever the motor rotates a turn. Rotational amount of the motor can have a feedback by counting those pulses.

Using A and B phase on the encoder, a signal which is a multiple of existing frequency can be generated, and this signal-obtaining method is called “multiplication.” A in the picture shows 2 multiplication and B shows 4 multiplication.



Generally 4 multiplication method is used, and it is set using [AxmMotSetEnclInputMethod API](#) and setting values are 8 types as below.

Define	Meaning
ObverseUpDownMode	Forward direction Up/Down
ObverseSqr1Mode	Forward direction 1 multiplication
ObverseSqr2Mode	Forward direction 2multiplication
ObverseSqr4Mode	Forward direction 4multiplication
ReverseUpDownMode	Reverse direction Up/Down
ReverseSqr1Mode	Forward direction 1 multiplication
ReverseSqr2Mode	Forward direction 2 multiplication
ReverseSqr4Mode	Forward direction 4 multiplication

```
//Set 4 multiplication for the input method on the assigned axis
long lAxisNo = 0;
DWORD uMethod = ObverseSqr4Mode;
AxmMotSetEnclInputMethod(lAxisNo, uMethod);
```

```
/ Ex2_AXM_SettingParameter.cpp : Defines the entry point for the console application.
// Initialize library, verify the information of axis and end..

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

void main(void)
{
    // Initialize library.
    // 7 means IRQ. IRQ is set automatically in PCI.
    DWORD Code = AxlOpen(7);
```

```

if (Code == AXT_RT_SUCCESS)
{
    printf("Library is initialized.\n");

    //Inspect whether motion module exists or not
    DWORD dwStatus;
    Code = AxmlInfoIsMotionModule(&dwStatus);
    if(Code == AXT_RT_SUCCESS)
    {
        if(dwStatus == STATUS_EXIST)
        {
            printf("Motion module exists.\n");

            //Ordered value on axis 0 is to be mm unit.
            long lAxisNo = 0;
            long lPulse = 10000;
            double dUnit = 10;
            AxmMotSetMoveUnitPerPulse(lAxisNo, dUnit, lPulse);
            // Verify the set value.
            AxmMotGetMoveUnitPerPulse(lAxisNo, &dUnit, &lPulse);
            printf("UnitPerPulse setting => Unit : %d, Pulse : %f \n", dUnit, lPulse);

            //Set Initial velocity on axis 0 by 1. Default : 1
            double dMinVelocity = 1;
            AxmMotSetMinVel(lAxisNo, dMinVelocity);
            // Verify the set value.
            AxmMotGetMinVel(lAxisNo, &dMinVelocity);
            printf("MinVelocity setting => %2f \n", dMinVelocity);

            //Pulse output method on axis 0 is to be TwoCwCcwHigh.
            Default : OneLowHighLow
            DWORD uMethod = TwoCcwcwHigh;
            AxmMotSetPulseOutMethod (lAxisNo, uMethod);

            // Verify the set value.
            AxmMotGetPulseOutMethod(lAxisNo, &uMethod);
            printf("PulseOutMethod setting => ");
            switch(uMethod)
            {
                case OneHighLowHigh:
                    printf("OneHighLowHigh \n"); break;
                case OneHighHighLow:
                    printf("OneHighHighLow \n"); break;
                case OneLowLowHigh:
                    printf("OneLowLowHigh \n"); break;
                case OneLowHighLow:
                    printf("OneLowHighLow \n"); break;
                case TwoCcwcwHigh:
                    printf("TwoCcwcwHigh \n"); break;
                case TwoCcwcwLow:
                    printf("TwoCcwcwLow \n"); break;
                case TwoCwCcwcwHigh:
                    printf("TwoCwCcwcwHigh \n"); break;
                case TwoCwCcwcwLow:
                    printf("TwoCwCcwcwLow \n"); break;
            }
        }
    }
}

```

```

        printf("TwoCwCcLow \\n");
    }

// Set 4 multiplication for the encoder input method on axis 0.
uMethod = ObverseSqr4Mode;
AxmMotSetEnclInputMethod(lAxisNo, uMethod);
//Verify the set value.
AxmMotGetEnclInputMethod(lAxisNo, &uMethod);
printf("PulseOutMethod setting => ");
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 ("AxmlInfoIsMotionModule() : ERROR ( NOT STATUS_EXIST )
        code 0x%x\\n",Code);
}

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

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

// End library.
if (AxlClose())
    printf("Library is ended.\\n");
else
    printf("Library is not ended normally.\\n");

}

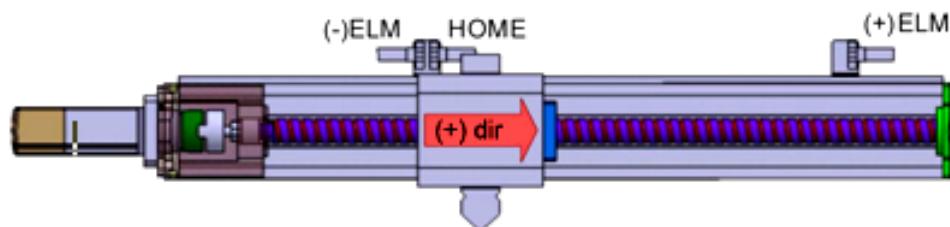
```

Motion Signal

Signal is generally divided into motion signal, universal input signal, and universal output signal. Motion signal is structured with variety of sensors that indicate driving condition of mechanical part, and signals that show condition of servo drive. Universal signal is input signals that can be used by user arbitrarily, and universal output signal is output signals that can be used by user arbitrarily. Among the universal input/output signal, actual bit that can be used by user arbitrarily are 3 each of input and output, since super ordinate 2-bit is already used as a specific function on the motion controller.

1) Motion Signal

Motion signal is named for variety of sensors that indicate driving condition of mechanical part and signals that show condition of servo drive.



End Limit

It is a signal that is transferred to motion controller generated from a sensor which assigns transfer section of mechanical part. Showing on the picture below, it is divided into + and -, and installed on the both side. Generally, if limit signal is emitted during transfer of mechanical part, motion controller stops pulse output and mechanical part makes an emergency stop.

Inposition

It is an output signal of servo drive, and servo drive transfers inposition signal to motion controller when a position of servo motor is in the inposition range that is set on servo drive. In other words, it is a signal which is for verification whether it reached the command position or not..

Alarm

It is an output signal of servo drive, and servo drive transfers alarm signal to motion controller when an abnormal current is generated on servo motor or malfunction is occurred.

Emergency Stop

When this signal is inputted to motion controller, it stops pulse output immediately and mechanical part makes an emergency stop. This signal is usually connected with an emergency switch installed in equipment.

2) Universal Input Signal

It supports 6-bit universal input that can be utilized by user arbitrarily. User can use a signal of additional sensors using those universal input bit.

3) Universal Output Signal

It supports 6-bit universal output that can be utilized by user arbitrarily. User can apply to signal output or operation switch of actuator using those universal output bit.

Motion Signal Setting

Reading and writing general purpose input/output signals

Saving and Reading configuration information

Motion Signal Setting

1) Signal Level and Setting for Use

Different signals are generated because of the characteristics of each mechanical part sensor, and active levels are set to meet the characteristics of those signals. There are LOW(0), HIGH(1), UNUSED(2), USED(3) for setting values, and if LOW or HIGH level is set, it senses as USED automatically. For example, if the internal signal of motion controller is HIGH when a sensor detects an object, it is set as ActiveHIGH. If it is set that each signal is not used, it does not care even when a signal is inputted. User needs to understand the existence of sensor installation of mechanical part before setting.

● *End Limit Signal*

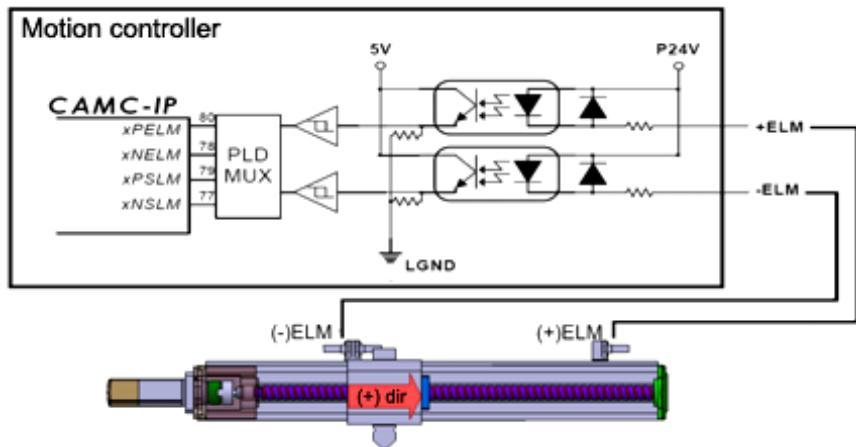
Motion controller has an electrical insulation of external signal (24V) that comes from outside and internal signal (5V) using photo coupler to protect internal circuit.

Showing on the picture below, if the limit sensor transmits HIGH signal to B contact usually and detects transfer section and generates LOW signal, internal signal is to be HIGH as a result of reverse by photo coupler of motion controller. Therefore, this sensor has to be set as Active High.

● *End Limit Signal*

Motion controller has an electrical insulation of external signal (24V) that comes from outside and internal signal (5V) using photo coupler to protect internal circuit.

Showing on the picture below, if the limit sensor transmits HIGH signal to B contact usually and detects transfer section and generates LOW signal, internal signal is to be HIGH as a result of reverse by photo coupler of motion controller. Therefore, this sensor has to be set as Active High



End Limit sensor has two directions of positive(+) and negative(-) on the mechanical section. If user wants to set End Limit Level both Active High and to set to make an emergency stop when limit sensor is detected, follow below.

```
// Set active level of -End limit and +End limit on axis 0 as HIGH.           Default : HIGH
long lAxisNo = 0;
DWORD uStopMode = 0             // [0]Emergency stop, [1]Slowdown stop
DWORD uPositiveLevel = HIGH;
DWORD uNegativeLevel = HIGH;
AxmSignalSetLimit(lAxisNo, uStopMode, uPositiveLevel, uNegativeLevel);
```

● Inposition Signal

For the use of inposition signal and when setting Active Level, use [AxmSignalSetInpos](#) API. If it is needed to set Active Level of inposition signal to Active HIGH, follow below.

```
// Set input level of inposition signal on axis 0.           Default : HIGH
long lAxisNo = 0;
DWORD uUse = HIGH;
AxmSignalSetInpos (lAxisNo, uUse);
```

● Alarm Signal

For the use of alarm signal and when setting Active Level, use [AxmSignalSetServoAlarm](#) API. If it is needed to set Active Level of alarm signal to Active LOW, follow below.

```
// Set input level of alarm signal on axis 0.           Default : HIGH
long lAxisNo = 0;
DWORD uUse = LOW;
AxmSignalSetServoAlarm (lAxisNo, uUse);
```

● Emergency Stop Signal

For the use of emergency stop signal and when setting Active Level, use [AxmSignalSetStop](#) API. If it is needed to set Active Level of emergency stop signal to Active HIGH, follow below.

```
// Set input level of ESTOP on axis 0.
long lAxisNo = 0;
DWORD uStopMode = 0;      // [0] Emergency stop, [1] Slowdown stop
DWORD uLevel = HIGH;
AxmSignalSetStop (lAxisNo, uStopMode, uLevel);
```

2) Signal Reading

● Input Verification of End Limit Signal

If it is needed to verify whether +/-End Limit signal is inputted or not when the mechanical part passes through End Limit sensor, use [AxmSignalReadLimit](#) API and return current state of End Limit signal to 0 or 1.

```
// Verify End Limit signal on axis 0.
long lAxisNo = 0;
DWORD uPositiveStatus, uNegativeStatus;
AxmSignalReadLimit (lAxisNo, &uPositiveStatus, &uNegativeStatus);
printf("+ELM : %x , -ELM : %x \n", uPositiveStatus, uNegativeStatus);
```

● Input Verification of Inposition Signal

If it is needed to verify whether inposition signal is inputted or not which is generated when the transfer part moved from motion controller to the commanded position, use [AxmSignalReadInpos](#) API.

```
// Use when verifying whether inposition signal is inputted or not.
long lAxisNo = 0;
DWORD uStatus;
AxmSignalReadInpos (lAxisNo, &uStatus);
printf("whether inposition on axis 0 exists or not : %x \n", uStatus);
```

● Input Verification of Alarm Signal

When the mechanical part does not move to the designated position and has malfunction, servo drive outputs alarm signal to motion controller. If it is needed to verify whether this signal is inputted or not, use [AxmSignalReadServoAlarm](#) API.

```
// Use when verifying whether alarm signal is inputted or not.
long lAxisNo = 0;
DWORD uStatus;
```

```
AxmSignalReadServoAlarm (lAxisNo, &uStatus);
printf("whether there is an alarm on axis 0 or not: %x \n", uStatus);
```

● Input Verification of Emergency Stop Signal

When the mechanical part stops operation urgently due to malfunction or abnormal operation, an emergency stop occurs. If it is needed to verify whether it is inputted or not, use [AxmSignalReadStop](#) API.

```
// Use when verifying whether emergency stop signal is inputted or not.
long lAxisNo = 0;
DWORD uStatus;
AxmSignalReadStop (lAxisNo, &uStatus);
printf("whether there is an emergency stop on axis 0 or not : %x \n", uStatus);
```

```
// Ex3_AXM_SettingMotionSignal.cpp : Defines the entry point for the console application.
// Set all sorts of signals of limit and alarm, and verify input state
```

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

void main(void)
{
    // Initialize library.
    // 7 means IRQ. IRQ is set automatically in PCI.
    DWORD Code = AxlOpen(7);
    if (Code == AXT_RT_SUCCESS)
    {
        printf("Library is initialized.\n");

        //Inspect whether motion module exists or not
        DWORD dwStatus;
        Code = AxmInfoIsMotionModule (&dwStatus);
        if(Code == AXT_RT_SUCCESS)
        {
            if(dwStatus == STATUS_EXIST)
            {
                printf("Motion module exists.\n");

                long lAxisNo = 0;

                // Pulse output method on axis 0 is to be TwoCwCcwHigh.Default :
                // OneLowHighLow
                DWORD uMethod = TwoCcwCwHigh;
                AxmMotSetPulseOutMethod (lAxisNo, uMethod);

                // Set Active level of -End limit and +End limit to HIGH. Default : HIGH
            }
        }
    }
}
```

```

DWORD uStopMode = 0;           // [0]emergency stop, [1]slowdown stop
DWORD uPositiveLevel = HIGH;
DWORD uNegativeLevel = HIGH;
AxmSignalSetLimit(IAxisNo, uStopMode, uPositiveLevel, uNegativeLevel);

// Set input level of inposition signal on axis 0.          Default : HIGH
DWORD uUse = HIGH;
AxmSignalSetInpos (IAxisNo, uUse);

// Set input level of alarm signal on axis 0.          Default : HIGH
uUse = LOW;
AxmSignalSetServoAlarm (IAxisNo, uUse);

// Set active input level of ESTOP on axis 0.
uStopMode = 0;      // [0] Emergency stop, [1] Slow-Down stop
DWORD uLevel = HIGH;
AxmSignalSetStop (IAxisNo, uStopMode, uLevel);

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

printf("[ESC] : Exit \n");
printf("*****\n");
printf("Verify input state of all sorts of signals\n\n");

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

while (!fExit)           // Infinite loop
{
    if (kbhit())         // Exit loop pressing any key.
    fExit = TRUE;

    // Verify End Limit signal on axis 0.
    AxmSignalReadLimit (IAxisNo, &uPositiveStatus, &uNegativeStatus);

    // Use when verifying whether inposition signal on axis 0 is inputted or not.
    AxmSignalReadInpos (IAxisNo, &ulnpositionStatus);

    // Use when verifying whether alarm signal is inputted or not
    AxmSignalReadServoAlarm (IAxisNo, &uAlarmStatus);

    // Use when verifying whether emergency stop signal is inputted or not.
    AxmSignalReadStop (IAxisNo, &uEStopStatus);

    printf("\rPEnd_Limit(%x), NEnd_Limit(%x), Inposition(%x), Alarm(%x),
           EStop(%x)", uPositiveStatus, uNegativeStatus, ulnpositionStatus,
           uAlarmStatus, uEStopStatus);

} //End of While()
}
else
printf ("AxmlInfoMotionModule() : ERROR ( NOT STATUS_EXIST )"

```

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

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

// End library.
if (AxlClose())
    printf("Library is ended.\n");
else
    printf("Library is not ended normally.\n");

}
```

Universal Input/Output Signal Reading and Output Writing

As mentioned before, universal input/output of 5-bit is provided. UIN0, UIN1 is already used for Home sensor and Z phase signal of Encoder among the available universal input signal by user arbitrarily, and 3-bit of the rest of UIN2~4 can be set arbitrarily by user. Also, among universal input signal, 2-bit of UOUT0, OUT1 is connected with Servo On, Alarm Clear basically, and 3-bit of the rest UOUT2~4 can be set arbitrarily by user.

State verification of universal input/output bit has a method that reads at once using [AxmSignalReadInpos](#) API or AxmReadOutput API, and that verifies the state of 5 bits from 0 to 4 individually using [AxmSignalReadInputBit](#) API or [AxmSignalReadOutputBit](#) API. And it returns 0 or 1 according to the signal state of the concerned bit..

```
// Verify the state of universal input bit signal of number 0 on axis 0.
long lAxisNo = 0;
long lBitNo = 0;
DWORD uOn;
AxmSignalReadInputBit (lAxisNo, lBitNo, &uOn);
printf("state of universal input bit signal of number 0 on axis 0 : %x",uOn);

// Verify the state of universal output bit signal of number 0 on axis 0.
AxmSignalReadOutputBit (lAxisNo, lBitNo, &uOn);
printf("state of universal output bit signal of number 0 on axis 0 : %x",uOn);
```

In order to use on/off of universal output signal, it can be set at once using AxmWriteOutput API, and can be set by bit unit using [AxmSignalWriteOutputBit](#) API. When universal output signal is HIGH, bit value is 1, and when is LOW, bit value is 0. In this manner, it is set bit of number 0~4 signal.

```
// Set universal output number 0 bit signal to 1.(Servo On)
long lAxisNo = 0;
long lBitNo = 0;
DWORD uOn = 1;
AxmSignalWriteOutputBit (long lAxisNo, long lBitNo, DWORD uOn);

// Set universal output number 0 bit signal to 0.(Servo Off)
uOn = 0;
AxmSignalWriteOutputBit (long lAxisNo, long lBitNo, DWORD uOn);
```

```
// Ex4_AXM_SettingUniversalO.cpp : Defines the entry point for the console application.
// Set and verify universal input/output.

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

void main(void)
{
    // Initialize library.
```

```

// 7 means IRQ. IRQ is set automatically in PCI.
DWORD Code = AxIOpen(7);
if (Code == AXT_RT_SUCCESS)
{
    printf("Library is initialized.\n");

    //Inspect whether motion module exists or not
    DWORD dwStatus;
    Code = AxmInfoIsMotionModule (&dwStatus);
    if(Code == AXT_RT_SUCCESS)
    {
        if(dwStatus == STATUS_EXIST)
        {
            printf("Motion module exists.\n");

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

            printf("[ESC] : Exit \n");
            printf("[0 ~ 4] : universal output signal On / Off \n");
            printf("*****\n");

            long lAxisNo = 0;
            long lBitNo;
            DWORD uOn, uStatusInput[5], uStatusOutput[5];
            BOOL fExit = FALSE;
            while (!fExit) // Infinite loop
            {
                if (kbhit()) // Press any key
                {
                    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, &uOn);
                    if(uOn == 1) {AxmSignalWriteOutputBit (lAxisNo, lBitNo, 0); }
                    else { AxmSignalWriteOutputBit (lAxisNo, lBitNo, 1); }
                }

                //State verification using API that verifies the information of individual
                //input/output bit
                for(int nBitNo =0; nBitNo <5; nBitNo++)
                {
                    AxmSignalReadInputBit (lAxisNo, nBitNo, &uStatusInput[nBitNo]);
                    AxmSignalReadOutputBit (lAxisNo, nBitNo, &uStatusOutput[nBitNo]);
                }
                printf("\ninput0(%x),1(%x),2(%x),3(%x),4(%x)\n");
            }
        }
    }
}

```

```

        /output0(%x),1(%x),2(%x),3(%x),4(%x)",
        uStatusInput[0], uStatusInput[1], uStatusInput[2] ,uStatusInput[3],
        utatusInput[4] ,uStatusOutput[0], uStatusOutput[1],atusOutput[2],
        uStatusOutput[3], uStatusOutput[4]);

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

// End library.
if (AxlClose())
    printf("Library is ended.\n");
else
printf("Library is not ended normally.\n");

}

```

Saving and Opening Setting Information

Every time when it is needed to draft program, it does not need to set every motion parameter and motion signal, and initialization work can be done easily by calling the setting file which is stored.

In order to save the information that is stored in the register of CAMC-IP, QI to file, use [AxmMotSaveParaAll](#) API. The API stores the information of all axis to file. If it is needed to store the setting information of all axis to SMC2V03.MOT file, follow this.

```

// Save setting information of all axis to SMC2V03.MOT file.
char *pFilename = "SMC2V03.MOT";
DWORD uOption = 0;
AxmMotSaveParaAll\_ (pFilename);

```

In order to set the information that is stored in file to the register of CAMC-IP, QI, use [AxmMotLoadParaAll](#) API. The API reads the information of all axis. If it is needed to read the setting information of all axis that is stored in SMC2V03.MOT file, follow this.

```

// Read and set the setting information of all axis stored in SMC2V03.MOT file.
char *pFilename = "SMC2V03.mot";
AxmMotLoadParaAll\_ (pFilename);

```

Initial Value of Internal Variable That File Is Set

No	Signal Related Variable	Parameter Description	Setting Value
0	AXIS_NO	Axis number	Setting axis number
0	PULSE_OUT METH OD	Setting of the pulse output method which is used as a servo pack order on the motion board	TwoCcwCwHigh
0	Enc[Encoder]	Setting of the input method of encoder signal	Sqr4Mode
0	InP[InPos]	Setting of the use of completion signal for servo pack position decision and Active Level	2
0	Alm[Alarm]	Setting of the use of servo pack alarm signal and Active Level	1
0	NLimit	Setting of the use of machine (-) limit sensor and Active Level	1
0	PLimit	Setting of the use of machine (+) limit sensor and Active Level	1
0	MinSpd	Setting of the initial drive velocity when the motor drives	1.0
0	MaxSpd	Setting of the maximum drive velocity when the motor drives	700000.0
0	HmSig	Setting of a signal to use as home sensor	4
1	HmLev	Setting of Active Level of home sensor	1
1	HmDir	Setting of the direction of initial search process during home sensor search	0
1	Zphas_Lev	Active level on encoder Z phase	1
1	Zphas_Use	Setting of the search of encoder Z phase after home sensor search	0
1	Stop Signal Mode	Mode when used ESTOP, SSTOP	0
1	Stop Signal level	Use level of ESTOP, SSTOP	0
1	VelFirst	Initial high speed search velocity during home search (unit of velocity is PPS[Pulses/sec] if Unit/pulse is 1/1)	10000.0
1	VelSecond	Velocity that comes out from the reverse direction	10000.0

		after the 1 st sensor search during home search	
1	VelThird	Velocity of research process after the 1 st sensor search during home search	2000.0
1	VelLast	Setting of the final search velocity during home search [Accuracy decision of home search]	100.0
2	AccFirst	Initial high speed search acceleration during home search (unit of acceleration is PPS[Pulses/sec^2] if Unit/pulse is 1/1)	40000.0
2	AccSecond	Acceleration that comes out from the reverse direction after the 1 st sensor search during home search (unit of acceleration is PPS[Pulses/sec^2] if Unit/pulse is 1/1)	40000.0
2	HClrTim	Waiting time before clear of Cmd, Act position after home search completion	1000.0
2	HOffset	Moving offset value during reset of machine home after home search completion	0.0
2	NSWLimit	Setting of (-) Software Limit value to apply after home search completion	0.0
2	PSWLimit	Setting of (+) Software Limit value to apply after home search completion	0.0
2	OenPulse	Setting of pulse number that requires for 1 rotation of motor [refer to servo pack electric gear ratio]	1.0
2	OneUnit	Moving amount that is moved for 1 rotation of motor [mm, degree, etc]	1
2	InitPosition	Position for initialization	200
2	InitVel	Velocity for the initialization (unit of velocity is PPS[Pulses/sec] if Unit/pulse is 1/1)	200
3	InitAccel	Acceleration for the initialization (acceleration unit of acceleration is PPS[Pulses/sec^2] if Unit/pulse is 1/1)	400
3	InitDecel	Deceleration for the initialization (deceleration unit of acceleration is PPS[Pulses/sec^2] if Unit/pulse is 1/1)	400
3	AbReMode	Absolute/relative mode for the initialization	0
3	ProfMode	Velocity profile mode for the initialization	4

Whenever program is written, it is easy to work initialization since there is no need to set all motion

parameter and motion signal, and just call the setting file that is stored .

Basically motion related parameters are stored in MotorParalP.mot file, but specified initial position, initial velocity, initial acceleration, initial deceleration, initial absolute and relative position mode, and initial profile mode are stored in file and used for the user's convenience. Those values can be set or called using [AxmMotSetParaLoader](#) [AxmMotGetParaLoad](#) API. Especially, those values are loaded automatically when other parameters initialize library, and used with store or call based on the necessity in contrast to the automatic store when library ends.

Parameter 28 – 31 can be set and brought using these APIs.

If you need to store to or callback from initial position, initial velocity, initial acceleration, and initial deceleration file for the specified axis, follow below.

```
/ Specified initial position, initial velocity, initial acceleration, initial deceleration, initial absolute and  
relative position mode,  
// Save initial profile mode to file..  
long lAxisNo = 0;  
double InitPos = 0;  
double InitVel = 200;  
double InitAccel = 400;  
double InitDecel = 400;  
AxmMotSetParaLoad(lAxisNo, InitPos, InitVel, InitAccel, InitDecel);
```

```
// Specified initial position, initial velocity, initial acceleration, initial deceleration, initial absolute and  
relative position mode,  
// Call initial profile mode from file.  
long lAxisNo = 0;  
double InitPos, InitVel, InitAccel, InitDecel;  
AxmMotGetParaLoad(lAxisNo, &InitPos, &InitVel, &InitAccel, &InitDecel);  
Printf("%d축 초기 위치(%f), 초기 속도(%f), 초기가속도(%f), 초기감속도(%f)",  
lAxisNo, InitPos, InitVel, InitAccel, InitDecel);
```

Motion Drive

Introduction of Motion Drive

- [1\) One Axis](#)
- [2\) Multi Axis](#)

Glossary of Motion Drive

- [1\) Servo On](#)
- [2\) Velocity Profile](#)
- [3\) Exit API at Start Point and Exit API at End Point](#)
- [4\) Relative Motion Drive and Absolute Motion Drive](#)

One-Axis Drive

- [1\) Position Drive](#)
- [2\) Velocity Drive](#)
- [3\) Signal Search Drive](#)
- [4\) Home Search Drive](#)
- [5\) Stop of Motion Drive](#)
- [6\) Verification of Motion Drive State](#)

Multi-axis Drive

- [1\) Multi-axis Position Drive](#)
- [2\) Interpolation Drive](#)
- [3\) Continuous Interpolation Drive](#)

Introduction of Motion Drive

Motion drive is to control a specific mechanical part that stopped or is moving using an intermediate of program. Motion drive can be divided into one-axis drive which drives one axis and multi-axis which drives more than two axes.

- [1\) One Axis](#)
- [2\) Multi Axis](#)

1) One Axis

● Position Drive

Position drive is the most general drive. This drive means, if the distance to move is given, movement with output of pulse that is the value calculated by velocity profile which is set on motion chip in motion module after required factors to move, such as moving velocity, acceleration/deceleration, or acceleration/deceleration time, are inputted.

This position drive is the mostly used drive API because general drive systems are designed to move the position that is already known repeatedly.

● Velocity Drive

Velocity drive is different from position drive which moves already assigned distance, and is a drive that moves with a constant velocity until stop command or stop signal is generated by limit sensor. This velocity drive API can be used for work that moves to (+) direction if (+) direction is pressed on jog button, and stops if the button is released.

● Signal Search Drive

Signal search drive is a drive that sets search target, such as limit signal or universal input/output signal previously, then moves with a constant velocity and stops when a set signal is detected. It can be used for work that is required to stop exactly on the position of end limit of home sensor of mechanical part.

● Home Search Drive

Home search drive means to search for home which is to be a datum of the coordination system. This drive API is provided to make home search easy by running an API after basic setting for home search.

2) Multi Axis

Generally, drive system has many cases that move several axes to a specified position at once, and for this drive, multi axis API is provided.

● Position Drive

This means a drive that moves with pulse output by the calculated value from motion chip in motion module after receiving factors that are required for movement of moving velocity, acceleration/deceleration,

or acceleration/deceleration time on 2 axes included in one module. This drive has a big difference from the drive using one axis drive API that two axes start to move at the same time.

● *Interpolation Drive*

This means a drive that moves from the current position to the designated position with interpolation along with a constant path of straight line or circular arc using 2 axes included in one module. Interpolation drive is generally divided into straight line interpolation drive and circular interpolation drive. Also there are general drive that follows the calculated path with velocity change of acceleration–constant velocity–deceleration starting from a point as well as continuous interpolation drive that decelerates lastly passing the designated many points without velocity change after starting from one point.

Glossary of Motion Drive

- [1\) Servo On](#)
- [2\) Velocity Profile](#)
- [3\) Exit API at Start Point and Exit API at End Point](#)
- [4\) Relative Motion Drive and Absolute Motion Drive](#)

1) Servo On

This means to turn On the servo drive, since output pulse that comes from motion card does not have influence on the actual drive if servo drive was not turned on, so servo drive must be turned On before drive.

First, ServoOn signal level has to be set using [AxmSignalSetServoOnLevel](#) API to turn ON/OFF servo drive. Most servo drives are Active HIGH, but some are not.

```
//Set ServoOn level to HIGH.
long lAxisNo = 0;
DWORD uLevel = HIGH;
AxmSignalSetServoOnLevel (lAxisNo, uLevel);
```

Next, Axm ServoOn API gives an on signal to servo drive and [AxmSignallsServoOn](#) API verifies if servo drive is on or not.

```
//Turn on servo drive on axis 0.
long lAxisNo = 0;
DWORD uUse = ENABLE;
AxmSignalServoOn (lAxisNo, uUse);
```

```
//Turn off servo drive on axis 0.
long lAxisNo = 0;
DWORD uUse = DISABLE;
AxmSignalServoOn(lAxisNo, uUse);
```

```
//Verifythat servo drive is on.
long lAxisNo = 0;
DWORD uUse;
AxmSignalsServoOn (lAxisNo, &uUse);
if(uUse == ENABLE) { printf("Servo drive is on."); }
else if(uUse == DISABLE) { printf("Servo drive is off."); }
```

2) Velocity Profile

There is a process if a user give a command to move through motion drive API, mechanical part starts the drive with Start Stop Speed that is previously set, accelerates to the inputted acceleration, maintains a constant velocity when it reaches the inputted velocity, decelerates to the inputted acceleration, and then stops. Velocity profile indicates how it accelerates and decelerates with which shape of velocity change when ac/decelerated.

AXL supports trapezoid ac/deceleration, and S-curve ac/deceleration, and can make an asymmetrical profile that has different velocity when ac/decelerated. Therefore, 5 types of velocity profile below are provided, and have setting values from 0 to 4 each.

Ac/Deceleration Profile	Setting Value (ProfileMode)
Symmetrical trapezoid ac/deceleration	0
Asymmetrical trapezoid ac/deceleration	1
Symmetrical Quasi-S Curve	2
Symmetrical S curve ac/deceleration	3
Asymmetrical S curve ac/deceleration	4

In library API, [AxmMotSetProfileMode](#) API is used to set the following.

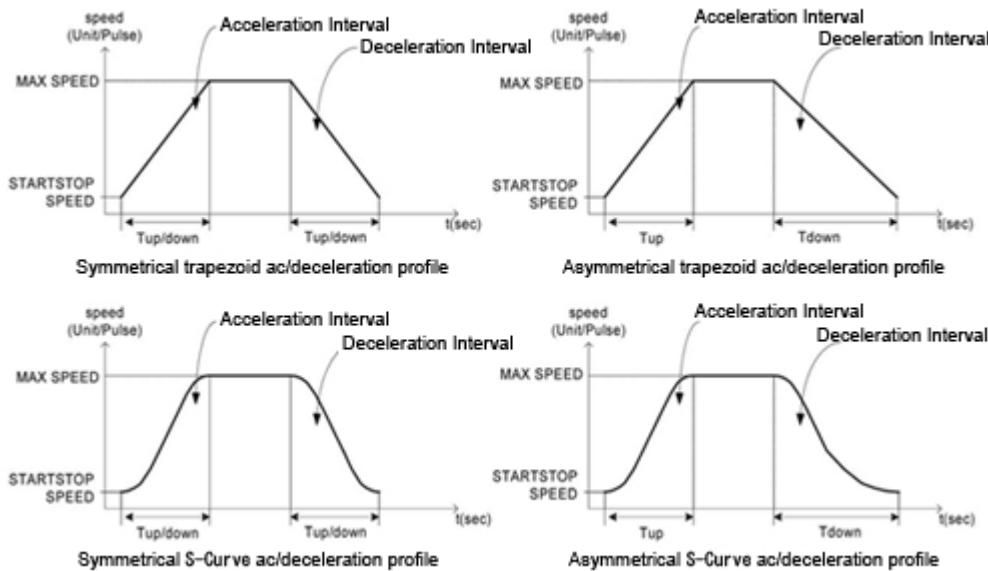
```
//Make velocity profile on axis 0 to the symmetrical S curve ac/deceleration.
long lAxisNo = 0;
DWORD uProfileMode = 4;
AxmMotSetProfileMode (lAxisNo, uProfileMode);
```

To verify the current set value, do the following.

```
// API that reads set ac/deceleration profile
long lAxisNo=0;
DWORD uProfileMode;
AxmMotGetProfileMode(lAxisNo, &upProfileMode);
switch(upProfileMode) {
    case 0 : printf("current velocity profile : symmetrical trapezoid ac/deceleration"); break;
    case 1 : printf("current velocity profile : asymmetrical trapezoid ac/deceleration"); break;
    case 2 : printf("current velocity profile : symmetrical Quasi-S Curve"); break;
    case 3 : printf("current velocity profile : symmetrical S curve ac/deceleration"); break;
    case 4 : printf("current velocity profile : asymmetrical S curve ac/deceleration"); break;
}
```

Following picture shows velocity profile of symmetrical trapezoid, asymmetrical trapezoid, symmetrical S

curve, and asymmetrical S curve. On the picture, velocity change rate (acceleration) of S curve velocity profile has a smooth movement compared to trapezoid velocity profile.



3) Exit API at Start Point and Exit API at End Point

Exit API at start point and exit API at end point are that means to carry out the next command on a program after getting out of an API when a user calls the API on the program.

● *Exit API at Start Point*

This means to go on to the next command on a program as soon as the first pulse is outputted from motion chip. In other words, it means to exit an API as soon as pulse output is started. This method has an advantage that does not occupy CPU on PC and goes straight to the next command on program. This means PC can carry out other process while motion chip takes pulse output for motion control.

● *Exit API at End Point*

This method is to go on to the next command on program getting out of the API after outputting pulse that is corresponding to the command from motion chip. That is, until the end of drive, CPU on PC is occupied and it does not go on the next command on program.

Matters that require attention for pulse exit API at end point are to set whether getting out or not with inposition signal from servo drive to motion controller, and it is default that gets out without inposition signal. If a user needs more accurate control, it is recommended to use inposition signal.

4) Relative Motion Drive and Absolute Motion Drive

It is required to classify whether given drive command moves to the relative coordination or absolute coordination. In the relative coordination system, it moves as much as the distance on the basis of current position for the distance that is given with API factor, and in the absolute coordination system, it moves as much as the distance which subtracts current position for the distance that is given. [AxmMotSetAbsRelMode](#)

API is used to set, and must be used after 0 point is set conducting home search by all means because absolute coordination is structured with existence of 0 point in absolute position drive.

If relative coordination or absolute coordination is needed to be set, do as following.

```
// Set to absolute position drive
long lAxisNo = 0;
DWORD uAbsRelMode = 0;      // [0] Absolute coordination, [1] Relative coordination
AxmMotSetAbsRelMode (lAxisNo, uAbsRelMode);
```

If current setting value needs to be verified, follow below.

```
// API which reads set moving mode
long lAxisNo=0;
DWORD uAbsRelMode;
AxmMotGetAbsRelMode (lAxisNo, &uAbsRelMode);
if(uAbsRelMode == 0) printf("In drive of absolute position");
elseif(uAbsRelMode == 1) printf("In drive of relative position");
```

One-Axis Drive

Generally in motion drive, the basic drive is one-axis drive. One-axis drive is divided into position drive, velocity drive, signal search drive, and home search drive.

- [1\) Position Drive](#)
- [2\) Velocity Drive](#)
- [3\) Signal Search Drive](#)
- [4\) Home Search Drive](#)
- [5\) Stop of Motion Drive](#)
- [6\) Verification of Motion Drive State](#)

1) Position Drive

Position drive is a command to move the mechanical part to an assigned position. Axm PositionMove API and [AxmMoveStartPos](#) API are used for position drive. These two APIs make to move to the set distance or position with inputted velocity and acceleration. [AxmMovePos](#) API is an exit API at end point and carries out the next command on program after drive exits, and [AxmMoveStartPos](#) API is an exit API at start point and goes to the next command on program as soon as movement starts.

If [AxmMovePos](#) API which is an exit API at end point is to be used, follow below. As it is mentioned before, use API that sets coordination system and velocity profile separately, and set moving amount, velocity, and acceleration. In case of symmetrical profile drive as following, deceleration which is inputted is ignored and acceleration which is inputted is applied when accelerated.

```
//Move axis 0 to 100 position on the absolute coordination with symmetrical S curve drive mode
//(velocity 100, acceleration 200)
long lAxisNo = 0;
```

```

DWORD uAbsRelMode = 0;
DWORD uProfileMode = 3;
AxmMotSetAbsRelMode (IAxisNo, uAbsRelMode);
AxmMotSetProfileMode (IAxisNo, uProfileMode);

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

```

If [AxmMoveStartPos](#) API which is an exit API at start point is need to be used, follow as below. API is carried out immediately after drive starts because of exit API at start point. If the next API is a drive API, the next drive command has to be given after drive exits verifying whether it is in drive or not using [AxmStatusReadInMotion](#) API by all means. This is because if it is in drive, all motion drive commands return errors and are not carried out. Here, [AxmStatusReadInMotion](#) API returns TRUE if it is in drive, and returns FALSE if not in drive.

```

// Move axis 0 to 100 position on the absolute coordination with asymmetrical S curve drive mode
// (velocity 100, acceleration 200, deceleration 100)
long IAxisNo = 0;
DWORD uAbsRelMode = 1;
DWORD uProfileMode = 4;
AxmMotSetAbsRelMode (IAxisNo, uAbsRelMode);
AxmMotSetProfileMode (IAxisNo, uProfileMode);

double dPosition = 100;
double dMaxVelocity = 100;
double dMaxAccel = 200;
double dMaxDecel = 100;
AxmMoveStartPos (IAxisNo, dPosition, dMaxVelocity, dMaxAccel, dMaxDecel);

DWORD uStatus;
AxmStatusReadInMotion (IAxisNo, &uStatus);
while(uStatus) {
    AxmStatusReadInMotion (IAxisNo, &uStatus);
}

```

```

// Ex5_AXM_PositionMove.cpp : Defines the entry point for the console application.
// Move as much as 100 to position drive API and move again as much as -100, and exit.

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

#define AXIS_0 0

void main(void)

```

```
{
    // Initialize library.
    // 7 means IRQ. IRQ is set automatically in PCI.
    DWORD Code = AxlOpen(7);
    if (Code == AXT_RT_SUCCESS)
    {
        printf("Library is initialized.\n");

        //Inspect whether motion module exists or not
        DWORD dwStatus;
        Code = AxmInfoIsMotionModule (&dwStatus);
        if(Code == AXT_RT_SUCCESS)
        {
            if(dwStatus == STATUS_EXIST)
            {
                printf("Motion module exists.\n");

                // Set Active level of +End limit and -End limit on axis 0 to HIGH.
                AxmSignalSetLimit(AXIS_0, 0, HIGH, HIGH);
                // Set input level of inposition signal on axis 0 to HIGH.
                AxmSignalSetInpos (AXIS_0, HIGH);
                // Set input level of alarm signal on axis 0 to LOW.
                AxmSignalSetServoAlarm (AXIS_0, LOW);
                // Set Active input level of ESTOP signal on axis 0 to HIGH.
                AxmSignalSetStop (AXIS_0, 0, HIGH);

                //Make ordered value on axis 0 to be mm unit.
                AxmMotSetMoveUnitPerPulse(AXIS_0, 10, 10000);
                //Set initial velocity on axis 0 to 1. Default : 1
                AxmMotSetMinVel(AXIS_0, 1);
                //Make pulse output method on axis 0 to TwoCwCcwHigh.
                AxmMotSetPulseOutMethod (AXIS_0, TwoCwCcwHigh);
                //Set encoder input method on designated axis to 4 multiplication.
                AxmMotSetEnclInputMethod (AXIS_0, ObverseSqr4Mode);

                //Set current position to be home.
                AxmStatusSetActPos(AXIS_0, 0.0);
                AxmStatusSetCmdPos(AXIS_0, 0.0);

                //Servo On
                AxmSignalServoOn(AXIS_0, ENABLE);

                printf("Drive symmetrical S curve of axis 0 up to 100 position with absolute
                    coordination (velocity 100, acceleration 200, exit API at end point)\n");
                printf("Drive starts pressing any key.\n");
                getch();
                AxmMotSetAbsRelMode (AXIS_0, 0);
                AxmMotSetProfileMode (AXIS_0, 3);
                AxmMovePos (AXIS_0, 100, 100, 200, 200);
            }
        }
    }
}
```

```
printf("Drive asymmetrical S curve of axis 0 up to -100 position with relative  
      coordination (velocity 100, acceleration 200, exit API at start point)\n");  
printf("Drive starts pressing any key.\n");  
getch();  
AxmMotSetAbsRelMode (AXIS_0, 1);  
AxmMotSetProfileMode (AXIS_0, 4);  
AxmMoveStartPos (AXIS_0, -100, 100, 200, 200);  
  
//Wait if it is in drive currently.  
DWORD uStatus;  
AxmStatusReadInMotion (AXIS_0, &uStatus);  
while(uStatus) {  
    AxmStatusReadInMotion (AXIS_0, &uStatus);  
}  
  
}  
else  
    printf ("AxmlInfoIsMotionModule() : ERROR ( NOT STATUS_EXIST )  
           code 0x%x\n",Code);  
}  
else  
    printf ("AxmlInfoIsMotionModule() : ERROR ( Return FALSE )   code 0x%x\n",Code);  
  
}  
else  
    printf ("AxlOpen() : ERROR   code 0x%x\n",Code);  
  
// Exit library.  
if (AxlClose())  
    printf("Library is exited.\n");  
else  
    printf("Library is not exited normally.\n");  
}
```

2) Velocity Drive

Velocity drive is a type of drive that moves continuously until stop command is given with fixed velocity, acceleration, and acceleration time. Also, velocity drive API is a pulse exit API at start point that gets out of start point which starts pulse. For the velocity drive, [AxmMoveVel](#) API is used, and the API is a command to move with a specified straight velocity. Drive direction is determined by the symbol of velocity value.

```
// Start drive with symmetrical S curve velocity profile of axis 0 to (+) direction with velocity of 100.
long lAxisNo = 0;
DWORD uProfileMode = 3;
AxmMotSetProfileMode (lAxisNo, uProfileMode);

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

```
// Start drive with symmetrical trapezoid velocity profile of axis 0 to (-) direction with velocity of 100.
long lAxisNo = 0;
DWORD uProfileMode = 0;
AxmMotSetProfileMode (lAxisNo, uProfileMode);

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

Like this, if velocity drive API is started, it drives until stop command is given. Stop of drive will be discussed in detail later, so do as follows.

```
//Make axis 0 drive to stop with deceleration of set deceleration.
long lAxisNo = 0;
AxmMoveSStop(lAxisNo);
```

```
// Ex6_AXM_VelocityMove.cpp : Defines the entry point for the console application.
// Drive by using velocity drive API and exit by stop command.
```

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

#define AXIS_0 0

void main(void)
{
```

```

// Initialize library.
// 7 means IRQ. IRQ is set automatically in PCI.
DWORD Code = AxIOpen(7);
if (Code == AXT_RT_SUCCESS)
{
    printf("Library is initialized.\n");

    //Inspect whether motion module exists or not
    DWORD dwStatus;
    Code = AxmInfoIsMotionModule (&dwStatus);
    if(Code == AXT_RT_SUCCESS)
    {
        if(dwStatus == STATUS_EXIST)
        {
            printf("Motion module exists.\n");

            // Set Active level of +End limit and -End limit on axis 0 to HIGH.
            AxmSignalSetLimit(AXIS_0, 0, HIGH, HIGH);
            // Set input level of inposition signal on axis 0 to HIGH.
            AxmSignalSetInpos (AXIS_0, HIGH);
            // Set input level of alarm signal on axis 0 to LOW.
            AxmSignalSetServoAlarm (AXIS_0, LOW);
            // Set Active input level of ESTOP signal on axis 0 to HIGH.
            AxmSignalSetStop (AXIS_0, 0, HIGH);

            //Make ordered value on axis 0 to be mm unit.
            AxmMotSetMoveUnitPerPulse(AXIS_0, 10, 10000);
            //Set initial velocity on axis 0 to 1. Default : 1
            AxmMotSetMinVel(AXIS_0, 1);
            //Make pulse output method on axis 0 to TwoCwCcwHigh.
            AxmMotSetPulseOutMethod (AXIS_0, TwoCwCcwHigh);
            //Set encoder input method on designated axis to 4 multiplication.
            AxmMotSetEnclnputMethod (AXIS_0, ObverseSqr4Mode);

            //Set current position to be home.
            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] : Start velocity drive to (+) direction \n");
            printf("[2] : Start velocity drive to (-) direction \n");
            printf("[3] : STOP \n");
            printf("*****\n");

            DWORD uStatus;
            BOOL fExit = FALSE;
            while (!fExit) // Infinite loop
            {
                if (kbhit()) // Press any key

```

```

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

        case '1': // If not in drive, drive continuously symmetrical
                   trapezoid velocity profile to (+) direction.
            AxmStatusReadInMotion (AXIS_0, &uStatus);
            if(!uStatus)
            {
                AxmMotSetProfileMode (AXIS_0, 0);
                AxmMoveVel (AXIS_0, 100, 200, 200);
            }
            break;

        case '2': // If not in drive, drive continuously symmetrical
                   trapezoid velocity profile to (-) direction.
            AxmStatusReadInMotion (AXIS_0, &uStatus);
            if(!uStatus)
            {
                AxmMotSetProfileMode (AXIS_0, 3);
                AxmMoveVel (AXIS_0, -100, 200, 200);
            }
            break;

        case '3': //If in drive, make slow down stop.
            AxmStatusReadInMotion (AXIS_0, &uStatus);
            if(uStatus)
                AxmMoveSStop(AXIS_0);
            break;
        }
    }
} //End of While()
}
else
    printf ("%AxmlnfoIsMotionModule() : ERROR ( NOT STATUS_EXIST )
           code 0x%x\n",Code);
}
else
    printf ("%AxmlnfoIsMotionModule() : ERROR ( Return FALSE )
           code 0x%x\n",Code);
}
else
    printf ("AxlOpen() : ERROR   code 0x%x\n",Code);

// Exit library.
if (AxlClose())
    printf("Library is exited.\n");
else
    printf("Library is not exited normally.\n");

```

```
}
```

3) Signal Search Drive

Signal search drive is a drive that sets in advance search target such as limit signal or universal input/output signal, and that stops when a set signal is detected moving with a constant velocity. Signal search drive has [AxmMoveSignalSearch](#) API that decelerates or makes an emergency stop when a specified signal is detected, and [AxmMoveSignalCapture](#) API that stores the position when a signal is detected. There are 8 signals that can be detected showing on the table below, each signal has Up Edge and Down Edge, so there are total of 16 signals.

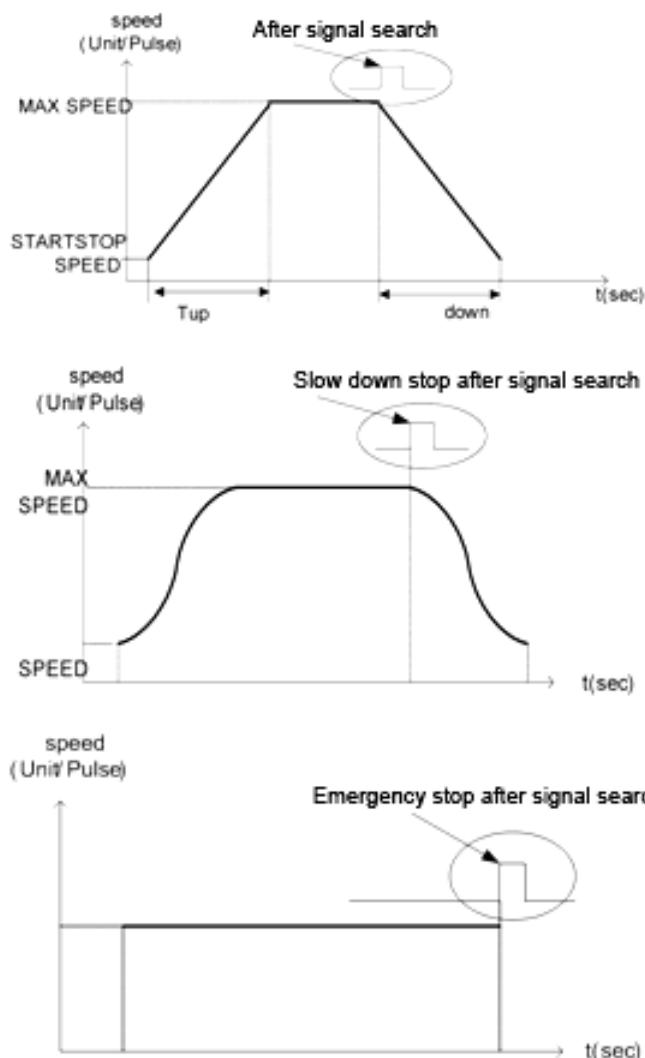
Define	value	Search Signal
PosEndLimit	0x0	+Elm(End limit) + direction limit sensor signal
NegEndLimit	0x1	-Elm(End limit) - direction limit sensor signal
PosSloLimit	0x2	+Slm(Slow Down limit) signal – Not in used
NegSloLimit	0x3	-Slm(Slow Down limit) signal – Not in used
HomeSensor	0x4	IN0(ORG) home sensor signal
EncodZPhase	0x5	IN1(Z phase) Encoder Z phase signal
Unilnput02	0x6	IN2(universal) universal input number 2 signal
Unilnput03	0x7	IN3(universal) universal number 3 signal

For the signal search drive, [AxmMoveSignalSearch](#) API is used, and this API is an API that makes an emergency stop or slow down stop with search of a specified input signal edge, has input of edge direction of input signal to search and input signal that is selected, drive direction, velocity and acceleration, and stop method. Drive direction is determined by the symbol of velocity value. Epically, since this API starts deceleration after corresponding signal is detected, in order to search a correct signal position, it has to be used that makes an emergency stop at the corresponding signal searching signal with low velocity and moving to the reverse direction after slow down stop in signal search drive.

```
// Search Up Edge of home sensor moving to - direction.
long lAxisNo = 0;
double dMaxVelocity = -100;
double dMaxAccel = 200;
long lDetectSignal = HomeSensor;
long lSignalEdge = UP_EDGE;
long lSignalMethod = 0; // [0]Slow down stop, [1]Emergency stop
AxmMoveSignalSearch (lAxisNo, dMaxVelocity, dMaxAccel, lDetectSignal, lSignalEdge, lSignalMethod);

// Search Down Edge of home sensor moving to + direction, and make an emergency stop.
double dMaxVelocity = 20;
double dMaxAccel = 0;
```

```
long lDetectSignal = HomeSensor;
long lSignalEdge = DOWN_EDGE;
long lSignalMethod = 1; // [0]Slow down stop, [1]Emergency stop
AxmMoveSignalSearch (lAxisNo, dMaxVelocity, dMaxAccel, lDetectSignal,
lSignalEdge ,lSignalMethod);
```



If it is simply needed not only to stop at the position that is detected signal but to verify the position of corresponding signal, use [AxmMoveSignalCapture](#) API. If this API is called, it starts to search a corresponding signal, and if signal is detected, it stores the corresponding position, and the stored position can be verified using [AxmMoveGetCapturePos](#) API. Stored position is one of Command Position or Actual Position API, and the distance can be verified using the home point.

```
// Search Up Edge of +ELM moving to + direction.
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]Slow down stop, [1]Emergency stop
AxmMoveSignalCapture (lAxisNo, dMaxVelocity, dMaxAccel, lDetectSignal, lSignalEdge, lTarget,
lSignalMethod);

//Verify +ELM sensor position.
double dCapPosition;
AxmMoveGetCapturePos (lAxisNo, &dCapPosition);
printf(" +ELM sensor position : %f", dCapPosition);

```

```

// Ex7_AXM_SignalSearchDrive.cpp : Defines the entry point for the console application.
// Move to limit sensor and home sensor using signal search drive API.

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

#define AXIS_0 0

void main(void)
{
    // Initialize library.
    // 7 means IRQ. IRQ is set automatically in PCI.
    DWORD Code = AxlOpen(7);
    if (Code == AXT_RT_SUCCESS)
    {
        printf("Library is initialized.\n");

        //Inspect whether motion module exists or not
        DWORD dwStatus;
        Code = AxmlInfoIsMotionModule (&dwStatus);
        if (Code == AXT_RT_SUCCESS)
        {
            if(dwStatus == STATUS_EXIST)
            {
                printf("Motion module exists.\n");
                // Set Active level of +End limit and -End limit on axis 0 to HIGH.
                AxmSignalSetLimit(AXIS_0, 0, HIGH, HIGH);
                // Set input level of inposition signal on axis 0 to HIGH.
                AxmSignalSetInpos (AXIS_0, HIGH);
                // Set input level of alarm signal on axis 0 to LOW.
                AxmSignalSetServoAlarm (AXIS_0, LOW);
                // Set Active input level of ESTOP signal on axis 0 to HIGH.
                AxmSignalSetStop (AXIS_0, 0, HIGH);

                //Make ordered value on axis 0 to be mm unit.
                AxmMotSetMoveUnitPerPulse(AXIS_0, 10, 10000);
                //Set initial velocity on axis 0 to 1. Default : 1

```

```

AxmMotSetMinVel(AXIS_0, 1);
//Make pulse output method on axis 0 to TwoCwCcwHigh.
AxmMotSetPulseOutMethod (AXIS_0, TwoCcwCcHigh);
//Set encoder input method on designated axis to 4 multiplication.
AxmMotSetEnclnputMethod (AXIS_0, ObverseSqr4Mode);

//Set current position to be home.
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] : Trapezoid slow down stop after moving to + direction until Up Edge of
+ELM sensor is detected \n");
printf("[2] : S-curve slow down stop after moving to - direction until Up Edge
of -ELM sensor is detected \n");
printf("[3] : Emergency stop after moving to + direction until IN/OUT5(ORG)
DownEdge is detected \n");
printf("*****\n");

DWORD uPositiveStatus, uNegativeStatus, uHomeStatus;
BOOL fExit = FALSE;
while (!fExit) // Infinite loop
{
    if (kbhit()) // press any key
    {
        int ch = getch();
        switch (ch)
        {
            case 27: // Esc key
                fExit = TRUE;
                AxmSignalServoOn(AXIS_0, DISABLE); //Servo Off
                break;

            case '1':
                // Trapezoid drive with velocity of 100 and acceleration and
                // deceleration of 200 on axis 0,
                // Slow down stop after driving to + direction until Up Edge
                // of +ELM sensor is detected.
                AxmMotSetProfileMode (AXIS_0, 0);
                AxmMoveSignalSearch (AXIS_0, 100, 200, PosEndLimit
                , UP_EDGE,0);
                break;

            case '2':
                // S-curve drive with velocity of 100 and acceleration and
                // deceleration of 200 on axis 0,
                // Slow down stop after driving to - direction until Up Edge
                // of -ELM sensor is detected.
        }
    }
}

```

```

        AxmMotSetProfileMode (AXIS_0, 3);
        AxmMoveSignalSearch (AXIS_0, -100, 200, NegEndLimit,
                                  UP_EDGE,0);
        break;

        case '3':
            // Emergency stop after driving to + direction until Down
            Edge of home sensor is detected with velocity of 20 on axis 0.
            AxmMoveSignalSearch (AXIS_0, 20, 0, HomeSensor,
                                      DOWN_EDGE,1);
            break;
        }
    }
// Verify signal state of each sensor.
AxmSignalReadLimit (AXIS_0, &uPositiveStatus, &uNegativeStatus);
AxmSignalReadInputBit (AXIS_0, 0, &uHomeStatus);
printf("Wr Signal state : 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);

// Exit library.
if (AxlClose())
    printf("Library is exited.\n");
else
printf("Library is not exited normally.\n");

}

```

4) Home Search Drive

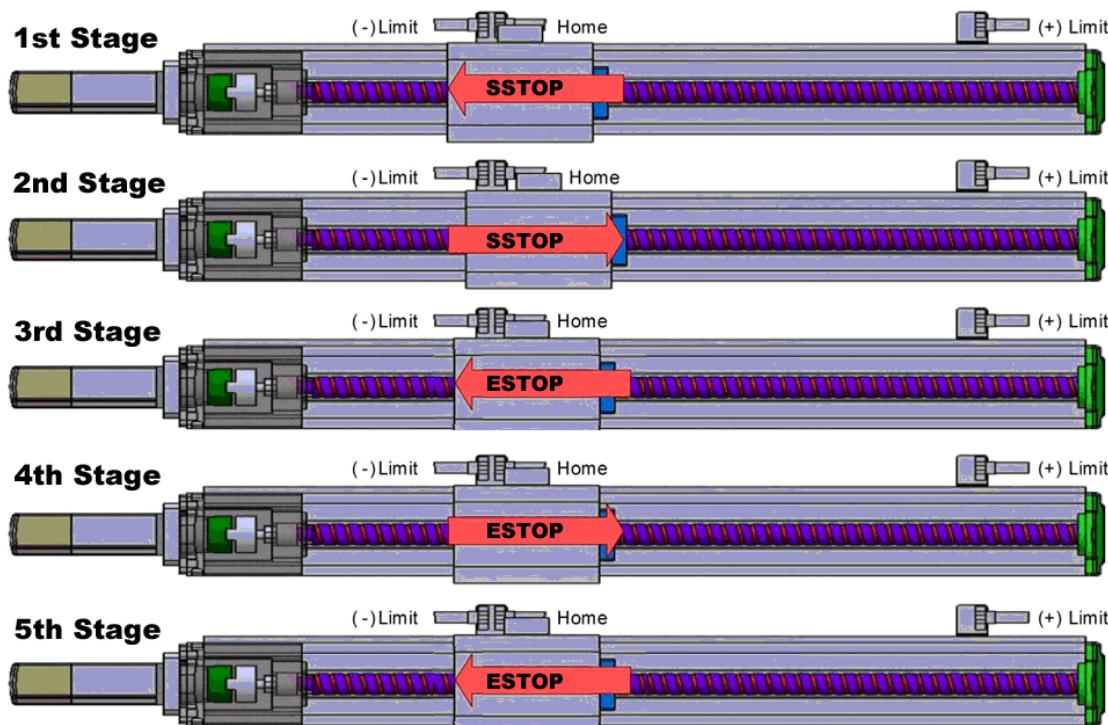
Even if the work for motion drive is ready with procedure of hardware initialization, parameter setting, and servo on, the state is that only the direction of coordination system is set and home which is to be a datum of coordination system is not set yet. Accurate home has to be ensured for the movement of mechanical part to a specified position which is a user designated. Home search method generally has two methods that are using signal search drive API and home search API. In order to carry out home search using signal search, since quite troublesome works are carried out, home search API is provided to make easier those process. For the use of home search API, set required data for home search using [AxmHomeSetMethod](#) and

[AxmHomeSetMethod](#) API, and start home search calling [AxmHomeSetStart](#) API.

Since home search API is operated generating individual threads, if home search of various axes has to be carried out, use the API listing each home search command

● Home Search Method and Velocity Setting

Home search generates various signal search order by mechanical part. But home search API of AXL has 7 stage of home search sequence below, and if home search sequence is compared with a factor of API for home search, it shows on the picture below.



1st Stage : slow down stop after searching Rising Edge signal of home sensor moving to – direction (if home sensor is already in HIGH state, omit 1st stage)

2nd Stage : slow down stop after searching Falling Edge signal of home sensor moving to + direction

3rd Stage : emergency stop after searching Rising Edge signal of home sensor moving to – direction

< In case of searching Z phase >

4th Stage : emergency stop after searching Falling Edge signal of Z phase moving to + direction

5th Stage : emergency stop after searching Rising Edge signal of Z phase moving to – direction

< In case of searching Home sensor >

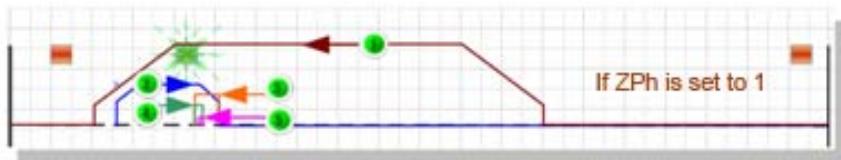
4th Stage : emergency stop after searching Falling Edge signal of home sensor moving to + direction

5th Stage : emergency stop after searching Rising Edge signal of home sensor moving to – direction

6th Stage : move additionally by Offset value if it exists

7th Stage : set to 0 after waiting for a while until vibration is decreased before setting to 0

● Understanding of Home Search Velocity Setting by Step



① Use of VelFirst, AccFirst

High speed search of home sensor to HmDir direction and slow down stop with HAccF deceleration

② Use of VelSecond, AccSecond

Search Down Edge of home sensor to HmDir reverse direction and slow down stop with HAccF deceleration

③ Use of VelThird

Search Up Edge of home sensor to HmDir direction and emergency stop

④ Use of VelLast

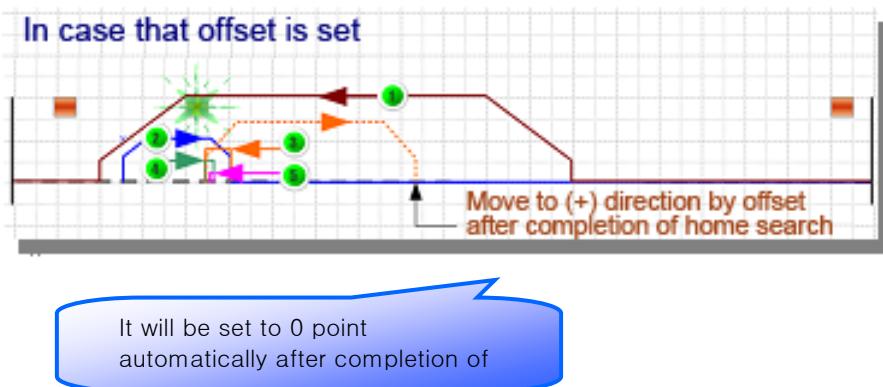
Search Down Edge on Z phase to HmDir reverse direction and emergency stop

⑤ Use of VelFirst

Search Up Edge on Z phase to HmDir direction and emergency stop

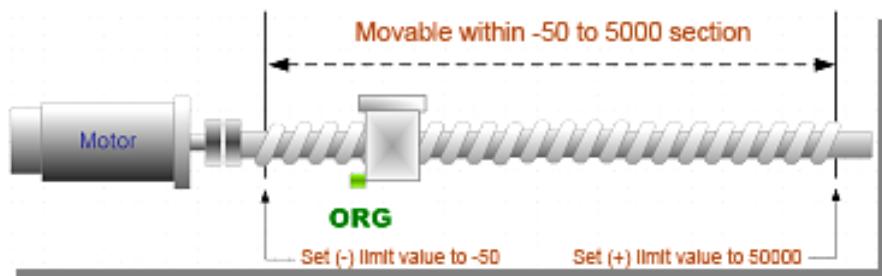
● Understanding of Offset Value Setting

If it needs to be set home with movement to a specified position after completion of mechanical home sensor search, and ia value is set on HOffset, home will be reset after completion of movement by corresponding offset automatically after completion of home sensor search. Here, be careful to set offset within the range of (+) or (-)Limit. Otherwise, home search can not be carried out regularly.



● Understanding of Software Limit Value Setting

If Soft Limit function is already set, Soft Limit will be re-set after completion of home search regularly. Soft Limit is used if Limit sensor is not installed mechanically or Limit needs to be changed by condition. Here, be careful that Soft Limit is set regularly when PLimit(+) value is bigger than NLimit(-). If NSwLimit value is equal to or bigger than PSwLimit, Soft Limit will not be set.



● Variable Description of Home Search API

Set Variable Related to Home Search	Variable Description
HmDir	Set initial search progress direction when home sensor detects Example) 1: (+)direction, 0: (-)direction
HmSig	Set a signal to be used as home sensor Example) Home sensor(4), +Limit sensor(0), etc.
Zphas	Set whether detected of encoder Z phase after home sensor detect

	Example) 1: To use, 0: Not to use Use to increase home search velocity and accuracy by, on the 4th step, searching Down Edge of Z phase and, on the 5th step, researching Up Edge of Z phase
HClrTim	Waiting time before clear of Command, Actual(Encoder) position after completion of home search
HOffset	Moving offset value during mechanical home reset after completion of home search
VelFirst	Initial high speed search velocity during home search (if initial home sensor did not detected)
VelSecond	Velocity coming out from the reverse direction after 1st sensor search during home search
VelThird	Velocity progressing research after 1st sensor search during home search
VelLast	Final search velocity setting [determination of home search accuracy] during home search
AccFirst	Initial high speed search acceleration during home search
AccSecond	Acceleration coming out from the reverse direction after 1st sensor search during home search

Note : When the level is not set correctly, it can operate to + direction even though it is set to - direction, and it may be a problem to find home.

```
// API that sets home search method
long lAxisNo = 0; //Axis number that home search will be carried out
long nHmDir = 0; //Moving direction of home search 1st, input (-)direction
DWORD uHomeSignal = HomeSensor; //Use signals from signal search drive
DWORD uZphas = 1; // Whether Z phase detected, detect if 1, do not if 0
double dHomeClrTime = 2000.0; // Waiting time to set home search Enc value
double dHomeOffset = 0.0; // Additional movement by offset value after home search
AxmHomeSetMethod (lAxisNo,nHmDir,uHomeSignal,uZphas,dHomeClrTime,dHomeOffset);

// Set velocity that will be used during home search
double dVelFirst = 100; //Velocity in 1st stage
double dVelSecond = 20; //Velocity in 2nd stage
double dVelThird = 10; //Velocity in 3rd stage
double dVelLast = 10; //Velocity for index search and accurate search (apply when Offset value moves)
double dAccFirst = 200; //Acceleration in 1st stage
double dAccSecond = 40; //Acceleration in 2nd stage
AxmHomeSetVel(lAxisNo, dVelFirst, dVelSecond, dVelThird, dVelLast, dAccFirst, dAccSecond);
```

Here, velocity of home search can be high speed in 1st stage, but it has to be a very small value in 2nd and 3rd stage because accurate work is carried out. Also, acceleration can be selected properly in 1st stage,

but it is desirable to set quite big value in 2nd stage.

Sometimes, home search sequence that is provided above based on the state of mechanical part cannot be used, and in this case, make and use home search sequence that meets its own mechanical part using signal search drive API.

● Start Home Search Drive

So far set values that need to be set for home search are mentioned, but [AxmHomeSetStart](#) API is to carry out home search actually. If this API is called, home search is started on the assigned axis by setting value that is set previously.

```
//Start home search on axis 0.
long lAxisNo = 0;
AxmHomeSetStart (lAxisNo);
```

● Verification of Home Search Progress Rate

Once home search is carried out, progress rate of current home search can be verified. Progress rate is verified using [AxmHomeGetRate](#) API, and if it passes a specified stage, value is increased, and if it becomes 100 starting from 0, it means home search is exited.

```
//Verify progress rate of home search on axis 0.
long lAxisNo = 0;
DWORD uHomeMainStepNumber, uHomeStepNumber;
AxmHomeGetRate(lAxisNo, &uHomeMainStepNumber , &uHomeStepNumber);
printf("current progress rate : %d ", uHomeStepNumber);
// uHomeMainStepNumber is used to verify whether master,slave operates or not, in case of gantry.
// In case of master axis movement, it returns 0, and in case of slave movement, returns 10.
```

● Completion and Stop of Home Search

If home search API is carried out, home search is carried out making a new thread. If home search is exited, the result of home search needs to be returned. To verify the result of home search, [AxmHomeGetResult](#) API is used and the result below is returned.

```
//Return the result of home search.
long lAxisNo = 0;
DWORD uHomeResult;
AxmHomeGetResult (lAxisNo, &uHomeResult);
```

Define	Description
HOME_ERR_UNKNOWN	When home search is started with an unknown axis number

HOME_ERR_GNT_RANGE	If home search result of master and slave axis of gantry drive axis is off OffsetRange that is set
HOME_ERR_USER_BREAK	If user carried out stop command during home search
HOME_ERR_VELOCITY	If home search velocity setting is not set
HOME_ERR_AMP_FAULT	If servo pack alarm is occurred during home search
HOME_ERR_NEG_LIMIT	If (-) limit sensor is detected during home sensor search with (+) direction
HOME_ERR_POS_LIMIT	If (+) limit sensor is detected during home sensor search with (-) direction
HOME_ERR_NOT_DETECT	If home sensor is not detected
HOME_SEARCHING	If home search is in progress currently
HOME_SUCCESS	If home search is exited successfully

5) Stop of Motion Drive

Stop of motion, which is the final stage of motion drive, has emergency stop, slow down stop, and stop method with specified deceleration. It stops motion using each API of [AxmMoveEStop](#) for emergency stop realization, [AxmMoveSStop](#) for slow down stop realization, and [AxmMoveStop](#) for stop with specified deceleration. Particularly, [AxmMoveStop](#) is to be set deceleration, and if this API is called, existing inputted deceleration is to be ineffective and it stops with newly inputted deceleration. If the deceleration is to be 0, it has same function with [AxmMoveEStop](#), and if it is set with original deceleration, has same function with [AxmMoveSStop](#).

```
// Make an emergency stop of motion drive on axis 0.
long lAxisNo = 0;
AxmMoveEStop (lAxisNo);
```

```
// Make slow down stop of motion drive on axis 0.
long lAxisNo = 0;
AxmMoveSStop(lAxisNo);
```

```
// Stop motion drive with deceleration of 100 on axis 0.
long lAxisNo = 0;
double dMaxDecel = 100;
AxmMoveStop (lAxisNo, dMaxDecel);
```

```

// Ex8_AXM_HomeSearchDrive.cpp : Defines the entry point for the console application.
// Carry out home search using home search API.

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

#define AXIS_0 0

void main(void)
{
    // Initialize library.
    // 7 means IRQ. IRQ is set automatically in PCI.
    DWORD Code = AxlOpen(7);
    if (Code == AXT_RT_SUCCESS)
    {
        printf("Library is initialized.\n");

        //Inspect whether motion module exists or not
        DWORD dwStatus;
        Code = AxmInfoIsMotionModule (&dwStatus);
        if(Code == AXT_RT_SUCCESS)
        {
            if(dwStatus == STATUS_EXIST)
            {
                printf("Motion module exists.\n");

                // Set Active level of +End limit and -End limit on axis 0 to HIGH.
                AxmSignalSetLimit(AXIS_0, 0, HIGH, HIGH);
                // Set input level of inposition signal on axis 0 to HIGH.
                AxmSignalSetInpos (AXIS_0, HIGH);
                // Set input level of alarm signal on axis 0 to LOW.
                AxmSignalSetServoAlarm (AXIS_0, LOW);
                // Set Active input level of ESTOP signal on axis 0 to HIGH.
                AxmSignalSetStop (AXIS_0, 0, HIGH);

                //Make ordered value on axis 0 to be mm unit.
                AxmMotSetMoveUnitPerPulse(AXIS_0, 10, 10000);
                //Set initial velocity on axis 0 to 1. Default : 1
                AxmMotSetMinVel(AXIS_0, 1);
                //Make pulse output method on axis 0 to TwoCwCcwHigh.
                AxmMotSetPulseOutMethod (AXIS_0, TwoCwCcwHigh);
                //Set encoder input method on designated axis to 4 multiplication.
                AxmMotSetEnclInputMethod (AXIS_0, ObverseSqr4Mode);

                //Servo On
                AxmSignalServoOn(AXIS_0, ENABLE);

                // Set home search method
                long lAxisNo = 0;           // Axis number that home search will be carried out
                long nHmDir = -1;          // Moving direction of home search 1st, input (-)direction
                DWORD uHomeSignal = HomeSensor; // Use signals from signal search drive
            }
        }
    }
}

```

```

DWORD uZphas = 1;           // Whether Z phase detected, detect if 1, do not if 0
double dHomeClrTime = 2000.0; // Waiting time to set home search Enc value
double dHomeOffset = 0.0;    // Additional movement by offset value after home search
AxmHomeSetMethod(AXIS_0, nHmDir, uHomeSignal, uZphas, dHomeClrTime,
dHomeOffset );

// Set velocity that will be used during home search
double dVelFirst = 100;      // Velocity in 1st stage
double dVelSecond = 20;       // Velocity in 2nd stage
double dVelThird = 10;        // Velocity in 3rd stage
double dvelLast = 10;         // Velocity for index search and accurate search
                             // (apply when Offset value moves)
double dAccFirst = 200;       // Acceleration in 1st stage
double dAccSecond = 40;       // Acceleration in 2nd stage
AxmHomeSetVel(AXIS_0, dVelFirst, dVelSecond, dVelThird, dvelLast, dAccFirst,
dAccSecond);

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

printf("[ESC] : Exit \n");
printf("[1] : Start home search \n");
printf("[2] : Exit home search \n");
printf("*****\n");

DWORD uHomeResult;
DWORD uHomeMainStepNumber , uHomeStepNumber;
BOOL fExit = FALSE;
while (!fExit)               // Infinite loop
{
    if (kbhit())            // Press any key
    {
        int ch = getch();
        switch (ch)
        {
            case 27:          // Esc key
                fExit = TRUE;
                AxmSignalServoOn(AXIS_0, DISABLE);    //Servo Off
                break;

            case '1'://Start home search if home search is not in progress on axis 0.
                AxmHomeGetResult(AXIS_0, &uHomeResult);
                if(uHomeResult)
                    //Home search is in progress : 0, Home search is completed : 1
                {
                    AxmHomeSetStart(AXIS_0);
                    printf("\nStart home search.\n");
                }
                break;

            case '2': //Exit home search if home search on axis 0 is in progress.
                AxmHomeGetResult(AXIS_0, &uHomeResult);
                if(uHomeResult == HOME_SEARCHING)
                    // Home search is in progress : 0
        }
    }
}

```

```

        {
            AxmMoveSStop(AXIS_0);
            printf("WnExit home search.Wn");
        }
        break;
    }
}

//Verify current home search progress rate
AxmHomeGetRate(AXIS_0, &uHomeMainStepNumber, &uHomeStepNumber);
printf("Wr Home search progress rate : %d      ", uHomeStepNumber);

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

}

else
printf ("AxlOpen() : ERROR   code 0x%xWn",Code);

// Exit library.
if (AxlClose())
    printf("Library is exited.Wn");
else
    printf("Library is not exited normally.Wn");

}

```

6) Verification of Motion Drive State

If motion drive is carried out, mechanical part moves as much as the pulse value due to pulse generation from motion controller. In this motion drive, it is a drive and stop state monitoring that verifies information of current position or velocity, drive state, cause of stop, and so on.

● *Verification of Current Position and Ordered Position*

From motion controller, it can be verified how much of movement is commanded currently and what the actual moving distance of mechanical part which is gotten feedback from motor encoder is. When an actual position of specified axis is verified, follow below using [AxmStatusGetActPos](#) API.

```
//Verify current position on axis 0.
long lAxisNo = 0;
```

```
double dPosition;
AxmStatusGetActPos (lAxisNo, &dPosition);
printf("Actual position on axis 0 : %f", dPosition);
```

When command position is verified, do the following using [AxmStatusGetCmdPos](#) API.

```
//Verify ordered position on axis 0.
long lAxisNo = 0;
double dPosition;
AxmStatusGetCmdPos (lAxisNo, &dPosition);
printf("Ordered position on axis 0 : %f", dPosition);
```

When an error value which is the difference between command position and actual position is verified, follow below using [AxmStatusReadPosError](#) API.

```
//Verify difference between ordered position and actual position on axis 0.
long lAxisNo = 0;
double dError;
AxmStatusReadPosError (lAxisNo, &dError);
printf("Difference between ordered position and actual position on axis 0 : %f", dError);
```

● Verification of Current Velocity

Velocity means that obtains the velocity of current drive of mechanical part. Use [AxmStatusReadVel](#) API showing below when current drive velocity is verified.

```
//Verify current velocity on axis 0.
long lAxisNo = 0;
double dVelocity;
AxmStatusReadVel (lAxisNo, &dVelocity);
printf("Current velocity on axis 0 : %f", dVelocity);
```

● Verification Whether Motion Drive Exists

This is to verify pulse output state of assigned axis, and simply it means to verify that mechanical part is moving or exited drive. When it is needed to verify a movement of specified axis, use [AxmStatusReadInMotion](#) API.

```
//Verify if axis 0 drives.
long lAxisNo = 0;
DWORD uStatus;
AxmStatusReadInMotion (lAxisNo, &uStatus);
```

● Verification of Drive State Information

Drive state of specified axis can be read by verification of DriveStatus register on the specified axis.

[AxmStatusReadInMotion](#) API is used to verify drive state of specified axis

```
// Read drive state on axis 0.
long lAxisNo = 0;
DWORD uStatus;
AxmStatusReadMotion(lAxisNo, &uStatus);
```

Since [AxmStatusReadMotion](#) API can verify more detailed information of each CAMC-IP chip and CAMC-QI chip in addition to the information that is mentioned here, refer to the corresponding manual. Especially most used values are showed below.

* Mostly used SMC-2V03 Drive Status

Define	Description
IPDRIVE_STATUS_BUSY	BUSY (in DRIVE)
IPDRIVE_STATUS_DOWN	DOWN (in deceleration DRIVE)
IPDRIVE_STATUS_CONST	CONST (in constant velocity DRIVE)
IPDRIVE_STATUS_UP	Up (in acceleration DRIVE)
IPDRIVE_STATUS_PRESET_DRIVING	In position drive
IPDRIVE_STATUS_CONTINUOUS_DRIVING	In velocity drive
IPDRIVE_STATUS_SENSOR_DRIVING	In sensor position drive

* Mostly used PCI-N804/404 Drive Status

Define	Description
QIDRIVE_STATUS_0	BUSY (in DRIVE)
QIDRIVE_STATUS_1	DOWN (in deceleration DRIVE)
QIDRIVE_STATUS_2	CONST (in constant velocity DRIVE)
QIDRIVE_STATUS_3	Up (in acceleration DRIVE)
QIDRIVE_STATUS_4	In continuous drive
QIDRIVE_STATUS_5	In specified distance drive
QIDRIVE_STATUS_6	In MPG drive

● Verification of Stop State Information

Exit state of specified axis can be verified by verification of EndStatus register on an assigned axis. To verify the exit state of specified axis, [AxmStatusReadStop](#) API is used.

```
//Verify exit state on axis 0.
long lAxisNo = 0;
```

DWORD uStatus;
[AxmStatusReadStop](#) (lAxisNo, &uStatus);

Showing on the library manual, since [AxmStatusReadStop](#) API returns various stop state, it can be verified why it is stopped currently. If it returns 0, it means drive is exited normally, and if returns other than 0, it is exited by corresponding exit cause. Most used values are following.

* Mostly used SMC-2V03 End Status

Define	Description
IPEND_STATUS_ELM	Exit by limit signal input
IPEND_STATUS_ESTOP_SIGNAL	Exit by emergency stop signal input
IPEND_STATUS_SSTOP_COMMAND	Exit by deceleration stop command
IPEND_STATUS_ESTOP_COMMAND	Exit by emergency stop command
IPEND_STATUS_ALARM_SIGNAL	Exit by alarm signal input
IPEND_STATUS_DATA_ERROR	Exit by data setting error
IPEND_STATUS_ORIGIN_DETECT	Exit by home search
IPEND_STATUS_SIGNAL_DETECT	Exit by signal search

* Mostly used PCI-N804/404 End Status

Define	Description
QIEND_STATUS_0	Exit by forward direction limit signal (PELM)
QIEND_STATUS_1	Exit by reverse direction limit signal (NELM)
QIEND_STATUS_4	Drive exit by forward direction soft limit emergency stop function
QIEND_STATUS_5	Drive exit by reverse direction soft limit emergency stop function
QIEND_STATUS_6	Drive exit by forward direction soft limit deceleration stop function
QIEND_STATUS_7	Drive exit by reverse direction soft limit deceleration stop function
QIEND_STATUS_8	Drive exit by servo alarm function
QIEND_STATUS_9	Drive exit by emergency stop signal input
QIEND_STATUS_10	Drive exit by emergency stop command
QIEND_STATUS_11	Drive exit by deceleration stop command

```

// Ex9_AXM_MotionStatus1.cpp : Defines the entry point for the console application.
// Verify moving distance and velocity moving to (+) direction as much as 200 using relative position
drive API

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

#define AXIS_0 0

void main(void)
{
    // Initialize library.
    // 7 means IRQ. IRQ is set automatically in PCI.
    DWORD Code = AxlOpen(7);
    if (Code == AXT_RT_SUCCESS)
    {
        printf("Library is initialized.\n");

        //Inspect whether motion module exists or not
        DWORD dwStatus;
        Code = AxmInfoIsMotionModule (&dwStatus);
        if(Code == AXT_RT_SUCCESS)
        {
            if(dwStatus == STATUS_EXIST)
            {
                printf("Motion module exists.\n");

                // Set Active level of +End limit and -End limit on axis 0 to HIGH.
                AxmSignalSetLimit(AXIS_0, 0, HIGH, HIGH);
                // Set input level of inposition signal on axis 0 to HIGH.
                AxmSignalSetInpos (AXIS_0, HIGH);
                // Set input level of alarm signal on axis 0 to LOW.
                AxmSignalSetServoAlarm (AXIS_0, LOW);
                // Set Active input level of ESTOP signal on axis 0 to HIGH.
                AxmSignalSetStop (AXIS_0, 0, HIGH);

                //Make ordered value on axis 0 to be mm unit.
                AxmMotSetMoveUnitPerPulse(AXIS_0, 10, 10000);
                //Set initial velocity on axis 0 to 1. Default : 1
                AxmMotSetMinVel(AXIS_0, 1);
                //Make pulse output method on axis 0 to TwoCwCcwHigh.
                AxmMotSetPulseOutMethod (AXIS_0, TwoCcwCcwhigh);
                //Set encoder input method on designated axis to 4 multiplication.
                AxmMotSetEnclInputMethod (AXIS_0, ObverseSqr4Mode);

                AxmMotSetAbsRelMode (AXIS_0, 0);           //Absolute position drive
                AxmMotSetProfileMode (AXIS_0, 3);         //Symmetrical S-curve drive

                // Set home search method
                AxmHomeSetMethod (AXIS_0,-1,HomeSensor,1,2000.0,0.0);
            }
        }
    }
}

```

```

// Set velocity that will be used during home search
AxmHomeSetVel(AXIS_0, 100, 20, 10, 10, 200, 40);

//Servo On
AxmSignalServoOn(AXIS_0, ENABLE);

//Search home.
printf("Start home search pressing any key.\n");
getch();
AxmHomeSetStart (AXIS_0);

//Verify current home search progress rate
DWORD uHomeMainStepNumber , uHomeStepNumber;
DWORD uHomeResult;

AxmHomeGetResult (AXIS_0, &uHomeResult);
while(!uHomeResult)
{
    AxmHomeGetResult (AXIS_0, &uHomeResult);
    AxmHomeGetRate(AXIS_0, &uHomeMainStepNumber ,
        &uHomeStepNumber);
    printf("\r In home search . . . %d %", uHomeStepNumber);
}

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("Move to (+) direction as much as 200 with S-curve velocity profile on axis 0
    (velocity:100, acceleration:200, exit API at start point)\n");
printf("Drive starts pressing any key.");
getch();
AxmMoveStartPos (AXIS_0, 200, 100, 200, 200);

// Verify the drive state during in drive.
DWORD uStatus;
double dActualPosition, dCommandPosition, dError, dVelocity;
AxmStatusReadInMotion ( AXIS_0, &uStatus);
while(uStatus)
{
    AxmStatusReadInMotion ( AXIS_0, &uStatus);
    AxmStatusGetActPos (AXIS_0, & dActualPosition);
        //Verify current position on axis 0.
    AxmStatusGetCmdPos (AXIS_0, & dCommandPosition);
        //Verify ordered position on axis 0.
    AxmStatusReadPosError (AXIS_0, & dError);
        //Verify difference between ordered position and actual position on axis 0.
    AxmStatusReadVel (AXIS_0, & dVelocity); //Verify current velocity on axis 0.

    printf("\rAP:%.1f, CP:%.1f, Error:%.1f, CV:%.1f ,"

```

```
    dActualPosition, dCommandPosition, dError, dVelocity );  
  
} //End of While()  
  
printf("WnDrive is completed.Wn");  
AxmlSignalServoOn(AXIS_0, DISABLE); //Servo Off  
}  
else  
    printf ("AxmlInfoIsMotionModule() : ERROR ( NOT STATUS_EXIST )  
           code 0x%xWn",Code);  
}  
else  
    printf ("AxmlInfoIsMotionModule() : ERROR ( Return FALSE ) code 0x%xWn",Code);  
  
}  
else  
    printf ("AxlOpen() : ERROR code 0x%xWn",Code);  
  
// Exit library.  
if (AxlClose())  
    printf("Library is exited.Wn");  
else  
    printf("Library is not exited normally.Wn");  
}
```

```

// Ex10_AXM_MotionStatus2.cpp : Defines the entry point for the console application.
// Verify drive and stop state moving to (+) direction as much as 200 using absolute position drive
API, after home search

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

#define AXIS_0 0

void main(void)
{
    // Initialize library.
    // 7 means IRQ. IRQ is set automatically in PCI.
    DWORD Code = AxlOpen(7);
    if (Code == AXT_RT_SUCCESS)
    {
        printf("Library is initialized.\n");

        //Inspect whether motion module exists or not
        DWORD dwStatus;
        Code = AxmInfoIsMotionModule (&dwStatus);
        if(Code == AXT_RT_SUCCESS)
        {
            if(dwStatus == STATUS_EXIST)
            {
                printf("Motion module exists.\n");

                // Set Active level of +End limit and -End limit on axis 0 to HIGH.
                AxmSignalSetLimit(AXIS_0, 0, HIGH, HIGH);
                // Set input level of inposition signal on axis 0 to HIGH.
                AxmSignalSetInpos (AXIS_0, HIGH);
                // Set input level of alarm signal on axis 0 to LOW.
                AxmSignalSetServoAlarm (AXIS_0, LOW);
                // Set Active input level of ESTOP signal on axis 0 to HIGH.
                AxmSignalSetStop (AXIS_0, 0, HIGH);

                //Make ordered value on axis 0 to be mm unit.
                AxmMotSetMoveUnitPerPulse(AXIS_0, 10, 10000);
                //Set initial velocity on axis 0 to 1. Default : 1
                AxmMotSetMinVel(AXIS_0, 1);
                //Make pulse output method on axis 0 to TwoCwCcwHigh.
                AxmMotSetPulseOutMethod (AXIS_0, TwoCcwCcwhigh);
                //Set encoder input method on designated axis to 4 multiplication.

                AxmMotSetEnclInputMethod (AXIS_0, ObverseSqr4Mode);

                // Set home search method
                AxmHomeSetMethod (AXIS_0,-1,HomeSensor,1,2000.0,0.0);
                // Set velocity that will be used during home search
                AxmHomeSetVel(AXIS_0, 100, 20, 10, 10, 200, 40);

```

```

//Servo On
AxmSignalServoOn(AXIS_0, ENABLE);

//Search home.
printf("Start home search pressing any key.\n");
getch();
AxmHomeSetStart (AXIS_0);

//Verify current home search progress rate
DWORD uHomeMainStepNumber , uHomeStepNumber;
AxmHomeGetRate(AXIS_0, &uHomeMainStepNumber , &uHomeStepNumber);
while(uHomeStepNumber < 100)
{
    AxmHomeGetRate(AXIS_0, &uHomeMainStepNumber ,
                      &uHomeStepNumber);
    printf("\r In home search . . . %d %", uHomeStepNumber);
}

// Move to (+) direction as much as 200 with S-curve velocity profile on axis 0
// (velocity:50, acceleration:100, exit API at start point )
printf("\nDrive starts pressing any key.\n");
getch();
AxmMotSetAbsRelMode (AXIS_0, 0); //Absolute position drive
AxmMotSetProfileMode (AXIS_0, 3); //Symmetrical S-curve drive
AxmMoveStartPos (AXIS_0, 200, 50, 100, 100);

DWORD uStatus;
AxmStatusReadInMotion ( AXIS_0, &uStatus);
while ( uStatus )
{
    AxmStatusReadInMotion ( AXIS_0, &uStatus);

    DWORD dDriveStatus;
    // Read drive state on axis 0.
    AxmStatusReadMotion(AXIS_0, &dDriveStatus);
    //Verify information of corresponding bit that is intended to verify.
    if(dDriveStatus & IPDRIVE_STATUS_DOWN) printf("Current state :
        in deceleration.");
    if(dDriveStatus & IPDRIVE_STATUS_CONST) printf("Current state :
        in constant velocity.");
    if(dDriveStatus & IPDRIVE_STATUS_UP)     printf("Current state :
        in acceleration.");

} //End of While()
printf("\nDrive is completed.\n");

//Verify drive exit state.
DWORD dEndStatus;
AxmStatusReadStop (AXIS_0, &dEndStatus);
switch(dEndStatus) {
    case 0 :           printf("Normal end \n");           break;
    case IPEND_STATUS_SLM :
        printf("Exit by Slow Limit signal input \n"); break;
}

```

```

        case IPEND_STATUS_ELM :
            printf("Exit by Limit signal input \n"); break;
        case IPEND_STATUS_SSTOP_SIGNAL :
            printf("Exit by deceleration stop signal input \n"); break;
        case IPEND_STATUS_ESTOP_SIGNAL :
            printf("Exit by emergency stop signal input \n"); break;
        case IPEND_STATUS_SSTOP_COMMAND :
            printf("Exit by deceleration stop command \n"); break;
        case IPEND_STATUS_ESTOP_COMMAND :
            printf("Exit by emergency stop command \n"); break;
        case IPEND_STATUS_ALARM_SIGNAL :
            printf("Exit by alarm signal input \n"); break;
        case IPEND_STATUS_DATA_ERROR :
            printf("Exit by data setting error \n"); break;
        case IPEND_STATUS_ORIGIN_DETECT :
            printf("Exit by home search \n"); break;
        case IPEND_STATUS_SIGNAL_DETECT :
            printf("Exit by signal search \n"); break;
        case IPEND_STATUS_PRESET_PULSE_DRIVE :
            printf("Exit by preset pulse drive \n"); break;
        case IPEND_STATUS_SENSOR_PULSE_DRIVE :
            printf("Exit by sensor pulse drive \n"); break;
        case IPEND_STATUS_LIMIT :
            printf("Exit by Limit complete stop \n"); break;
        case IPEND_STATUS_SOFTLIMIT:
            printf("Exit by Soft limit \n"); break;
        case IPEND_STATUS_INTERPOLATION_DRIVE :
            printf("Exit interpolation drive \n"); break;
        default:   printf("Exit by unknown cause(%x) \n",dEndStatus);
    }

    AxmSignalServoOn(AXIS_0, DISABLE); //Servo Off
}
else
printf ("AxmInfoMotionModule() : ERROR ( NOT STATUS_EXIST )
        code 0x%x\n",Code);
}
else
printf ("AxmInfoMotionModule() : ERROR ( Return FALSE )
        code 0x%x\n",Code);
}
else
printf ("AxIOpen() : ERROR     code 0x%x\n",Code);

// Exit library.
if (AxIClose())
    printf("Library is exited.\n");
else
    printf("Library is not exited normally.\n");
}

```

Multi-axis Drive

Generally, drive system moves several axes to a specified position at once in equipment, but there are many cases that each axis has related motion with a constant pattern. Multi-axis API is provided for these drives.

Multi-axis drive setting is to set one-axis drive setting for each axis that has been previously. For example, if there are 2 axes, set one-axis drive API setting to axis 0 and 1 axis each.

Multi-axis drive API is divided into multi-axis position drive that moves several axes at once with set position and velocity, and continuous interpolation drive that makes to drive several axes to a specified position along with specified path.

- [1\) Multi-axis Position Drive](#)
- [2\) Interpolation Drive](#)
- [3\) Continuous Interpolation Drive](#)

1) Multi-axis Position Drive

Multi-axis position drive is what position drive of one-axis drive is carried out on two axes at the same time. Axes are started at once to the exit position of each axis that is inputted from current position, and moving velocity and acceleration of two axes are available to each setting. This multi-axis position drive has [AxmMoveMultiPos](#) API which is exit API at end point and [AxmMoveStartMultiPos](#) API which is exit API at start point.

Two API has only difference between exit API at start point and exit API at end point, and input factor is same. Here, axis number must be inputted with an order of ascending series. Following example is to move 0 and 1 axis to the position of 100 and 50 each with same velocity and acceleration. Position drive API is not an interpolation drive, and moves individually by the position, velocity, acceleration value that is set on each axis. There is no difference except starting at the same time. Obviously, 1 axis reaches the target point earlier than axis 0, and axis 0 still moves in the state that 1 axis is stopped. [AxmMoveMultiPos](#) API which is an exit API at end point comes out of API after stop of all axes that command is ordered, and carries out the next command on program.

```
//With axis 0 as a relative coordination, move to (100,50) position with symmetrical S-curve drive
mode (velocity 100, acceleration 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);
```

If [AxmMoveStartMultiPos](#) API which is an exit API at start point is needed to use, follow below. The next API is carried out immediately after starting drive because of exit API at the start point, and if it is needed to

verify whether exiting drive, use [AxmStatusReadInMotion](#) API for the each axis.

```
// With axis 0 as a relative coordination, move to (100,50) position with symmetrical S-curve drive
mode (velocity 100, acceleration 200)
DWORD uAbsRelMode = 0;
DWORD uProfileMode = 3;
AxmMotSetAbsRelMode (IAxisNo, uAbsRelMode);
AxmMotSetProfileMode (IAxisNo, 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);

//Wait until all motions are exited.
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.
// Return to home again moving to (100,50) position after home setting of 2 axes mechanical part
#include "stdafx.h"
#include "AXL.h"
#include <conio.h>
#include "stdio.h"

#define AXIS_0 0
#define AXIS_1 1

void main(void)
{
    // Initialize library..
    // 7 means IRQ. IRQ is set automatically in PCI.
    DWORD Code = AxlOpen(7);
    if (Code == AXT_RT_SUCCESS)
    {
        printf("Library is initialized.\n");
        // Inspect whether motion module exists or not
        DWORD dwStatus;
```

```

Code = AxmlInfoIsMotionModule (&dwStatus);
if(Code == AXT_RT_SUCCESS)
{
    if(dwStatus == STATUS_EXIST)
    {
        printf("Motion module exists.\n");

        for(int nAxisNo = 0; nAxisNo<2; nAxisNo++)
        {
            // Set Active level of +End limit and -End limit on axis 0 to HIGH.
            AxmSignalSetLimit(nAxisNo, 0, HIGH, HIGH);
            // Set input level of inposition signal on axis 0 to HIGH.
            AxmSignalSetInpos (nAxisNo, HIGH);
            // Set input level of alarm signal on axis 0 to LOW.
            AxmSignalSetServoAlarm (nAxisNo, LOW);
            // Set Active input level of ESTOP signal on axis 0 to HIGH.
            AxmSignalSetStop (nAxisNo, 0, HIGH);

            // Make ordered value on axis 0 to be mm unit.
            AxmMotSetMoveUnitPerPulse(nAxisNo, 10, 10000);
            // Set initial velocity on axis 0 to 1. Default : 1
            AxmMotSetMinVel(nAxisNo, 1);
            // Make pulse output method on axis 0 to TwoCwCcwHigh.
            AxmMotSetPulseOutMethod (nAxisNo, TwoCwCcwHigh);
            // Set encoder input method on designated axis to 4 multiplication.
            AxmMotSetEnclInputMethod (nAxisNo, ObverseSqr4Mode);

            // Set home search method
            AxmHomeSetMethod (nAxisNo,-1,HomeSensor,1,2000.0,0.0);
            // Set velocity that will be used during home search
            AxmHomeSetVel(nAxisNo, 100, 20, 10, 10, 200, 40);

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

        // Search home
        printf("Start home search pressing any key.\n");
        getch();
        AxmHomeSetStart (AXIS_0);
        AxmHomeSetStart (AXIS_1);

        // Verify current home search progress rate
        DWORD uHomeResult_0, uHomeResult_1;
        DWORD uHomeMainStepNumber , uHomeStepNumber_0,uHomeStepNumber_1;
        AxmHomeGetResult (AXIS_0, &uHomeResult_0);
        AxmHomeGetResult (AXIS_1, &uHomeResult_1);
        // In home search : 0, Home search completion : 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("Wr In home search... axis 0 : %d%, 1 axis : %d ", uHomeStepNumber_0,
      uHomeStepNumber_1);
}

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

printf("Wn Move as much as (+)100 of axis 0 and (+)50 of 1 axis with trapezoid velocity
       profile (exit API at end point)Wn");
printf("Drive starts pressing any key.Wn");
getch();
for(nAxisNo = 0; nAxisNo<1; nAxisNo++)
{
    AxmMotSetAbsRelMode (nAxisNo, 1); //Set to relative coordination
    AxmMotSetProfileMode (nAxisNo, 0); //Set to symmetrical trapezoid velocity profile
}
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("Drive is completed.Wn");

printf("Move to home with S-curve velocity profile (exit API at start point)Wn");
printf("Drive starts pressing any key.");
getch();
for(nAxisNo = 0; nAxisNo<1; nAxisNo++)
{
    AxmMotSetAbsRelMode (nAxisNo, 0); // Set to absolute coordination
    AxmMotSetProfileMode (nAxisNo, 3); // Set to symmetrical S-curve velocity profile
}
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);

//Wait until all motions are exited.
BOOL WaitForDone = TRUE;
DWORD uStatus0, uStatus1;
while(WaitForDone) {
    AxmStatusReadInMotion (AXIS_0, &uStatus0);
    AxmStatusReadInMotion (AXIS_1, &uStatus1);
    if(uStatus0 == 0 && uStatus1 == 0)
        WaitForDone = FALSE;
}
printf("Drive is completed.Wn");

```

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

}

else
printf ("AxmInfoMotionModule () : ERROR ( NOT STATUS_EXIST )
code 0x%x\n",Code);
}

else
printf ("AxmInfoMotionModule () : ERROR ( Return FALSE ) code 0x%x\n",Code);

}

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

/ Exit library.
if (AxlClose())
printf("Library is exited.\n");
else
printf("Library is not exited normally.\n");
}
```

2) Interpolation Drive

Multi-axis position drive that is mentioned before is a multi-axis drive that two axes carry out each motion starting at the same time. However, it occurs that two axes move along with straight line or circular or assigned path with related movement mutually, and for those drive, straight line interpolation, circular interpolation, and continuous interpolation drive for each are provided. Since the setting of velocity profile and absolute/relative coordination drive for the interpolation drive is to be set along with an axis that has the earliest axis number among axes that had interpolation drive, please be careful to set.

● Straight Line Interpolation Drive

Straight line interpolation drive makes two axes included in one module to drive of straight interpolation using [AxmLineMove](#) API. In case of SMC-2V03 chip, only 2 axes interpolation is available because there are 2 axes in one chip, but PCI-N804/404 has 4 axes in one chip and arbitrary straight line interpolation of 2, 3, and 4 axis in 4 axes is available. This API drives properly to drive two axes with specified velocity and acceleration from the current position to the inputted exit point.

```
//Interpolation drive with S-curve to relative position of (100,50)
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;

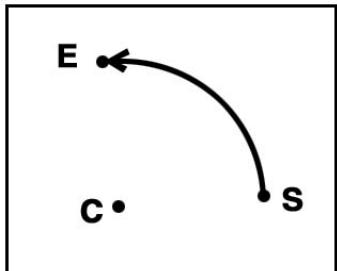
AxmContiWriteClear(lCoordinate);
AxmContiSetAxisMap(lCoordinate, lSize, lAxesNo);
AxmContiSetAbsRelMode (lCoordinate, uAbsRelMode); // Set to relative position drive

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

● Circular Interpolation Drive

Circular interpolation drive is to be interpolation drive with circular path based on the corresponding setting from the current position to the assigned position. And this circular interpolation drive APIs provide [AxmCircleCenterMove](#), [AxmCircleRadiusMove](#), [AxmCircleAngleMove](#), and [AxmCirclePointMove](#) API.

- [AxmCircleCenterMove](#)



[AxmCircleCenterMove](#) API carries out circular interpolation drive from the current position (S) with input of exit point (E) and center point (C). Since this is a method to calculate circular arc using the position of three points, circle is composed of that accurate center point and exit point are inputted, and API is operated. If a wrong coordination is inputted, API will not be operated and returns FALSE. If circular interpolation drive is needed to carry out counterclockwise with center of (0, 100) from the current position (0,0) to (100,100) position, do the following.

```
// Carry out circular interpolation with center of (0, 100) from the current position (0,0) to (100,100)
position.
long lSize = 2;
long lAxesNo = {0,1};
double dCenPos[2];
double dEndPos[2];
long lCoordinate = 0;
DWORD uAbsRelMode = 1;
AxmContiWriteClear(lCoordinate);
AxmContiSetAxisMap(lCoordinate, lSize, lAxesNo);
AxmContiSetAbsRelMode (lCoordinate, uAbsRelMode); // Set to relative position drive

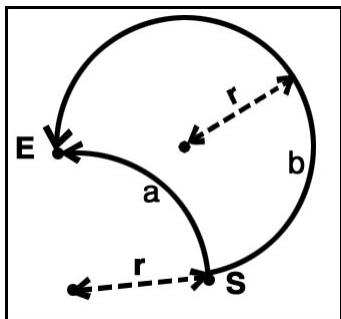
double dCenterXPosition = 0;
double dCenterYPosition = 100;
double dEndXPosition = 100;
double dEndYPosition = 100;

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

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

double dMaxVelocity = 100;
double dMaxAccel = 200;
double dMaxDecel = 200;
DWORD uCWDdir = 1; // [0] Ccw direction, [1] Cw direction
AxmCircleCenterMove(lCoordinate, lAxesNo, dCenPos, dEndPos, dMaxVelocity, dMaxAccel,
dMaxDecel, uCWDdir);
```

● [AxmCircleRadiusMove](#)



[AxmCircleRadiusMove](#) API carries out circular interpolation drive from the current position (S) with input of exit point (E) and radius (r). Since this is a method to calculate arc that passes two points and has constant radius, two circles are to be structured. Therefore, if uShortDistance value is 0 for the input radius, circle with short distance (a) is selected, and if 1, circle with long distance (b) is selected.

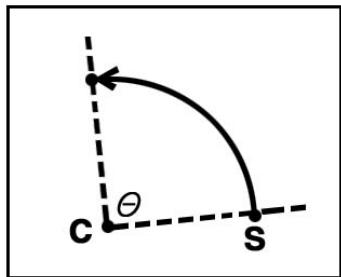
If circular interpolation is needed to carry out clockwise with long distance that has a radius of 120 from the current position to (300,200) position, do the following.

```
// Carry out circular interpolation from the current position to (300,200) position with long distance
// that has a radius of 120
long lSize = 2;
long lAxesNo = {0,1};
double dEndPos[2];
long lCoordinate = 0;

DWORD uAbsRelMode = 1;
DWORD uProfileMode = 3;
AxmContiWriteClear(lCoordinate);
AxmContiSetAxisMap(lCoordinate, lSize, lAxesNo);
AxmContiSetAbsRelMode (lCoordinate, uAbsRelMode); // Set to relative position drive

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 uCWDdir = 0; // [0] Cw direction, [1] Ccw direction
DWORD uShortDistance = 1; // [0]: Small (circular) distance , [1] Big (circular) distance
AxmCircleRadiusMove(lCoordinate, lAxesNo, dRadius, dEndPos, dMaxVelocity, dMaxAccel,
dMaxDecel, uCWDdir, uShortDistance);
```

● [AxmCircleAngleMove](#)



[AxmCircleAngleMove](#) API carries out circular interpolation drive from the current position (S) with input of center point (C) and rotational angle (θ). This is a method to calculate circular arc using radius and angle of arc, and a circle is structured.

Circular interpolation is carried out as much as 200 degrees with center of (200,200) from the current position, and if interpolation drive is needed to carry out clockwise, do the following.

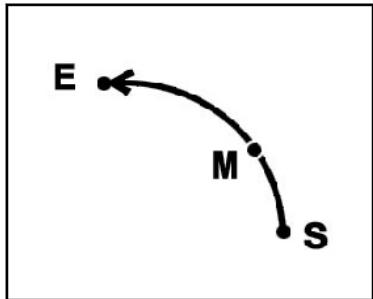
```
// Carried out circular interpolation as much as 200 degrees with center of (200,200) from the
// current position.
long lSize = 2;
long lAxesNo = {0,1};
double dCenPos[2];
long lCoordinate = 0;

DWORD uAbsRelMode = 1;
DWORD uProfileMode = 3;
AxmContiWriteClear(lCoordinate);
AxmContiSetAxisMap(lCoordinate, lSize, lAxesNo);
AxmContiSetAbsRelMode (lCoordinate, uAbsRelMode); // Set to relative position drive

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 uCWDDir = 0; // [0] Cw direction, [1] Ccw direction
AxmCircleAngleMove(lCoordinate, lAxesNo, dCenPos, dAngle, dMaxVelocity, dMaxAccel,
dMaxDecel, uCWDDir);
```

● [AxmCirclePointMove](#)



[AxmCirclePointMove](#) API carries out circular interpolation drive from the current position (S) with input of middle point (M) and exit point (E). This is a method to calculate circular arc that passes three points, and a circle is structured.

If interpolation drive is needed to carry out from the current position to (150,150) passing (100,150), do the following.

```
// Carry out circular interpolation from the current position to (150,150) passing (100,150).
long lSize = 2;
long lAxesNo = {0,1};
double dMidPos[2];
double dEndPos[2];
long lCoordinate = 0;
DWORD uAbsRelMode = 1;
DWORD uProfileMode = 3;
AxmContiWriteClear(lCoordinate);
AxmContiSetAxisMap(lCoordinate, lSize, lAxesNo);
AxmContiSetAbsRelMode (lCoordinate, uAbsRelMode); // Set to relative position drive

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.
// Start interpolation drive after setting home of 2 axes mechanical part.
#include "stdafx.h"
#include "AXL.h"
#include <conio.h>
#include "stdio.h"

#define AXIS_0 0
#define AXIS_1 1

void main(void)
{
    // Initialize library..
    // 7 means IRQ. IRQ is set automatically in PCI.
    DWORD Code = AxlOpen(7);
    if (Code == AXT_RT_SUCCESS)
    {
        printf("Library is initialized.\n");

        // Inspect whether motion module exists or not
        DWORD dwStatus;
        Code = AxmlInfoIsMotionModule (&dwStatus);
        if(Code == AXT_RT_SUCCESS)
        {
            if(dwStatus == STATUS_EXIST)
            {
                printf("Motion module exists.\n");

                for(int nAxisNo = 0; nAxisNo<2; nAxisNo++)
                {
                    // Set Active level of +End limit and -End limit on axis 0 to HIGH.
                    AxmSignalSetLimit(nAxisNo, 0, HIGH, HIGH);
                    // Set input level of inposition signal on axis 0 to HIGH.
                    AxmSignalSetInpos (nAxisNo, HIGH);
                    // Set input level of alarm signal on axis 0 to LOW.
                    AxmSignalSetServoAlarm (nAxisNo, LOW);
                    // Set Active input level of ESTOP signal on axis 0 to HIGH.
                    AxmSignalSetStop (nAxisNo, 0, HIGH);

                    // Make ordered value on axis 0 to be mm unit.
                    AxmMotSetMoveUnitPerPulse(nAxisNo, 10, 10000);
                    // Set initial velocity on axis 0 to 1. Default : 1
                    AxmMotSetMinVel(nAxisNo, 1);
                    // Make pulse output method on axis 0 to TwoCwCcwHigh.
                    AxmMotSetPulseOutMethod (nAxisNo, TwoCwCcwHigh);
                    // Set encoder input method on designated axis to 4 multiplication.
                    AxmMotSetEnclInputMethod (nAxisNo, ObverseSqr4Mode);

                    // Set home search method
                    AxmHomeSetMethod (nAxisNo,-1,HomeSensor,1,2000.0,0.0);
                    // Set velocity that will be used during home search
                    AxmHomeSetVel(nAxisNo, 100, 20, 10, 200, 40);
                }
            }
        }
    }
}

```

```

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

// Search home
printf("Start home search pressing any key.\n");
getch();
AxmHomeSetStart (AXIS_0);
AxmHomeSetStart (AXIS_1);

// Verify current home search progress rate
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("\r In home search... axis 0 : %d%, 1 axis : %d ", uHomeStepNumber_0,
          uHomeStepNumber_1);
}

printf("\nCarry out straight line interpolation drive to relative position of (150,0).\n");
printf("Drive starts pressing any key.\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];

AxmContiWriteClear(lCoordinate);
AxmContiSetAxisMap(lCoordinate, lSize, lpAxesNo);
AxmContiSetAbsRelMode (lCoordinate, 1); // Set to relative position drive
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("WnCarry out circular interpolation drive to relative position of (50,50) with center of
      (0,50) and with Ccw direction.Wn");
printf("Drive starts pressing any key.Wn");
getch();

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

//Wait until all motions are exited.
WaitForDone = TRUE;
while(WaitForDone) {
    AxmStatusReadInMotion (AXIS_0, &uStatus0);
    AxmStatusReadInMotion (AXIS_1, &uStatus1);
    if(uStatus0 == 0 && uStatus1 == 0)
        WaitForDone = FALSE;
}

printf("WnCarry out straight line interpolation drive to relative position of (0,100).Wn");
printf("Drive starts pressing any key.Wn");
getch();
dpPosition[0] = 0; dpPosition[1] = 100;
AxmLineMove (lCoordinate, dpPosition, dMaxVelocity, dMaxAccel, dMaxDecel);
//Wait until all motions are exited.
WaitForDone = TRUE;
while(WaitForDone) {
    AxmStatusReadInMotion (AXIS_0, &uStatus0);
    AxmStatusReadInMotion (AXIS_1, &uStatus1);
    if(uStatus0 == 0 && uStatus1 == 0)
        WaitForDone = FALSE;
}

printf("Wn Carry out circular interpolation drive to relative position of (-50,50)
      having small distance with radius of 50 and with Ccw direction.Wn");
printf("Drive starts pressing any key.Wn");
getch();

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

//Wait until all motions are exited.
WaitForDone = TRUE;
while(WaitForDone) {
    AxmStatusReadInMotion (AXIS_0, &uStatus0);
    AxmStatusReadInMotion (AXIS_1, &uStatus1);
    if(uStatus0 == 0 && uStatus1 == 0)
        WaitForDone = FALSE;
}

printf("Wn Carry out circular interpolation drive as much as 90 degrees with Ccw

```

```

        direction and with center of relative position of (0,-150). \n);
printf("Drive starts pressing any key.\n");
getch();

//Wait until all motions are exited.
dCenPos[0] = 0; dCenPos[1] = -150;
AxmCircleAngleMove(lCoordinate, lpAxesNo, dCenPos, 90, dMaxVelocity, dMaxAccel,
dMaxDecel, 1);

//Wait until all motions are exited.
WaitForDone = TRUE;
while(WaitForDone) {
    AxmStatusReadInMotion (AXIS_0, &uStatus0);
    AxmStatusReadInMotion (AXIS_1, &uStatus1);
    if(uStatus0 == 0 && uStatus1 == 0)
        WaitForDone = FALSE;
}

printf("\n Carry out circular interpolation drive as much as (100,0) with Ccw direction
        having middle point of relative position of (50,50).\n");
printf("Drive starts pressing any key.\n");
getch();

//Wait until all motions are exited.
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("\nCarry out straight line interpolation drive to home.\n");
printf("Drive starts pressing any key.\n");
getch();
dpPosition[0] = 0; dpPosition[1] = -50;
AxmLineMove (lCoordinate, dpPosition, dMaxVelocity, dMaxAccel, dMaxDecel);
//Wait until all motions are exited.
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);

```

```

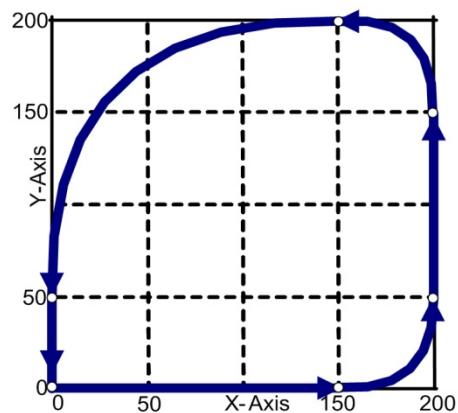
    }
    else
        printf ("AxmInfoIsMotionModule () : ERROR ( NOT STATUS_EXIST )
                code 0x%x\n",Code);
    }
    else
        printf ("AxmInfoIsMotionModule () : ERROR ( Return FALSE )   code 0x%x\n",Code);

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

// Exit library.
if (AxIClose())
    printf("Library is exited.\n");
else
    printf("Library is not exited normally.\n");
}

```

* If an example of straight line interpolation drive above is carried out, following result can be verified.



3) Continuous Interpolation Drive

● *Straight Line Continuous Interpolation Drive*

Continuous interpolation drive API is what is to drive interpolation drive mentioned before continuously, and has a drive that accelerates after starting drive, passes assigned positions with constant velocity, and decelerates and stops.

One unique thing is that, after axes to use for continuous interpolation takes mapping procedure to the new coordination system, stores commands to carry out interpolation drive continuously in the memory area of “continuous interpolation queue,” if continuous interpolation start command is given later, value goes into motion control chip after coming out one by one from the stored continuous interpolation data, and carries out interpolation drive continuously. Of course, interpolation data can be inputted after interpolation drive is started.

In order to carry out continuous interpolation drive, first, axes to be used for continuous interpolation must take a process of mapping. Interpolation drive APIs that are used for continuous interpolation are to be a combination of 2 axes interpolation drives, but they have to take mapping to one coordination system that includes all axes to be used after mapping of axes. For example, if it is carried out interpolation drive of 1 and 2 axis after interpolation drive of 0 and 1 axis continuously using three axes of 0, 1, and 2, they have to take mapping to one coordination system. After mapping, set velocity profile to use in corresponding coordination system, and set absolute/relative position drive.

```
// Carry out mapping of 0,1,2 axis to coordination system of 0 number that is to use for continuous
// interpolation drive.
long lCoordinate = 0;
long lSize = 3;
long lRealAxesNo = {0,1,2};
AxmContiSetAxisMap(lCoordinate, lSize, lRealAxesNo);

//Set 0 number coordination system to relative position drive.
DWORD uAbsRelMode = 1; // [0] : Absolute position drive, [1] : relative position drive
AxmContiSetAbsRelMode(lCoordinate, uAbsRelMode);
```

To input straight continuous interpolation data on continuous interpolation drive queue, if [AxmContiBeginNode](#) and [AxmContiEndNode](#) API are called, it is to be [AxmLineMove](#) API. And to input circular continuous interpolation data, if [AxmCircleCenterMove](#), [AxmCirclePointMove](#), [AxmCircleRadiusMove](#), and [AxmCircleAngleMove](#) API are used, it is to be an API that stores in internal queue.

```
//Input data on interpolation queue to carry out line interpolation drive to (100,150,0). (velocity : 100,
acceleration : 200)
long lCoordinate = 0;
double dPosition = {100,150,0};
double dMaxVelocity = 100;
double dMaxAccel = 200;
double dMaxDecel = 200;
AxmContiBeginNode (lCoordinate);
AxmLineMove (lCoordinate, dPosition, dMaxVelocity, dMaxAccel, dMaxDecel);
AxmContiEndNode (lCoordinate);
```

```
// Input data on interpolation queue to carry out circular interpolation drive with center of (0,100)
from the current position (0,0) to (100,100) using 0 and 1 axis
long lSize = 2;
long lAxesNo = {0,1};
double dCenPos[2];
double dEndPos[2];
long lCoordinate = 0;
double dCenterXPosition = 0;
double dCenterYPosition = 100;
double dEndXPosition = 100;
double dEndYPosition = 100;
double dMaxVelocity = 100;
double dMaxAccel = 200;
double dMaxDecel = 200;

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

dEndPos[0] = dEndXPosition;
dEndPos[1] = dEndYPosition;
DWORD uCWDdir = 1;           // [0] Ccw direction, [1] Cw direction
AxmContiWriteClear(lCoordinate);
AxmContiSetAxisMap(lCoordinate, lSize, lAxesNo);
AxmContiBeginNode (lCoordinate);
AxmCircleCenterMove(lCoordinate, lAxesNo, dCenPos, dEndPos,, dMaxVelocity, dMaxAccel,
dMaxDecel, uCWDdir);
AxmContiEndNode (lCoordinate);
```

```
// Input data on interpolation queue to carry out circular interpolation with radius of 120 from the
current position to (300,200) using 1 and 2 axis
long lSize = 2;
long lAxesNo = {1,2};
double dEndPos[2];
long lCoordinate = 0;
double dRadius = 120;
double dEndXPosition = 300
double dEndYPosition = 200;
double dMaxVelocity = 100;
double dMaxAccel = 200;
double dMaxDecel = 200;
DWORD uCWDdir = 0;           // [0] Ccw direction, [1] Cw direction
DWORD uShortDistance = 1;     // [0]: short (circular) distance , [1] big (circular) distance
AxmContiWriteClear(lCoordinate);
AxmContiSetAxisMap(lCoordinate, lSize, lAxesNo);
AxmContiBeginNode (lCoordinate);
AxmCircleRadiusMove(lCoordinate, lAxesNo, dRadius, dEndPos, dMaxVelocity, dMaxAccel,
dMaxDecel, uCWDdir, uShortDistance);
AxmContiEndNode (lCoordinate);
```

```
// Input data on interpolation queue to carry out circular interpolation as much as 200 degrees with
center of (200,200) from the current position using 0 and 2 axis
long lSize = 2;
long lAxesNo = {0,2};
double dCenPos[2];
long lCoordinate = 0;
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 uCWDdir = 0; // [0] Ccw direction, [1] Cw direction
AxmContiBeginNode (lCoordinate);
AxmCircleAngleMove(lCoordinate, lAxesNo, dCenPos, dAngle, dMaxVelocity, dMaxAccel,
dMaxDecel, uCWDdir);
AxmContiEndNode (lCoordinate);
```

```
// Input data on interpolation queue to carry out circular interpolation from the current position to
(150,150) passing (100,50) and using 0 and 1 axis
long lSize = 2;
long lAxesNo = {0,1};
double dMidPos[2];
double dEndPos[2];
long lCoordinate = 0;
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;
AxmContiWriteClear(lCoordinate);
AxmContiSetAxisMap(lCoordinate, lSize, lAxesNo);
AxmContiBeginNode (lCoordinate);
AxmCirclePointMove (lCoordinate, lAxesNo, dMidPos, dEndPos, dEndXPosition, dEndYPosition,
dMaxVelocity, dMaxAccel, dMaxDecel);
AxmContiEndNode (lCoordinate);
```

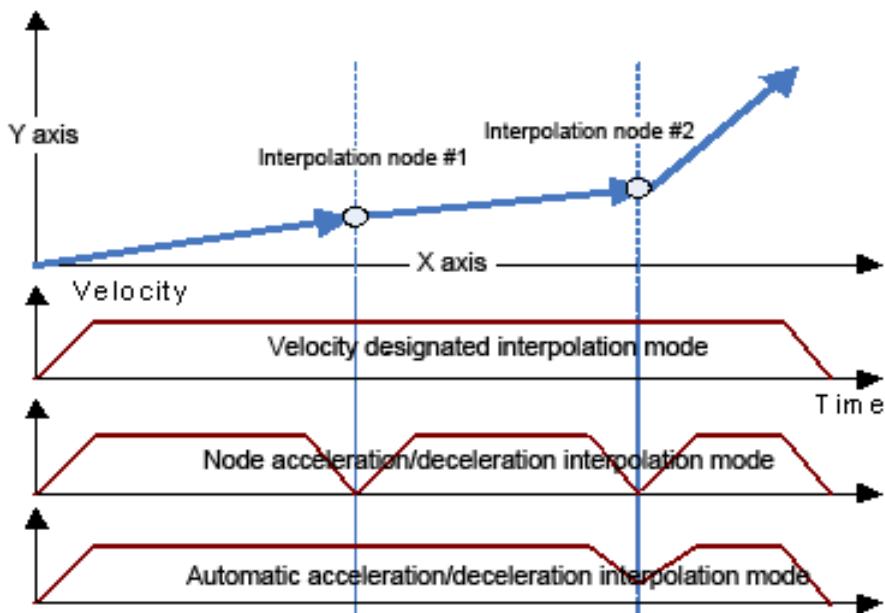
After input data into interpolation, when AxmContiStart API is called, continuous interpolation drive is started using the stored data. Here, velocity profile mode in drive of continuous interpolation can be

designated, and there are velocity designated interpolation mode, node acceleration/deceleration interpolation mode, and automatic acceleration/deceleration interpolation mode.

Velocity designated interpolation mode means a general continuous interpolation drive that progresses continuous interpolation drive with maintaining velocity when it reaches to the designated velocity after start drive and acceleration. Node acceleration/deceleration interpolation mode is carried out at each node of continuous interpolation, and automatic acceleration/deceleration interpolation mode is to accelerate/decelerate automatically in the section of sharp curve occurrence, even though no velocity change along with continuous path.

```
//Start continuous interpolation using stored interpolation data.
long lCoordinate = 0;
DWORD dwProfileset = 0;      // (0): Velocity designated interpolation mode
//(1): node acceleration/deceleration interpolation mode
//(2): automatic acceleration/deceleration interpolation mode
AxmContiStart (lCoordinate, dwProfileset, 0);
```

If continuous interpolation started like this is needed to exit arbitrary, use [AxmMoveSStop](#) API.



```
//Exit continuous interpolation drive.
long lCoordinate = 0
AxmMoveSStop (lCoordinate); // Input master axis of lCoordinate into factor
```

In addition to APIs previously mentioned, to carry out more various drive during continuous interpolation, following APIs are provided additionally. [AxmContiReadFree](#) API is provided to verify whether interpolation drive queue is cleared, and [AxmContiReadIndex](#) API is provided to verify what number of data is in drive on interpolation queue.

```
//Verify whether interpolation drive queue is cleared.
long lCoordinate = 0;
```

```
DWORD uQueueFree;
AxmContiReadFree (ICoordinate, &uQueueFree);
if(uQueueFree) { printf("Interpolation queue is cleared."); }
else { printf("Interpolation queue is not cleared."); }
```

```
//Verify what number of data is inputted into chip.
long lCoordinate = 0;
long lQueueIndex;
AxmContiReadIndex (ICoordinate, &lQueueIndex);
printf("In drive of interpolation currently Index : %d \n", lQueueIndex);
```

API is described that verifies index in drive of interpolation currently above. But, if it is in actual drive, there will be something weird. If it is in drive of interpolation with inputting 10 data, naturally when it is in first drive, it has to be 1. But it is 1 for a while and is changed to 2, and it always will have the value that is bigger by 1 than data number in actual drive. [AxmContiReadIndex](#) API which verifies index of data sent from memory to chip has the value that is bigger by 1 than actual because motion chip always has 1 more extra data besides the data in drive currently, in process of input data from interpolation data queue memory into motion chip internally.

Similar to this API, there is [AxmContiGetNodeNum](#) API that verifies node number in drive currently, and it returns data number in actual drive. Also, [AxmContiGetTotalNodeNum](#) API is used to verify total number of data that are in interpolation drive queue. If it is needed to verify what number of node is in drive currently out of what number of total interpolation drive node, do the following.

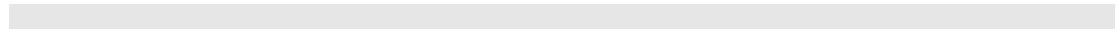
```
//Verify what number of node is in drive currently.
long lCoordinate = 0;
long lNodeNum, lNodeTotal;
AxmContiGetNodeNum (ICoordinate, &lNodeNum);
AxmContiGetTotalNodeNum (ICoordinate, &lNodeTotal);
printf("Currently %d number of node out of total of %d number is in drive of interpolation, \n",
lNodeTotal, lNodeNum);
```

To verify whether continuous interpolation drive is in progress currently, use [AxmContiIsMotion](#) API.

```
//Verify whether it is in drive of interpolation currently.
long lCoordinate = 0;
DWORD ulnMotion;
AxmContiIsMotion (ICoordinate, &ulnMotion);
if(ulnMotion) { printf("It is in drive of continuous interpolation."); }
else { printf("It is not in drive of continuous interpolation."); }
```

If continuous interpolation drive is exited, interpolation queue is to be clear, but sometimes it needs to clear in progress of interpolation drive. In such case, [AxmContiWriteClear](#) API is used.

```
//Clear data of interpolation queue.
long lCoordinate = 0;
AxmContiWriteClear (ICoordinate);
```



```

// Ex13_AXM_ContilInterpolation.cpp : Defines the entry point for the console application.
// Start continuous interpolation drive after home setting of 2 axes mechanical part .
#include "stdafx.h"
#include "AXL.h"
#include <conio.h>
#include "stdio.h"

#define AXIS_0
#define AXIS_1 1

void main(void)
{
    // Initialize library..
    // 7 means IRQ. IRQ is set automatically in PCI.
    long lNodeTotal;

    DWORD Code = AxlOpen(7);
    if (Code == AXT_RT_SUCCESS)
    {
        printf("Library is initialized.\n");

        // Inspect whether motion module exists or not
        DWORD dwStatus;
        Code = AxmInfoIsMotionModule (&dwStatus);
        if(Code == AXT_RT_SUCCESS)
        {
            if(dwStatus == STATUS_EXIST)
            {
                printf("Motion module exists.\n");

                for(int nAxisNo = 0; nAxisNo<2; nAxisNo++)
                {
                    // Set Active level of +End limit and -End limit on axis 0 to HIGH and emergency stop.

AxmSignalSetLimit(nAxisNo, 0, HIGH, HIGH);
                    // Set input level of inposition signal on axis 0 to HIGH.
AxmSignalSetInpos (nAxisNo, HIGH);
                    // Set input level of alarm signal on axis 0 to LOW.
AxmSignalSetServoAlarm (nAxisNo, LOW);
                    // Set Active input level of ESTOP signal on axis 0 to HIGH.
AxmSignalSetStop (nAxisNo, 0, HIGH);

                    // Make ordered value on axis 0 to be mm unit.
AxmMotSetMoveUnitPerPulse(nAxisNo, 10, 10000);
                    // Set initial velocity on axis 0 to 1. Default : 1
AxmMotSetMinVel(nAxisNo, 1);
                    // Make pulse output method on axis 0 to TwoCwCcwHigh.
AxmMotSetPulseOutMethod (nAxisNo, TwoCcwCcHigh);
                    // Set encoder input method on designated axis to 4 multiplication.
AxmMotSetEnclInputMethod (nAxisNo, ObverseSqr4Mode);

                    //Set to relative coordination drive
                }
            }
        }
    }
}

```

```

AxmMotSetAbsRelMode (nAxisNo, 1);
//Set to symmetrical S-curve velocity profile
AxmMotSetProfileMode (nAxisNo, 3);

// Set home search method
AxmHomeSetMethod (nAxisNo,-1,HomeSensor,1,2000.0,0.0);
// Set velocity that will be used during home search
AxmHomeSetVel(nAxisNo, 100, 20, 10, 10, 200, 40);

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

// Search home
printf("Start home search pressing any key.\r\n");
getch();
AxmHomeSetStart (AXIS_0);
AxmHomeSetStart (AXIS_1);

// Verify current home search progress rate
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("\r In home search... axis 0 : %d%, 1 axis : %d ", uHomeStepNumber_0,
           uHomeStepNumber_1);
}

printf("\rMove with straight line interpolation drive to relative position (50,25).");

// Carry out mapping of 0 and 1 axis to coordination system of 0 number that is to use
// for continuous interpolation drive.
long lCoordinate = 0;
long lSize = 2;
long lRealAxesNo[2] = {0,1};
double dCenPos[2]; double dEndPos[2]; double dMidPos[2];
AxmContiSetAxisMap(lCoordinate, lSize, lRealAxesNo);

//Delete existing continuous interpolation data.
AxmContiWriteClear(lCoordinate);

//Set 0 number coordination system to relative position drive.
AxmContiSetAbsRelMode(lCoordinate, 1);

double dpPosition[2] = {50,25};
AxmLineMove (lCoordinate, dpPosition, 100, 200, 200);

```

```

//Wait until all motions are exited.
BOOL WaitForDone = TRUE;
DWORD uStatus0, uStatus1;
while(WaitForDone) {
    AxmStatusReadInMotion (AXIS_0, &uStatus0);
    AxmStatusReadInMotion (AXIS_1, &uStatus1);
    if(uStatus0 == 0 && uStatus1 == 0)
        WaitForDone = FALSE;
}

// Input data on interpolation queue to carry out continuous drive to relative position
for(int i=0;i<5;i++)           //Repeat 5 times.
{
    // Input data on interpolation queue to move to (100,0). (velocity : 100,
    // acceleration : 200)

    AxmContiBeginNode (ICoordinate); //Start interpolation registration

    dpPosition[0] = 100; dpPosition[1] = 0;
    AxmLineMove (ICoordinate, dpPosition, 100, 200, 200);
    dCenPos[0] = 0; dCenPos[1] = 25; dEndPos[0] = 25; dEndPos[1] = 25;
    // Input data on interpolation queue to carry out circular interpolation
    // counter-clockwise with center of (0,25) to (25,25)
    AxmCircleCenterMove(ICoordinate, IRealAxesNo, dCenPos, dEndPos, 100, 200,
                           200, 0);

    // Input data on interpolation queue to move to (0,100)
    dpPosition[0] = 0; dpPosition[1] = 100;
    AxmLineMove (ICoordinate, dpPosition, 100, 200, 200);
    dEndPos[0] = 25; dEndPos[1] = -25;
    // Input data on interpolation queue to carry out circular interpolation with radius of
    // 25 to (-25,25)
    AxmCircleRadiusMove(ICoordinate, IRealAxesNo, 25, dEndPos ,100, 200, 200, 0,
                           0);

    // Input data on interpolation queue to move to (-100,0)
    dpPosition[0] = -100; dpPosition[1] = 0;
    AxmLineMove (ICoordinate, dpPosition, 100, 200, 200);
    dCenPos[0] = 0; dCenPos[1] = -25;
    // Input data on interpolation queue to carry out circular interpolation with a right
    // angle and with center of (0,-25)
    AxmCircleAngleMove(ICoordinate, IRealAxesNo, dCenPos, 90, 100, 200, 200, 0);

    // Input data on interpolation queue to move to (0,-100)
    dpPosition[0] = 0; dpPosition[1] = -100;
    AxmLineMove (ICoordinate, dpPosition, 100, 200, 200);
    dMidPos[0] = 7; dMidPos[1] = -18; dEndPos[0] = 25; dEndPos[1] = -25;

    // Input data on interpolation queue to carry out circular interpolation to (25,-25)
    // passing (7,-18) and using 0 and 1 axis
    AxmCirclePointMove (ICoordinate, IRealAxesNo, dMidPos, dEndPos, 100, 200,
                           200,0);
    AxmContiEndNode (ICoordinate); //End of interpolation queue registration
}

```

```

    }

AxmContiGetTotalNodeNum (ICoordinate, &INodeTotal);
printf("WnTotal of %d number of node is stored currently." , INodeTotal);
printf("WnStart continuous interpolation with velocity specified mode using stored
      interpolation data, pressing any key.");
getch();
AxmContiStart (ICoordinate, 0, 0);

printf("WnStop immediately continuous drive pressing any key.");
getch();
AxmMoveSStop (0); // Make to stop master axis of ICoordinate.
getch();

}

else
printf ("AxmlInfoIsMotionModule() : ERROR ( NOT STATUS_EXIST )
          code 0x%x\n",Code);
}

else
printf ("AxmlInfoIsMotionModule() : ERROR ( Return FALSE )   code 0x%x\n",Code);

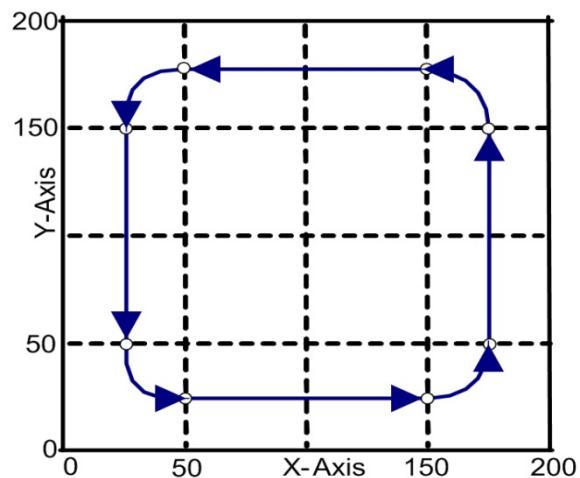
}

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

// Exit library.
if (AxlClose())
printf("Library is exited.\n");
else
printf("Library is not exited normally.\n");
}

```

** If an example of straight line interpolation drive above is carried out, following result can be verified.*



Expanded Motion Drive

So far, general various APIs for motion control are mentioned. In this chapter, APIs for advanced motion drive and positively necessary functions, even if they are not used commonly, will be mentioned.

[CRC Signal Setting](#)

[Override Drive](#)

[MPG Drive \(Manual Pulse Generation\)](#)

[Sync Drive](#)

[Gantry Drive](#)

- [1\) Setting and Cancellation of Gantry Drive](#)
- [2\) Home Search in Gantry Drive](#)

[Virtual Axis Mapping](#)

[Interrupt Setting](#)

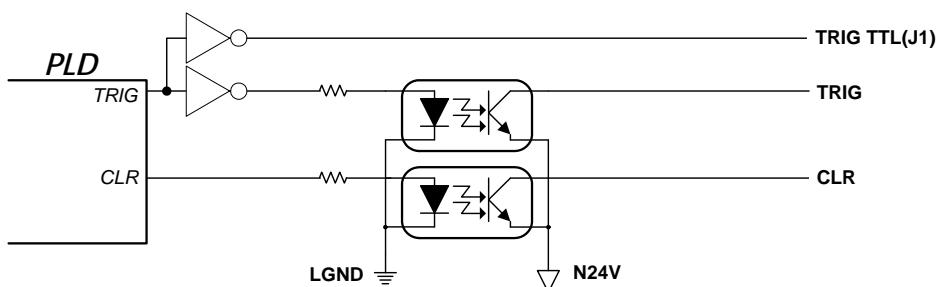
- [1\) Interrupt Management Method Setting](#)
- [2\) Interrupt Use Setting](#)

[Use of Trigger Signal](#)

CRC Signal Setting

In case of specified servo pack, at the point of drive completion or when limit sensor is detected, it occurs that deletes remaining pulse servo pack has with CRC(Current Remaining Clear) signal from outside. For this, signal pin is assigned and provided to connect to servo pack directly. Also, API is provided to set whether CRC signal is used, output level, whether limit signal is used, and so on. In addition, API is provided so that user can generate CRC signal arbitrarily.

Output level is being of reversed level of chip output, i.e. when chip output level is logic "1," CLR output is N24V, and when of logic "0," output is open collector. All the signals are isolated to external signal with photo-coupler same as universal output. These signals are outputted by external specified position or condition, and time delay is by the time passing through Photo-Coupler and by the time delay of PLD internal circuit. In case of CLR signal which is inputted PELM and NELM, approximately $6\mu s \sim 8\mu s$ of time delay is occurred. Since Photo Coupler output is a type of Normally Open, to obtain 24V level of voltage, user shall put on pull-up resistance. Pull-up resistance uses approximately $10k\Omega$ of resistance.



● EndLimit setting API

When mechanical part is put on End Limit, it can be set to generate CRC signal automatically. For this, [AxmCrcSetMaskLevel](#) API that sets CRC signal level and [AxmCrcSetEndLimit](#) API that sets CRC signal level corresponding to each EndLimit sensor.

SMC-2V03

```

//Set initially to use CRC on axis 0. If level setting is wrong, motor may not be operated.

//Set whether CRC signal is used on axis 0.
long lAxisNo = 0;
DWORD uLevel = HIGH;
AxmCrcSetMaskLevel (lAxisNo, uLevel, 6);
//IP does not use Method.

//Set whether CRC signal is used for the limit signal on axis 0
DWORD uPositiveLevel = HIGH;
DWORD uNegativeLevel = HIGH;
AxmCrcSetEndLimit(lAxisNo, uPositiveLevel, uNegativeLevel);
    
```

PCI-N804/404

```

//Set initially to use CRC on axis 0. If level setting is wrong, motor may not be operated.

//Set whether CRC signal is used on axis 0.
long lAxisNo = 0;
DWORD uLevel = HIGH;
AxmCrcSetMaskLevel (lAxisNo, uLevel, 6); //Set 100msec of CRC signal range
//Whether server remaining pulse remove signal output is used by limit/alarm/emergency
    
```

```
stop/sync stop signal on axis 0 (Only QI is used, IP is non-used)
//QI is PositiveLevel , NegativeLevel is non-used
AxmCrcSetEndLimit(IAxisNo, PositiveLevel, NegativeLevel);
```

● Forced output setting API to program

No CRC signal is needed only when it is detected by limit sensor. CRC signal is needed to generate by conditions, and in this case, [AxmCrcSetOutput](#) API is used.

```
// Set initially to use CRC on axis 0. If level setting is wrong, motor may not be operated.

// Set whether CRC signal is used on axis 0.
long IAxisNo = 0;
DWORD uLevel = HIGH;
AxmCrcSetMaskLevel (IAxisNo, uLevel);

//Send CRC signal on axis 0 by force.
DWORD uUse = ENABLE;           // 0: DISABLE, 1:ENABLE
AxmCrcSetOutput (IAxisNo, uUse);
```

```
// Ex14_AXM_AdvancedCRC.cpp : Defines the entry point for the console application.
// Set and verify CRC output.

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

#define AXIS_0 0

void main(void)
{
    // Initialize library..
    // 7 means IRQ. IRQ is set automatically in PCI.
    DWORD Code = AxIOpen(7);
    if (Code == AXT_RT_SUCCESS)
    {
        printf("Library is initialized.\n");

        // Inspect whether motion module exists or not
        DWORD dwStatus;
        Code = AxmlInfoIsMotionModule (&dwStatus);
        if(Code == AXT_RT_SUCCESS)
        {
            if(dwStatus == STATUS_EXIST)
            {
                printf("Motion module exists.\n");

                // Set Active level of +End limit and -End limit on axis 0 to HIGH.
                AxmSignalSetLimit(nAxisNo, 0, HIGH, HIGH);
            }
        }
    }
}
```

```

// Set input level of inposition signal on axis 0 to HIGH.
AxmSignalSetInpos(nAxisNo, HIGH);
// Set input level of alarm signal on axis 0 to LOW.
AxmSignalSetServoAlarm(nAxisNo, LOW);
// Set Active input level of ESTOP signal on axis 0 to HIGH.
AxmSignalSetStop(nAxisNo, 0, HIGH);

// Make ordered value on axis 0 to be mm unit.
AxmMotSetMoveUnitPerPulse(nAxisNo, 10, 10000);
// Set initial velocity on axis 0 to 1. Default : 1
AxmMotSetMinVel(nAxisNo, 1);
// Make pulse output method on axis 0 to TwoCwCcwHigh.
AxmMotSetPulseOutMethod(nAxisNo, TwoCcwCwHigh);
// Set encoder input method on designated axis to 4 multiplication.
AxmMotSetEnclnputMethod(AXIS_0, ObverseSqr4Mode);

//Servo On
AxmSignalServoOn(AXIS_0, ENABLE);

// Set initially to use CRC on axis 0.
// If level setting is wrong, motor may not be operated.
// Set whether CRC signal is used on axis 0.
long lAxisNo = 0;
DWORD uLevel = HIGH;
AxmCrcSetMaskLevel(lAxisNo, uLevel, 6);
// IP does not use Method.
//Set whether CRC signal is used for the limit signal on axis 0
DWORD uPositiveLevel = HIGH;
DWORD uNegativeLevel = HIGH;

//QI is uPositiveLevel, uNegativeLevel is non-used
AxmCrcSetEndLimit(lAxisNo, uPositiveLevel, uNegativeLevel);
// help Message
printf("[INFORMATION]*****\n");
printf("[ESC] : Exit \n");
printf("[1] : Output to CRC HIGH \n");
printf("[2] : Output to CRC LOW \n");
printf("*****\n");

DWORD uCRCOnOff = 2;
BOOL fExit = FALSE;
while (!fExit) // Infinite loop
{
    if (kbhit()) // Press any key
    {
        int ch = getch();
        switch (ch)
        {
            case 27: // Esc key
                fExit = TRUE;
                AxmSignalServoOn(AXIS_0, DISABLE); //Servo Off
                break;
            case '1': // HIGH output of CRC signal by program and by force
        }
    }
}

```

```
        AxmCrcSetOutput(AXIS_0, ENABLE);
        break;

        case '2':      // LOW output of CRC signal by program and by force
            AxmCrcSetOutput(AXIS_0, DISABLE);
            break;
        }
    }
//Verify current CRC output.

AxmCrcGetOutput(AXIS_0, &uCRCOnOff);
printf("WrCRC output : %x",uCRCOnOff);

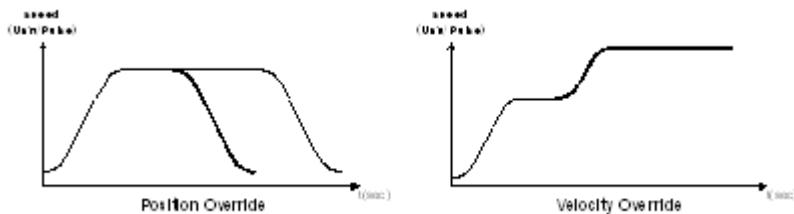
} //End of While()
}
else
    printf ("AxmInfoIsMotionModule() : ERROR ( NOT STATUS_EXIST )
            code 0x%x\n",Code);
}
else
    printf ("AxmInfoIsMotionModule() : ERROR ( Return FALSE )   code 0x%x\n",Code);
}
else
    printf ("AxlOpen() : ERROR   code 0x%x\n",Code);

// Exit library.
if (AxlClose())
    printf("Library is exited.\n");
else
    printf("Library is not exited normally.\n");

}
```

Override Drive

In this chapter, it will introduce velocity and position override drive. Override drive means change of work velocity during motion is in progress, and change of target distance during motion is in progress.



Position override uses [AxmOverridePos](#) API and velocity override uses [AxmOverrideVel](#) API. Override API must be called during in drive and is ignored when called after drive is ended.

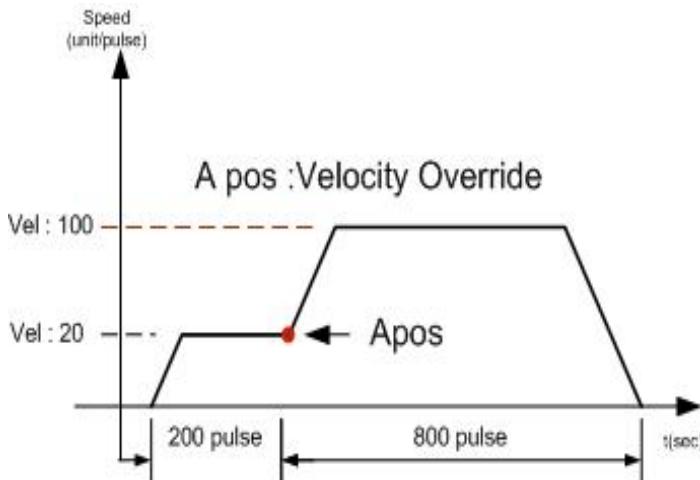
```
//Move axis 0 to 100 of absolute coordination with symmetric S-curve drive mode (velocity 100,
acceleration 200, exit API at start point)
long lAxisNo = 0;
AxmMotSetAbsRelMode (lAxisNo, 0);
AxmMotSetProfileMode (lAxisNo, 3);
AxmOverrideSetMaxVel(lAxisNo, 500); // Set maximum velocity of override velocity
// This API must be used to increase velocity than of initially assigned.
AxmMoveStartPos (lAxisNo, 100, 100, 200, 200);

AxmOverrideVel (lAxisNo, 500); // Adjust velocity to 500
// Adjust to 2000 of previous absolute coordination before axis 0 drive is ended
double dOverridePosition = 2000;
AxmOverridePos (lAxisNo, dOverridePosition);

// Adjust to 50 of previous velocity before axis 0 drive is ended
double dOverrideVelocity = 50;
AxmOverrideVel (lAxisNo, dOverrideVelocity);
```

[AxmOverrideVelAtPos](#) API is that carries out velocity override at the position of input velocity to a certain position point and override, and can be used to override after passing a specified position. This API has a difference that is not the one that changes in drive but that starts drive with corresponding information at stop state.

```
// Move axis 0 to 1000, and carry out override with velocity of 100 when passing 200 position.
long lAxisNo = 0;
double dMaxVelocity = 20;
double dMaxAccel = 100;
double dMaxDecel = 100;
double dOverrideVelocity = 100;
double dOverridePosition = 200;
double dPosition = 1000;
long lITarget = 1; // [0] Command Position , [1] Actual Position
AxmOverrideVelAtPos (lAxisNo, dPosition, dMaxVelocity, dMaxAccel, dMaxDecel,
dOverridePosition, dOverrideVelocity, lITarget);
```



```
// Ex15_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)
{
    // Initialize library..
    // 7 means IRQ. IRQ is set automatically in PCI.
    DWORD Code = AxlOpen(7);
    if (Code == AXT_RT_SUCCESS)
    {
        printf("Library is initialized.\n");

        // Inspect whether motion module exists or not
        DWORD dwStatus;
        Code = AxmInfoIsMotionModule (&dwStatus);
        if(Code == AXT_RT_SUCCESS)
        {
            if(dwStatus == STATUS_EXIST)
            {
                printf("Motion module exists.\n");

                // Set Active level of +End limit and -End limit on axis 0 to HIGH.
                AxmSignalSetLimit(nAxisNo, 0, HIGH, HIGH);
                // Set input level of inposition signal on axis 0 to HIGH.
                AxmSignalSetInpos (nAxisNo, HIGH);
                // Set input level of alarm signal on axis 0 to LOW.
                AxmSignalSetServoAlarm (nAxisNo, LOW);
                // Set Active input level of ESTOP signal on axis 0 to HIGH.
            }
        }
    }
}
```

```

AxmSignalSetStop (nAxisNo, 0, HIGH);

// Make ordered value on axis 0 to be mm unit.
AxmMotSetMoveUnitPerPulse(nAxisNo, 10, 10000);
// Set initial velocity on axis 0 to 1. Default : 1
AxmMotSetMinVel(nAxisNo, 1);
// Make pulse output method on axis 0 to TwoCwCcwHigh.
AxmMotSetPulseOutMethod (nAxisNo, TwoCcwcwHigh);
// Set encoder input method on designated axis to 4 multiplication.
AxmMotSetEnclnputMethod (AXIS_0, ObverseSqr4Mode);

//Use relative S-curve velocity profile
AxmMotSetAbsRelMode (AXIS_0, 1);
AxmMotSetProfileMode (AXIS_0, 3);

//Servo On
AxmSignalServoOn(AXIS_0, ENABLE);

// help Message
printf("[INFORMATION]*****\r\n");
printf("[ESC] : Exit \r\n");
printf("[1] : Start drive with velocity of 100 towards relative coordination 100 \r\n");
printf("[2] : Override position to 200 \r\n");
printf("[3] : Override velocity to 50 \r\n");
printf("[4] : Start drive up to -200 and override velocity to 50 at -100 \r\n");
printf("*****\r\n");

BOOL fExit = FALSE;
while (!fExit) // Infinite loop
{
    if (kbhit()) // Press any key
    {
        int ch = getch();
        switch (ch)
        {
            case 27: // Esc key
                fExit = TRUE;
                AxmSignalServoOn(AXIS_0, DISABLE); //Servo Off
                break;
            case '1': // Move axis 0 to 100 position with relative coordination with
                      // symmetric S-curve drive mode
                AxmOverrideSetMaxVel(AXIS_0, 500); // Set maximum velocity of
                // override velocity
                AxmMoveStartPos (AXIS_0, 100, 100, 200, 200);
                break;
            case '2': // Adjust to 200 of coordination before drive ends on axis 0
                AxmOverridePos (AXIS_0, 200);
                break;
            case '3': // Adjust velocity to 50 before drive ends on axis 0
                AxmOverrideVel (AXIS_0, 50);
                break;
            case '4': // axis 0 starts drive up to 200, and override velocity to 50
        }
    }
}

```

```
        when of 100 position.
        AxmOverrideVelAtPos (AXIS_0, -200, 100, 200, 200, -100,50,
                                COMMAND);
    }
}
} //End of While()
}
else
printf ("AxmlInfoIsMotionModule () : ERROR ( NOT STATUS_EXIST ) code
0x%x\n",Code);
}
else
printf ("AxmlInfoIsMotionModule () : ERROR ( Return FALSE ) code 0x%x\n",Code);

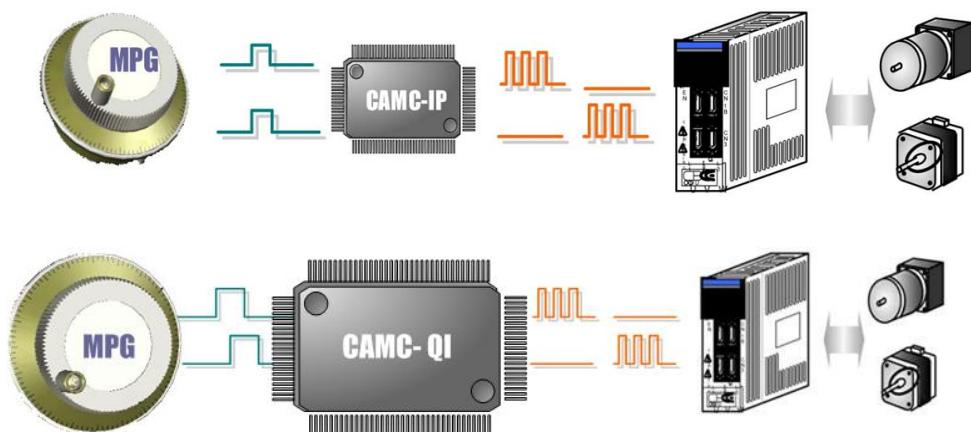
}

// Exit library.
if (AxlClose())
    printf("Library is exited.\n");
else
    printf("Library is not exited normally.\n");

}
```

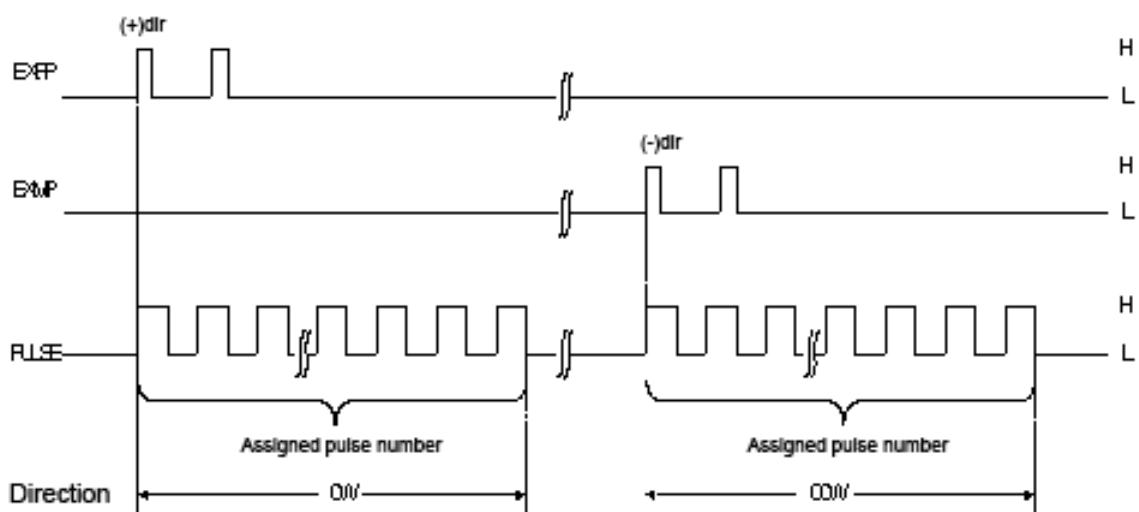
MPG Drive (Manual Pulse Generation)

Direction signal is decided by input of EXPP, EXMP signal, and number of pulse is decided by input option setting resister. Output pulse velocity is outputted by set Object Speed.



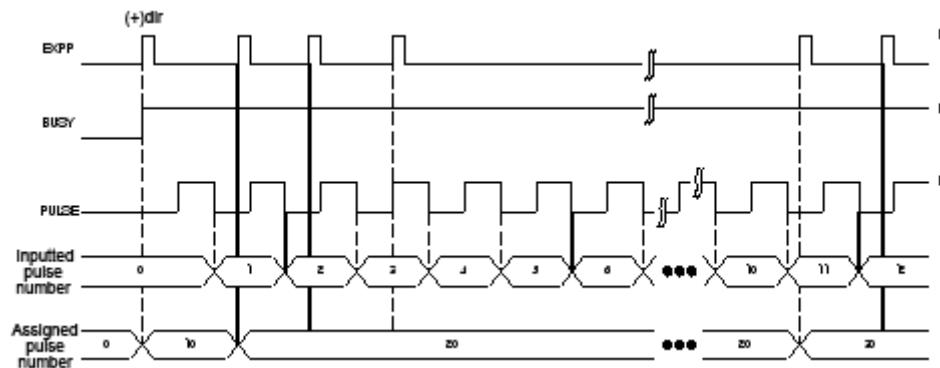
● Position drive by EXPP/EXMP (CAMC-IP)

Preset Mode outputs pulse with position drive type as much as number of pulse that is set in advance at output point of pulse. If Edge signal of EXPP/EXMP signal is inputted in drive, it carries out override as much as number of pulse that is set. Following figure corresponds to the case when EXPP/EXMP input option is single phase.



But, if Edge signal is inputted continuously, it ignores by inputted signal or decides override in comparison with outputted pulse number.

For example, when 10 of assigned pulse number is driven, it shows following figure.



● Velocity drive by EXPP/EXMP (CAMC-QI)

Cont Mode outputs pulse during EXPP/EXMP signal is to be Active Level. Drive type is same with Velocity Drive. While EXPP signal is Low('0') and EXMP is High('1'), it drives to (+) direction, and while EXMP signal is Low('0') and EXPP is High('1'), it drives to (-) direction.

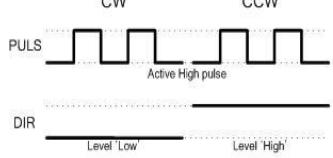
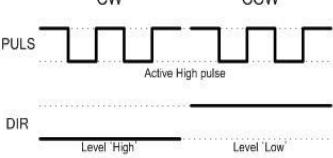
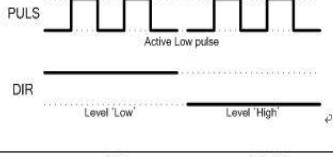
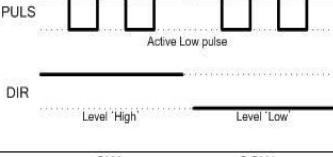
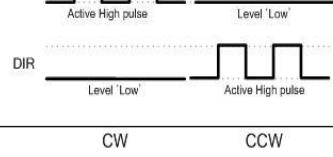
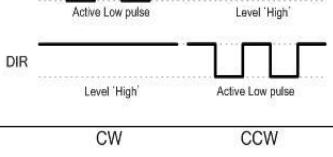
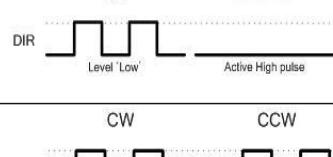
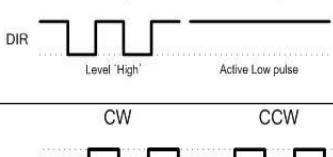
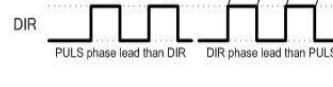
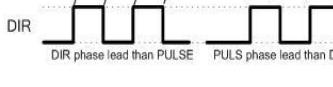
Input pulse type

Input method [2:0]	Input method [3] = '0'	Input method [3] = '1'
"000" (One pulse input) (Two phase 1 time)	<p>COUNT UP COUNT DOWN</p> <p>[CNTD] N N+1 N N-1</p> <p>COUNT UP COUNT DOWN</p> <p>[CNTD] N N+1 N N-1</p>	<p>COUNT DOWN COUNT UP</p> <p>[CNTD] N-1 N-2 N-1 N</p> <p>COUNT DOWN COUNT UP</p> <p>[CNTD] N N-1 N N+1</p>
"100" (Two pulse) "010" (Two phase 2 time)	<p>COUNT UP COUNT DOWN</p> <p>[CNTD] N N+1 N N-1</p> <p>COUNT UP COUNT DOWN</p> <p>[CNTD] N N+1 N N-1</p>	<p>COUNT DOWN COUNT UP</p> <p>[CNTD] N-1 N-2 N-1 N</p> <p>COUNT DOWN COUNT UP</p> <p>[CNTD] N N-1 N N+1</p>
"011" (Two phase 4 times)	<p>COUNT UP COUNT DOWN</p> <p>[CNTD] N N+1 N N+2 N N+3 N N+4</p> <p>COUNT UP COUNT DOWN</p> <p>[CNTD] N N+1 N N+2 N N+3 N N+4</p>	<p>COUNT DOWN COUNT UP</p> <p>[CNTD] N-1 N-2 N-3 N-4 N-5 N-6 N-7 N-8</p> <p>COUNT DOWN COUNT UP</p> <p>[CNTD] N N-1 N N-2 N N-3 N N-4</p>

● Interface with MPG outside

Two signals of EXPP and EXMP is used to input signal of external rotary encoder of SMC-2V03, PCI-N804/404 and connected with CAMC-QI chip through maximum of 1.2MHz high speed photo coupler.

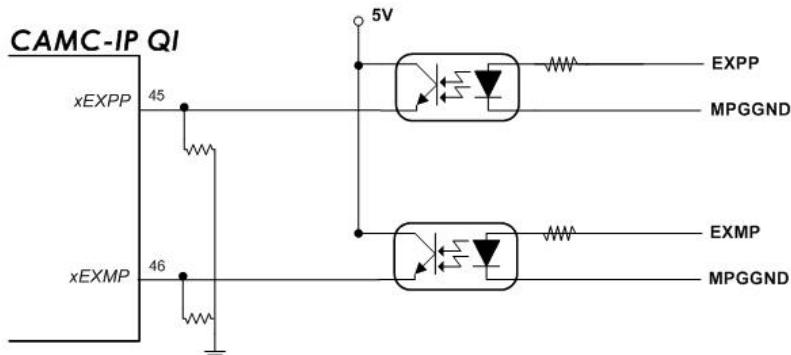
Output pulse type

[UCFG3][2~0]	[UCFG3][3] = '0'	[UCFG3][3] = '1'
"000" ^φ (One pulse mode)	CW CCW PULS Active High pulse DIR Level 'Low' Level 'High' 	CW CCW PULS Active High pulse DIR Level 'High' Level 'Low' 
"001" ^φ (One pulse mode)	CW CCW PULS Active Low pulse DIR Level 'Low' Level 'High' 	CW CCW PULS Active Low pulse DIR Level 'High' Level 'Low' 
"010" ^φ (Two pulse mode)	CW CCW PULS Active High pulse Level 'Low' DIR Level 'Low' Active High pulse 	CW CCW PULS Active Low pulse Level 'High' DIR Level 'High' Active Low pulse 
"011" ^φ (Two pulse mode)	CW CCW PULS Active High pulse Level 'Low' DIR Level 'Low' Active High pulse 	CW CCW PULS Active Low pulse Level 'High' DIR Level 'High' Active Low pulse 
"100" ^φ (Two phase mode)	CW CCW PULS DIR PULS phase lead than DIR PULS phase lead than PULSE 	CW CCW PULS DIR DIR phase lead than PULSE PULS phase lead than DIR 

● Interface with MPG outside

Two signals of EXPP and EXMP is used to input signal of external rotary encoder of SMC-2V03 and PCI-N804/404, and connected with SMC-2V03 AND PCI-N804/404 chip through maximum of 1.2MHz high speed photo coupler.

Output pulse type



PCI-N804/404

MPG input signal is not inputted individually on each axis but used jointly by two axes. MPG signal inputted on axis 0 is used on axis 2 jointly, inputted on axis 1 is used on axis 3, inputted on axis 4 is used on axis 6, inputted on axis 5 is used on axis 7 jointly. That is, MPG signal input pin of external connect J2/J4 connected with 2/3/6/7 axes is not used.

● EXPP/EXMP signal verification

There is no API for verification of current state of EXPP / EXMP signal, but since Mechanical Signal of number 8 and 9 bit is connected to EXMP and EXPP each, current state can be verified through this. To verify Mechanical Signal, [AxmStatusReadMechanical](#) API is used, and EXPP and EXMP are verified as following with bit operation.

SMC–	IPMECHANICAL_EXMP_LEVEL	Verify EXMP signal on SMC–2V03
2V03	IPMECHANICAL_EXPP_LEVEL	Verify EXPP signal on SMC–2V03
PCI–	QIMECHANICAL_EXMP_LEVEL	Verify EXMP signal on PCI–N804/404
N804/404	QIMECHANICAL_EXPP_LEVEL	Verify EXPP signal on PCI–N804/404

SMC–2V03

```
// Verify EXPP / EXMP signal on SMC–2V03 chip.
AxmStatusReadMechanical (AXIS_0, &MechanicalStatus);
if(MechanicalStatus & IPMECHANICAL_EXMP_LEVEL) { bExmp = 1; } else { bExmp = 0; }
if(MechanicalStatus & IPMECHANICAL_EXPP_LEVEL) { bExpp = 1; } else { bExpp = 0; }
printf("WrEXPP : %x , EXMP : %x ",bExpp,bExmp);
```

IPMECHANICAL_EXMP_LEVEL , IPMECHANICAL_EXPP_LEVEL means
bit 8 and 9 of MechanicalStatus on SMC–2V03.

PCI–N804/404

```
// Verify EXPP / EXMP signal on PCI–N804/404 chip.
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("WrEXPP : %x , EXMP : %x ",bExpp,bExmp);
```

QIMECHANICAL_EXMP_LEVEL , QIMECHANICAL_EXPP_LEVEL means
bit 11 and 12 of MechanicalStatus on PCI–N804/404.

● Use of MPG drive API

When MPG drive is set, from then, it drives by signal input to EXPP EXMP pin. Set as follow to use position drive mode.

```
//Set axis 0 to move by 10 per input pulse of position drive mode
AxmMPGSetEnable(0, MPG_DIFF_ONE_PHASE, 1, 10, 200, 400);
AxmMPGSetRatio(0, 10, 4096 ); //Not supported on IP.
```

```
// Return Mpg signal setting on axis 0.
long lplInputMethod, lpDriveMode;
double dpVel, dpAccel, dpDecel, dpMPGPos;
double dMPGdenominator, dMPGnumerator;

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

Here, dMPGdenominator = 4096 and MPGnumerator=1 mean output per 1 pulse with 1:1 as is if it is 200 pulse per rotation of MPG. If dMPGdenominator = 4096 and MPGnumerator=2, it means that send output per 2 pulses with 1:2.

Formula of MPG PULSE = ((Numerator) * (Denominator)/ 4096) is to be outputted to chip inside. If velocity mode needs to be used, follow the following.

```
//Set axis 0 to drive with velocity drive mode by MPG input
AxmMPGSetEnable(0, MPG_DIFF_ONE_PHASE, 0, 10, 200,400); //QI to 0, Set IP to 2.
AxmMPGSetRatio(0, 10, 4096 ); // Not supported on IP

// Return Mpg signal setting on axis 0.
long lplInputMethod, lpDriveMode;
double dpVel, dpAccel, dpDecel, dpMPGPos;
double dMPGdenominator, dMPGnumerator;

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

After these setting, MPG drive is carried out by EXPP / EXMP signal input. If needed to clear MPG drive, do the following.

```
//Clear MPG drive on axis 0.
long lAxisNo = 0;
AxmMPGReset(lAxisNo);
```

SMC-2V03

```
// Ex16_AXM_IPMPG.cpp : Defines the entry point for the console application.
// Carry out MPG drive.

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

#define AXIS_0 0

void main(void)
{
    // Initialize library..
    // 7 means IRQ. IRQ is set automatically in PCI.
```

```

DWORD Code = AxlOpen(7);
if (Code == AXT_RT_SUCCESS)
{
    printf("Library is initialized.\n");

    // Inspect whether motion module exists or not
    DWORD dwStatus;
    Code = AxmInfoIsMotionModule (&dwStatus);
    if(Code == AXT_RT_SUCCESS)
    {
        if(dwStatus == STATUS_EXIST)
        {
            printf("Motion module exists.\n");

            // Set Active level of +End limit and -End limit on axis 0 to HIGH.
            AxmSignalSetLimit(nAxisNo, 0, HIGH, HIGH);
            // Set input level of inposition signal on axis 0 to HIGH.
            AxmSignalSetInpos (nAxisNo, HIGH);
            // Set input level of alarm signal on axis 0 to LOW.
            AxmSignalSetServoAlarm (nAxisNo, LOW);
            // Set Active input level of ESTOP signal on axis 0 to HIGH.
            AxmSignalSetStop (nAxisNo, 0, HIGH);

            // Make ordered value on axis 0 to be mm unit.
            AxmMotSetMoveUnitPerPulse(nAxisNo, 10, 10000);
            // Set initial velocity on axis 0 to 1. Default : 1
            AxmMotSetMinVel(nAxisNo, 1);
            // Make pulse output method on axis 0 to TwoCwCcwHigh.
            AxmMotSetPulseOutMethod (nAxisNo, TwoCwCcwHigh);
            // Set encoder input method on designated axis to 4 multiplication.
            AxmMotSetEnclnputMethod (AXIS_0, ObverseSqr4Mode);

            //Use relative position S-curve velocity profile
            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_PRESET_MODE. \n");
            printf("[2] : MPG_CONT_MODE. \n");
            printf("*****\n");

            BOOL fExit = FALSE;
            DWORD MechanicalStatus;
            BOOL bExmp, bExpp;
            double dVelocity      = 2000;      //Setting value of velocity use
            double dAccel         = 4000;      // Setting value of acceleration use
            double dDecel         = 4000;      // Setting value of deceleration use
            long   lInputMethod;
        }
    }
}

```

```

long lDriveMode;
long lDirMode = 0; // MPG_SIGNAL_DIR (0), MPG_USER_DIR(1)
long lUserDir = 0; // MPG_CW_USER_DIR(0), MPG_CCW_USER_DIR(1)
double dMPGPosition = 10; // Moving distance if MPG_PRESET_MODE mode
double dMPGdenominator = 0; //Non-used on SMC-2V03
double dMPGnumerator = 0; //Non-used on SMC-2V03

while (!fExit) // Infinite loop
{
    if (kbhit()) // Press any key
    {
        int ch = getch();
        switch (ch)
        {
            case 27: // Esc key
                fExit = TRUE;
                AxmMPGReset(AXIS_0); // Exit MPG drive.
                AxmSignalServoOn(AXIS_0, DISABLE); //Servo Off
                break;

            case '1':
                //Exit MPG drive.
                AxmMPGReset(AXIS_0);

                printf("WnStart MPG drive with MPG_PRESET_MODE.Wn");
                lInputMethod = 0; //Input signal method MPG_DIFF_ONE_PHASE (0)
                lDriveMode = 1;
                //MPG_SLAVE_MODE(0), MPG_PRESET_MODE(1),
                // MPG_CONT_MODE(2)

                AxmMPGSetEnable(AXIS_0, lInputMethod, lDriveMode,
                    dMPGPosition, dVelocity, dAccel);
                break;

            case '2':
                //Exit MPG drive.
                AxmMPGReset(AXIS_0);

                printf("WnStart MPG drive with MPG_CONT_MODE.Wn");
                lInputMethod = 4;
                // Input signal method MPG_LEVEL_ONE_PHASE(4)
                lDriveMode = 2;
                // MPG_SLAVE_MODE(0), MPG_PRESET_MODE(1), MPG_CONT_MODE(2)

                AxmMPGSetEnable(AXIS_0, lInputMethod, lDriveMode,
                    dMPGPosition, dVelocity, dAccel);

                break;
        }
    }

    AxmStatusReadMechanical (AXIS_0, &MechanicalStatus);
    if(MechanicalStatus & IPMECHANICAL_EXMP_LEVEL) { bExmp = 1; }
    else { bExmp = 0; }
    if(MechanicalStatus & IPMECHANICAL_EXPP_LEVEL) { bExpp = 1; }
}

```

```

        else { bExpp = 0; }
        printf("WrEXPP : %x , EXMP : %x ",bExpp,bExmp);

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

}

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

// Exit library.
if (AxlClose())
printf("Library is exited.\n");
else
printf("Library is not exited normally.\n");

}

```

PCI-N804/404

```

// Ex17_AXM_QIMPG.cpp : Defines the entry point for the console application.
// Start MPG drive.

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

#define AXIS_0 0

void main(void)
{
    // Initialize library..
    // 7 means IRQ. IRQ is set automatically in PCI.
    DWORD Code = AxlOpen(7);
    if (Code == AXT_RT_SUCCESS)
    {
        printf("Library is initialized.\n");

        // Inspect whether motion module exists or not
        DWORD dwStatus;
        Code = AxmlInfoIsMotionModule (&dwStatus);
        if(Code == AXT_RT_SUCCESS)
        {
            if(dwStatus == STATUS_EXIST)

```

```

{
    printf("Motion module exists.\n");

    // Set Active level of +End limit and -End limit on axis 0 to HIGH.
    AxmSignalSetLimit(nAxisNo, 0, HIGH, HIGH);
    // Set input level of inposition signal on axis 0 to HIGH.
    AxmSignalSetInpos (nAxisNo, HIGH);
    // Set input level of alarm signal on axis 0 to LOW.
    AxmSignalSetServoAlarm (nAxisNo, LOW);
    // Set Active input level of ESTOP signal on axis 0 to HIGH.
    AxmSignalSetStop (nAxisNo, 0, HIGH);

    // Make ordered value on axis 0 to be mm unit.
    AxmMotSetMoveUnitPerPulse(nAxisNo, 10, 10000);
    // Set initial velocity on axis 0 to 1. Default : 1
    AxmMotSetMinVel(nAxisNo, 1);
    // Make pulse output method on axis 0 to TwoCwCcwHigh.
    AxmMotSetPulseOutMethod (nAxisNo, TwoCwCcwHigh);
    // Set encoder input method on designated axis to 4 multiplication.
    AxmMotSetEnclInputMethod (AXIS_0, ObverseSqr4Mode);

    // Use relative position S-curve velocity profile
    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;      //Setting value of velocity use
    double dAccel       = 4000;      // Setting value of acceleration use
    double dDecel       = 4000;      // Setting value of deceleration use
    long   lInputMethod;
    long   lDriveMode;
    long   lDirMode      =0;        // Non-used on PCI-N804/404
    long   lUserDir      =0;        // Non-used on PCI-N804/404
    double dMPGPosition = 10;       // Moving distance if MPG_PRESET_MODE mode
    double dMPGdenominator = 4095;
    // During MPG(input manual pulse generation device) drive, division value(calculated as +1 of input value)
    double dMPGnumerator = 0;
    // During MPG(input manual pulse generation device) drive, multiplication value(calculated as +1 of input
    value)

    while (!fExit) // Infinite loop
{

```

```

if (kbhit()) // Press any key
{
    int ch = getch();
    switch (ch)
    {
        case 27: // Esc key
            fExit = TRUE;
            AxmMPGReset(AXIS_0); // Exit MPG drive.
            AxmSignalServoOn(AXIS_0, DISABLE); // Servo Off
            break;

        case '1':
            // Exit MPG drive.
            AxmMPGReset(AXIS_0);

            printf("WnStart MPG drive with MPG_PRESET_MODE.Wn");
            lInputMethod = 0; // Input signal method 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, dAccel);
            AxmMPGSetRatio(AXIS_0, dMPGnumerator, dMPGdenominator );
            break;

        case '2':
            // Exit MPG drive.
            AxmMPGReset(AXIS_0);

            printf("Wn Start MPG drive with MPG_CONT_MODE.Wn");
            lInputMethod = 4; // Input signal method 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, dAccel);
            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("WrEXPP : %x , EXMP : %x ",bExpp,bExmp);

} //End of While()
}

```

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

}

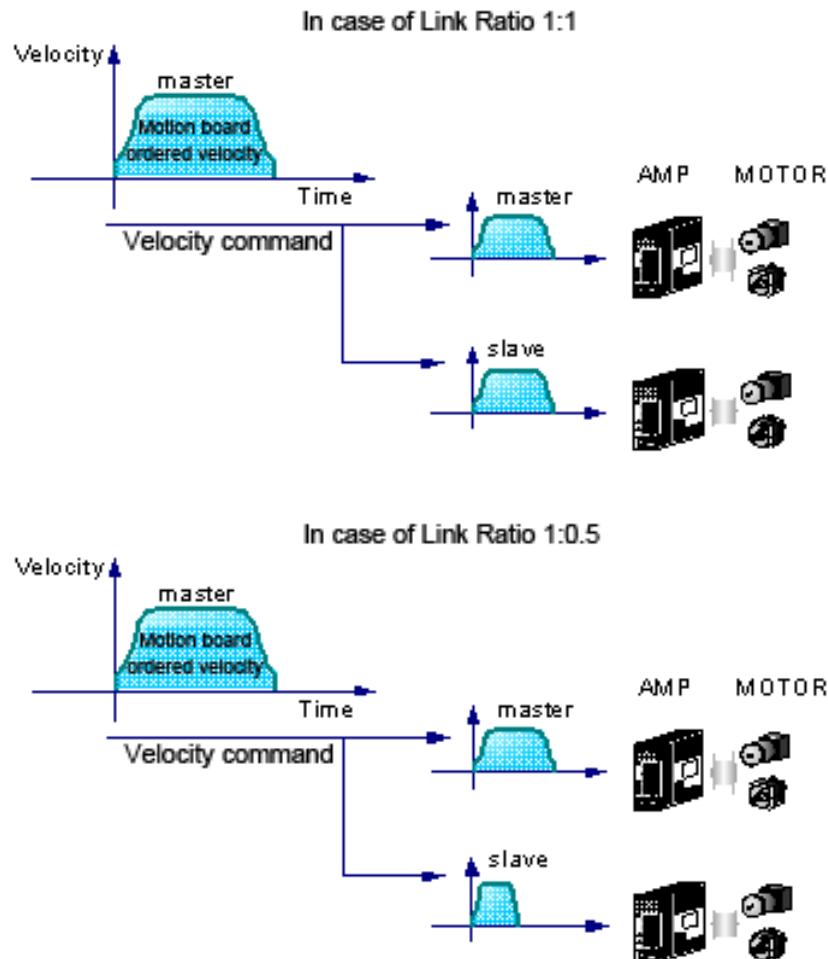
else
printf ("AxlOpen() : ERROR   code 0x%x\n",Code);

// Exit library.
if (AxlClose())
    printf("Library is exited.\n");
else
    printf("Library is not exited normally.\n");

}
```

Sync Drive

Sync drive is also called Electric Gear Mode, and it is a method to control independent two axes as one axis. User set a specific master and slave axis, and set drive ratio between two axes, then slave axis moves together with drive ratio set when master axis moves.



To carry out this sync drive, it needs to set through [AxmLinkSetMode API](#). Sync drive starts after [AxmLinkSetMode](#) API is called.

```
//Set to drive with 2:1 ratio, setting that axis 0 is to be Master and 1 axis is to be Slave
long IMasterAxisNo = 0;
long ISlaveAxisNo = 1;
double dSlaveRatio = 0.5;
AxmLinkSetMode (IMasterAxisNo, ISlaveAxisNo, dSlaveRatio);

//Drive axis 0 to as much as 100.
AxmMoveStartPos (AXIS_0, 100, 100, 200, 200);
```

If set this, while axis 0 drives as much as 100, 1 axis drives along as much as 50 with 2:1 ratio. To exit

sync drive, use [AxmLinkResetMode](#) API.

```
//Clear sync drive mode composed of 0 and 1 axis.
long lMasterAxisNo = 0;
AxmLinkResetMode (lMasterAxisNo);
```

```
// Ex18_AXM_ElectricGearMode.cpp : Defines the entry point for the console application.
// Carry out sync drive.
```

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

#define AXIS_0 0
#define AXIS_1 1

void main(void)
{
    // Initialize library..
    // 7 means IRQ. IRQ is set automatically in PCI.
    DWORD Code = AxlOpen(7);
    if (Code == AXT_RT_SUCCESS)
    {
        printf("Library is initialized.\n");

        // Inspect whether motion module exists or not
        DWORD dwStatus;
        Code = AxmlInfoIsMotionModule (&dwStatus);
        if(Code == AXT_RT_SUCCESS)
        {
            if(dwStatus == STATUS_EXIST)
            {
                printf("Motion module exists.\n");

                for(int nAxisNo = 0; nAxisNo<2; nAxisNo++)
                {
                    // Set Active level of +End limit and -End limit on axis 0 to HIGH.
                    AxmSignalSetLimit(nAxisNo, 0, HIGH, HIGH);
                    // Set input level of inposition signal on axis 0 to HIGH.
                    AxmSignalSetInpos (nAxisNo, HIGH);
                    // Set input level of alarm signal on axis 0 to LOW.
                    AxmSignalSetServoAlarm (nAxisNo, LOW);
                    // Set Active input level of ESTOP signal on axis 0 to HIGH.
                    AxmSignalSetStop (nAxisNo, 0, HIGH);
                    // Make ordered value on axis 0 to be mm unit.
                    AxmMotSetMoveUnitPerPulse(nAxisNo, 10, 10000);
                    // Set initial velocity on axis 0 to 1. Default : 1
                    AxmMotSetMinVel(nAxisNo, 1);
                    // Make pulse output method on axis 0 to TwoCwCcwHigh.
                    AxmMotSetPulseOutMethod (nAxisNo, TwoCcwCcwHigh);
                }
            }
        }
    }
}
```

```

// Set encoder input method on designated axis to 4 multiplication.
AxmMotSetEncInputMethod (nAxisNo, ObverseSqr4Mode);

//Use relative S-curve velocity profile
AxmMotSetAbsRelMode (AXIS_0, 1);
AxmMotSetProfileMode (AXIS_0, 3);

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

//Start sync drive.
long lMasterAxisNo = AXIS_0;
long lSlaveAxisNo = AXIS_1;
double dSlaveRatio = 0.5;
AxmLinkSetMode (lMasterAxisNo, lSlaveAxisNo,dSlaveRatio);

// help Message
printf("[INFORMATION]*****\n");
printf("Sync drive is set.\nDrive starts pressing any key.\n");
printf("*****\n");

//Wait until any key is pressed
getch();
//Drive axis 0 as much as 100.
AxmMoveStartPos (AXIS_0, 100, 100, 200, 200);
//Verify if it is in drive.
DWORD uStatus;
AxmStatusReadInMotion (AXIS_0, &uStatus);
while(uStatus) {
    AxmStatusReadInMotion (AXIS_0, &uStatus);
}

//Exit sync drive.
AxmLinkResetMode (AXIS_0);
//Servo Off
AxmSignalServoOn(AXIS_0, DISABLE);
AxmSignalServoOn(AXIS_1, DISABLE);
}

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

// Exit library.
if (AxlClose())
    printf("Library is exited.\n");
else
    printf("Library is not exited normally.\n");

```

```
}
```

Gantry Drive

To control Gantry drive system that two axes are linked mechanically showing on the picture below, Gantry drive API is provided. Using this API, if master axis (even number axis) set with Gantry control, corresponding slave axis is synchronized with master axis and driven. Even if drive or stop command is given to slave axis after Gantry setting, all commands are ignored.



- [1\) Setting and cancellation of Gantry Drive](#)
- [2\) Home Search in Gantry Drive](#)

1) Setting and Cancellation of Gantry Drive

To set Gantry drive, using [AxmGantrySetEnable](#) API, enable Gantry drive. Gantry system has a structure that each robot is arrayed side by side and linked mechanically. And home sensor of master and slave axis has a certain value of error. If home search is carried out, error that home sensor of two axes has is compensated at last. Error generated between sensors during machine assembly is to be set to dSIOffset.

```
//Start Gantry drive using 0 and 1 axis.
long lMasterAxisNo = 0;
long lSlaveAxisNo = 1;
DWORD uGantryOn;
DWORD bSIHomeUse = FALSE; // [0] : Select home search only for Home axis
// [1] : Select home search together for Home and Slave axis
double dSIOffset = 0; // Mechanical Offset value between home sensors of master
and slave
double dSIOffsetRange = 10; // Error limit between master home sensor and slave home
sensor during home search
AxmGantrySetEnable (lMasterAxisNo, lSlaveAxisNo, bSIHomeUse, dSIOffset, dSIOffsetRange);
// Setting value set directly by user can be read.
AxmGantryGetEnable (lMasterAxisNo, &SIHomeUse, &dSIOffset, &dSIORange, &uGantryOn);
```

After this setting, 0 and 1 axis are driven identically by command for axis 0 which is master axis. And all

commands for 1 axis are ignored. For home search, carry out home search API for axis 0 at the state.

```
//Carry out home search on Gantry drive.
long MasterAxisNo = 0;
long nHmDir = -1; //0 : (-) direction , 1 : (+) direction
DWORD nHmSig = 4; // [0]PELM, [1]NELM, [2]PSLM, [3]NSLM, [4]IN0, [5]IN1, [6]IN2, [7]IN3
BOOL bZphas = TRUE; //Whether Z phase is detected
double dHClrTime = 100.0; //Waiting time for encoder clear
double dHOffset = 0.0;
AxmHomeSetMethod(MasterAxisNo, nHmDir, nHmSig, bZphas, dHClrTime, dHOffset);

double dVelF = 10000; //Drive velocity towards forward direction of home search when
home sensor is not detected
double dVelA = 1000; // Drive velocity towards reverse direction of home search when
home sensor is detected
double dVelL = 500; //Velocity to be used for re-search after primary home search
double dvell = 10; //Velocity to be used during last home search
double dAccF = 20000; // Acceleration and deceleration to drive towards home sensor
direction of home search when home sensor is not detected
double dAccA = 10000; // Acceleration and deceleration to drive towards reverse direction
when home sensor is detected
AxmHomeSetVel(MasterAxisNo, dVelF, dVelA, dVelL, dvell, dAccF, dAccA);

AxmHomeSetStart(MasterAxisNo);
```

To exit Gantry drive, use [AxmGantrySetDisable](#) API.

```
//Exit Gantry drive.
long IMasterAxisNo = 0;
long ISlaveAxisNo = 1;
AxmGantrySetDisable(IMasterAxisNo, ISlaveAxisNo);
```

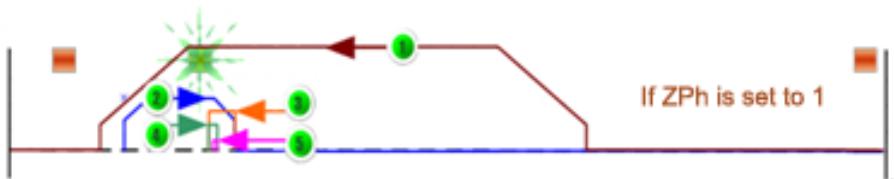
2) Home Search in Gantry Drive

Important thing in Gantry control is to carry out home search considering master and slave axis during home search. Generally, home sensor on master axis is searched first and then home sensor on slave axis is searched. At the beginning, if Gantry starts home drive after putting together with Gantry control of [AxmGantrySetEnable](#), drive is carried out and home search is carried out using one-axis home search API for master axis.

Setting Variable Related to Home Search	Variable Description
HmSig	Signal setting for home sensor Example) Home sensor(4), +Limit sensor(0), etc

HmLev	Active Level setting of home sensor Example) 0: A contact, 1: B contact
HmDir	Initial search direction setting during home sensor detection Example) 1: (+) direction, 0: (-) direction
Zphas	Setting of whether encoder Z phase is detected after home sensor detection
VelFirst	Initial high speed detection velocity during home search (if initial home sensor is not detected)
VelSecond	Velocity coming out from reverse direction after primary sensor detection during home search
VelThird	Velocity progressing re-search after primary sensor detection during home search
VelLast	Final detection velocity setting during home search [decision for accuracy of home search]
AccFirst	Initial high speed detection acceleration during home search
AccSecond	Acceleration coming out from reverse direction after primary sensor detection during home search
HClrTim	Waiting time before clear of Command and Actual(Encoder) position after home search completion
HOffset	Moving Offset value during machine home re-search after home search completion
NSWLimit	(-) Software Limit value setting to apply after home search completion
PSWLimit	(+) Software Limit value setting to apply after home search completion (If NSWLimit value is greater than or same with PSWLimits value, S/W Limit is not set)

● *Understanding of home search velocity setting and home search sequence by step
(Master, Slave use same velocity)*



1.

Use VelFirst, AccFirst

Carry out high speed search of home sensor towards HmDir direction, and slow down stop with HAccF deceleration

2.

Use VelSecond, AccSecond

Carry out Down Edge search of home sensor towards reverse HmDir direction, and slow down stop with HAccF

deceleration

3.

Use VelThird

Carry out Up Edge search of home sensor towards HmDir direction, and emergency stop

4.

Use VelThird

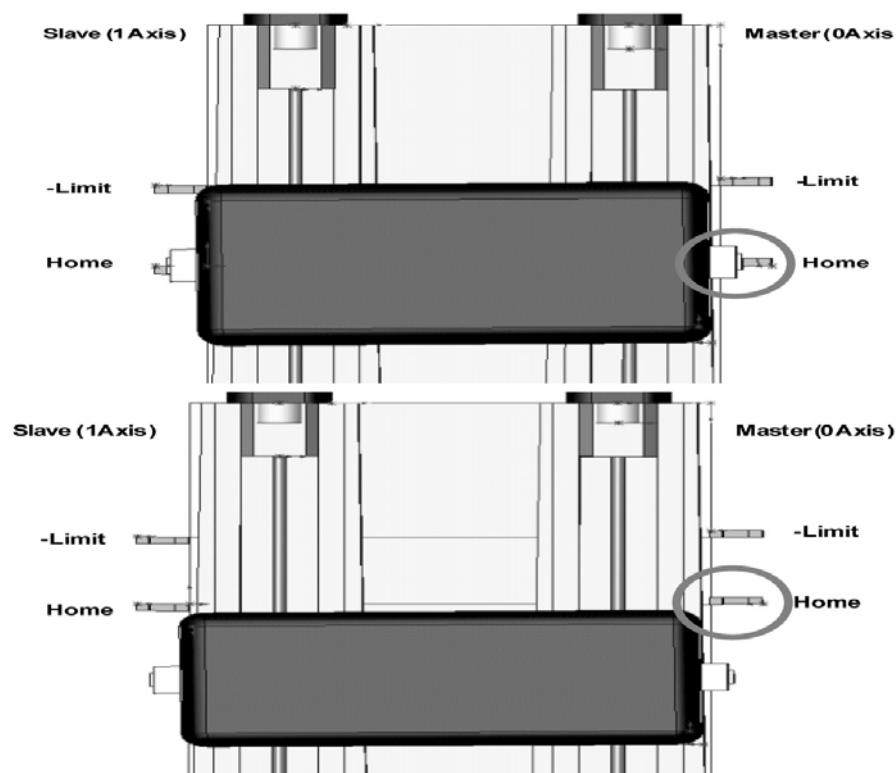
Carry out Down Edge search on Z phase towards reverse HmDir direction, and emergency stop

5.

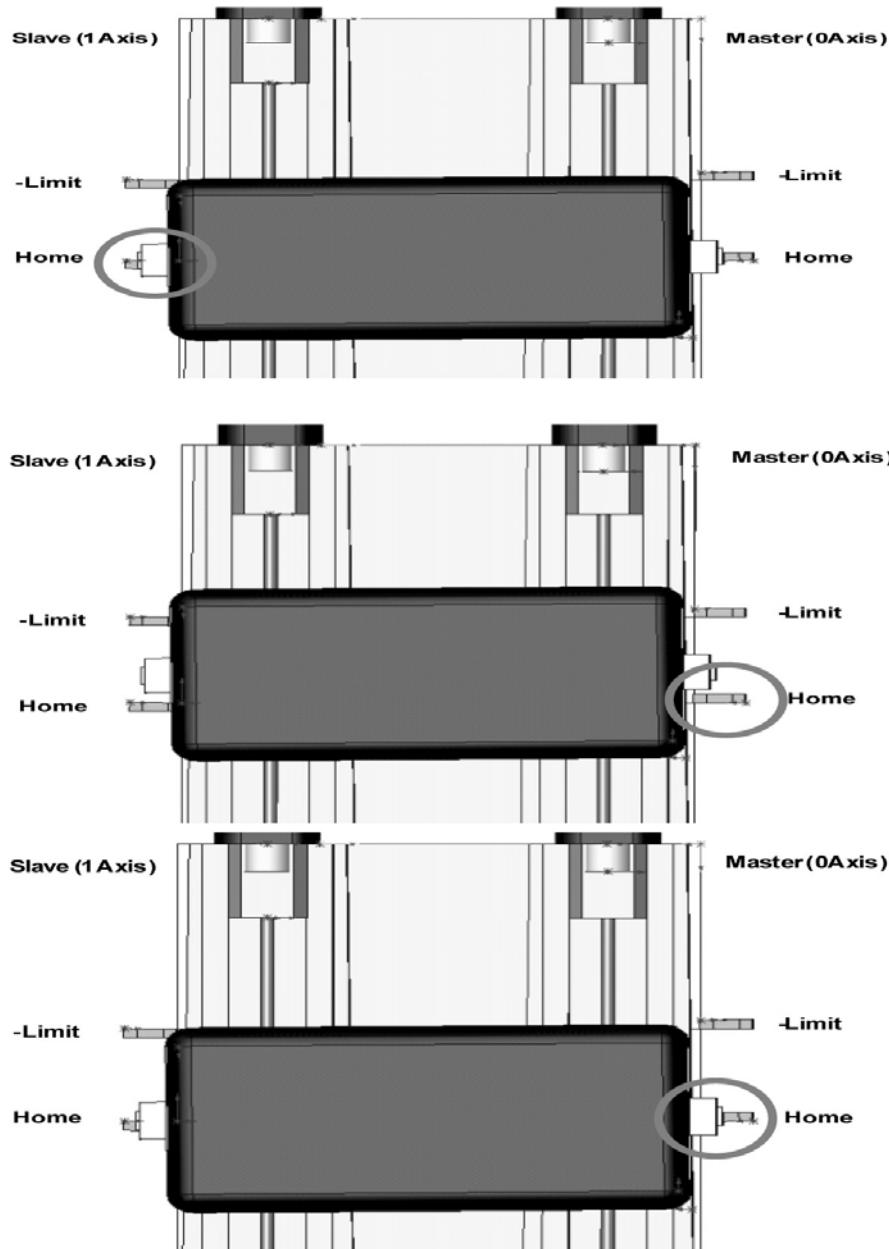
Use VelLast

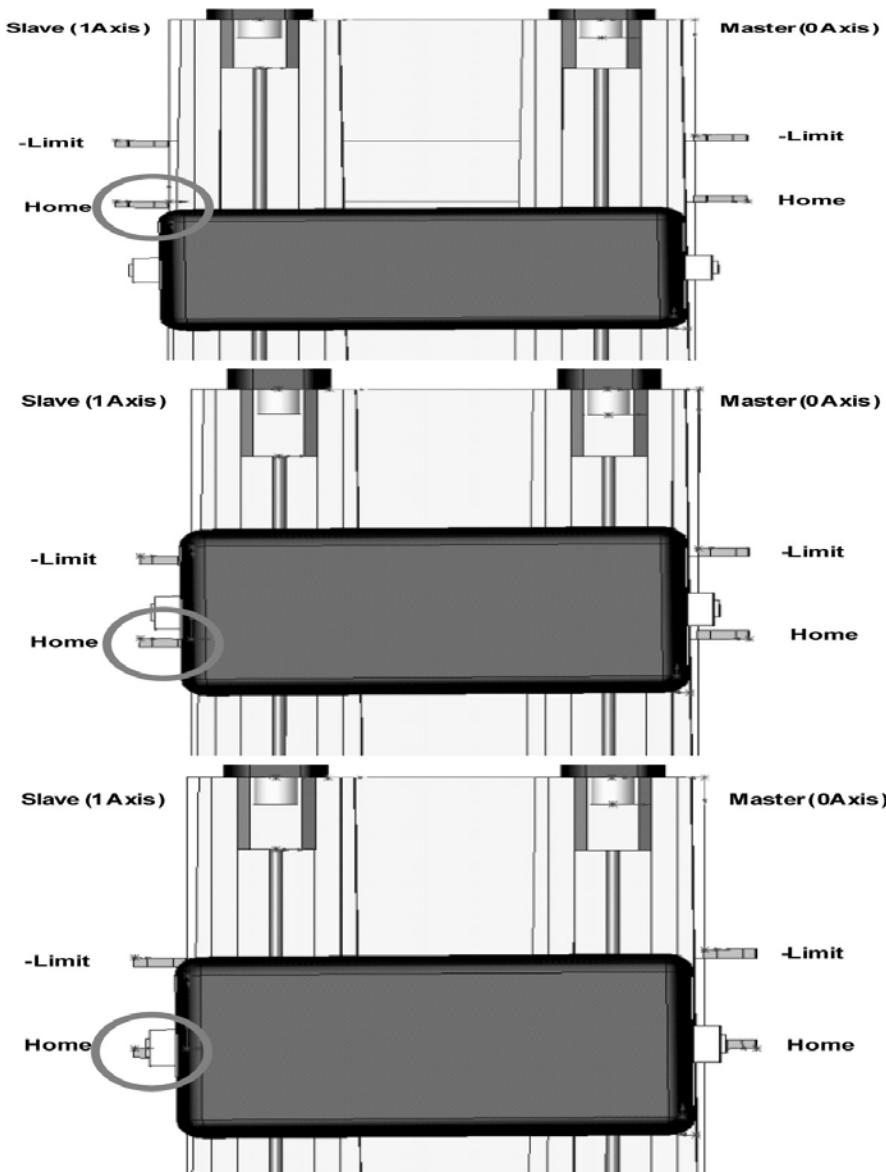
Carry out Up Edge search on Z phase towards HmDir direction, and emergency stop

1. Search home sensor on Master axis first. Make the position of 0 only for Master axis.

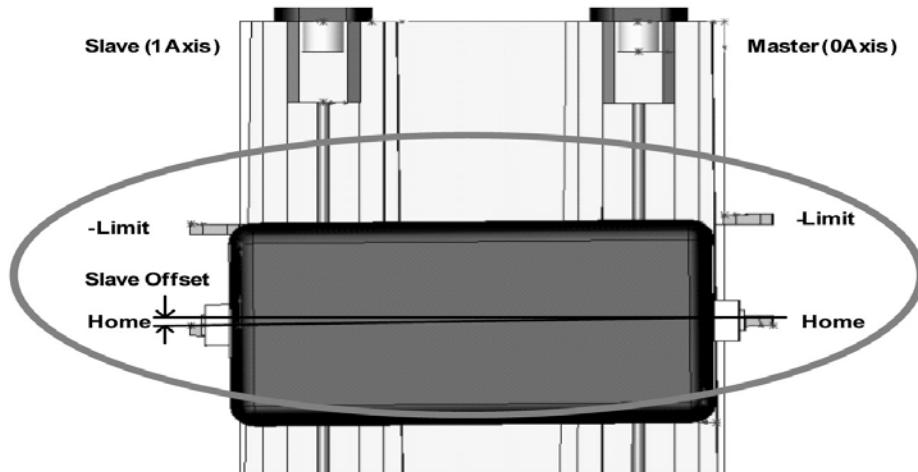


2. Search home sensor on Slave axis. Make the position of 0 only for slave axis. Then can obtain offset value of slave for master.

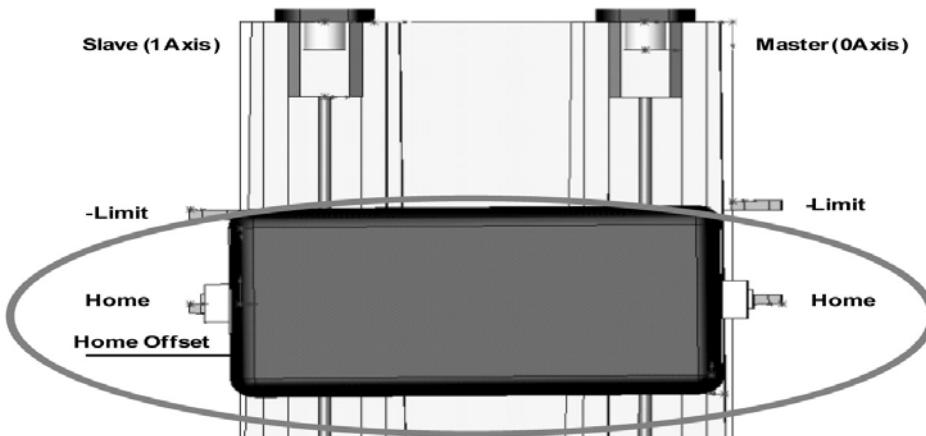




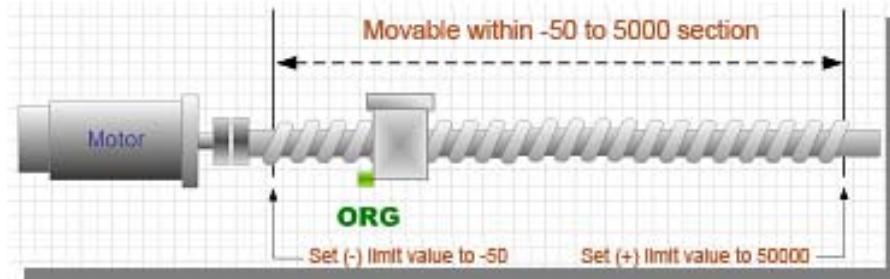
- Release link between Master and Slave, and move Slave axis as much as Slave offset. Then send Master axis to 0. And Link again.



- Move as much as home Offset, and make the position to be 0.
If there is no home Offset, do not move, and make the position to 0.



5. Carry out Software Limit setting and Gantry drive



- * Caution: Soft Limit can be set after completion of home search regularly. Soft Limit is used if Limit sensor is not installed mechanically or Limit needs to be changed by condition. Here, be careful that Soft Limit is set regularly when PLimit(+) value is bigger than NLimit(-). If NSwLimit value is equal to or bigger than PSwLimit, Soft Limit will not be set.

```
// Ex19_AXM_GantryDrive.cpp : Defines the entry point for the console application.
// Carry out Gantry drive.
```

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

#define AXIS_0 0
#define AXIS_1 1

void main(void)
{
    // Initialize library..
    // 7 means IRQ. IRQ is set automatically in PCI.
    DWORD Code = AxlOpen(7);
    if (Code == AXT_RT_SUCCESS)
    {
        printf("Library is initialized.\n");

        // Inspect whether motion module exists or not
        DWORD dwStatus;
        Code = AxmlInfoIsMotionModule (&dwStatus);
        if(Code == AXT_RT_SUCCESS)
        {
            if(dwStatus == STATUS_EXIST)
            {
                printf("Motion module exists.\n");

                for(int nAxisNo = 0; nAxisNo<2; nAxisNo++)
                {

```

```

// Set Active level of +End limit and -End limit on axis 0 to HIGH.
AxmSignalSetLimit(nAxisNo, 0, HIGH, HIGH);
// Set input level of inposition signal on axis 0 to HIGH.
AxmSignalSetInpos (nAxisNo, HIGH);
// Set input level of alarm signal on axis 0 to LOW.
AxmSignalSetServoAlarm (nAxisNo, LOW);
// Set Active input level of ESTOP signal on axis 0 to HIGH.
AxmSignalSetStop (nAxisNo, 0, HIGH);

// Make ordered value on axis 0 to be mm unit.
AxmMotSetMoveUnitPerPulse(nAxisNo, 10, 10000);
// Set initial velocity on axis 0 to 1. Default : 1
AxmMotSetMinVel(nAxisNo, 1);
// Make pulse output method on axis 0 to TwoCwCcwHigh.
AxmMotSetPulseOutMethod (nAxisNo, TwoCwCcwHigh);
// Set encoder input method on designated axis to 4 multiplication.
AxmMotSetEnclnputMethod (nAxisNo, ObverseSqr4Mode);

// Set home search method
AxmHomeSetMethod (nAxisNo,-1,HomeSensor,1,2000.0,0.0);
// Set velocity that will be used during home search
AxmHomeSetVel(nAxisNo, 100, 20, 10, 10, 200, 40);

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

//Set Gantry drive.
long lMasterAxisNo = AXIS_0;
long lSlaveAxisNo = AXIS_1;
BOOL bSIHomeUse = FALSE;
//TRUE : Search slave home sensor after master home sensor is searched
double dSIOffset = 0; //Offset between master and slave home sensors
double dSIOffsetRange = 10;
//Error limit between master and slave home sensor during home search
AxmGantrySetEnable (lMasterAxisNo, lSlaveAxisNo, bSIHomeUse, dSIOffset,
dSIOffsetRange);

// help Message
printf("[INFORMATION] *****\n");
printf("Gantry drive is set.\nStart drive pressing any key.\n");
printf("*****\n");

//Wait until pressed any key
getch();

printf("\nCarry out home search of Gantry drive.\n");
// Set home search method
AxmHomeSetMethod (lMasterAxisNo,-1,HomeSensor,1,2000.0,0.0);
// Set velocity that will be used during home search
AxmHomeSetVel(lMasterAxisNo, 100, 20, 10, 10, 200, 40);

```

```

//Carry out home search.
AxmHomeSetStart (IMasterAxisNo);

// Verify current home search progress rate
DWORD uHomeMainStepNumber , uHomeStepNumber_0;
AxmHomeGetRate(IMasterAxisNo, &uHomeMainStepNumber ,
    &uHomeStepNumber_0);
while(uHomeStepNumber_0 < 100)
{
    AxmHomeGetRate(IMasterAxisNo, &uHomeMainStepNumber ,
        &uHomeStepNumber_0);
    printf("Wr In home search... %d ", uHomeStepNumber_0);
}
printf("WrHome search is completed \n");

printf("Drive Master axis as much as 100.\n");
AxmMoveStartPos (IMasterAxisNo, 100, 100, 200, 200);
//Verify if in drive.
DWORD uStatus;
AxmStatusReadInMotion (IMasterAxisNo, &uStatus);
while(uStatus) {
    AxmStatusReadInMotion (IMasterAxisNo, &uStatus);
}

//Exit Gantry drive.
AxmGantrySetDisable (IMasterAxisNo,ISlaveAxisNo);

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

}

else
printf ("AxmlInfoIsMotionModule () : ERROR ( NOT STATUS_EXIST )
        code 0x%x\n",Code);
}

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

else
printf ("AxIOpen() : ERROR     code 0x%x\n",Code);

// Exit library.
if (AxIClose())
    printf("Library is exited.\n");
else
    printf("Library is not exited normally.\n");
}

```

Virtual Axis Mapping

Virtual axis mapping is an API that gives command using arbitrary axis number, not an actual corresponding axis number, when drive command is given from library. For example, when give a command on number axis 0, give a command as number 2 axis during command. Let us suppose there are axes from number 0 to 10 in the system, and number 3 and 4 axis in the middle are removed somehow. In this case, axis number will be run moved up from number 0 to 8 in sequence. Therefore, it is occurred that axis numbers on the program have to be changed one by one, and in this case, it can be solved by mapping with original axis number without toughing program source. In this case, number 3 to 8 axes just need to change to number 5 to 10 axes.

To use this virtual axis mapping, use [AxmVirtualSetAxisNoMap](#) API mapping one axis to virtual axis or [AxmVirtualSetMultiAxisNoMap](#) API mapping various axes to virtual axes. One thing to be careful is that command on previous axis number does not operate regularly because previous actual axis number does not exist after mapping to virtual axis, and newly setting axis number shall be used after setting.

```
//Carry out mapping axis 0 to number 2 axis.  
long lRealAxisNo = 0;  
long lVirtualAxisNo = 2;  
AxmVirtualSetAxisNoMap (lRealAxisNo, lVirtualAxisNo);
```

```
//Carry out mapping 0 and 1 axis to number 2 and 3.  
long lSize = 2;  
long lpRealAxesNo[2] = {0,1};  
long lpVirtualAxesNo[2] = {2,3};  
AxmVirtualSetMultiAxisNoMap (lSize, lpRealAxesNo, lpVirtualAxesNo);
```

To return mapped axis number to original, use [AxmVirtualResetAxisMap](#) API.

```
//Clear virtual axis mapping.  
AxmVirtualResetAxisMap ();
```

```

// Ex20_AXM_AxisNoMapping.cpp : Defines the entry point for the console application.
// Carry out virtual axis mapping.

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

#define AXIS_0 0
#define AXIS_2 2

void main(void)
{
    // Initialize library..
    // 7 means IRQ. IRQ is set automatically in PCI.
    DWORD Code = AxIOpen(7);
    if (Code == AXT_RT_SUCCESS)
    {
        printf("Library is initialized.\n");

        // Inspect whether motion module exists or not
        DWORD dwStatus;
        Code = AxmInfoIsMotionModule (&dwStatus);
        if(Code == AXT_RT_SUCCESS)
        {
            if(dwStatus == STATUS_EXIST)
            {
                printf("Motion module exists.\n");

                // Set Active level of +End limit and -End limit on axis 0 to HIGH.
                AxmSignalSetLimit(nAxisNo, 0, HIGH, HIGH);
                // Set input level of inposition signal on axis 0 to HIGH.
                AxmSignalSetInpos (nAxisNo, HIGH);
                // Set input level of alarm signal on axis 0 to LOW.
                AxmSignalSetServoAlarm (nAxisNo, LOW);
                // Set Active input level of ESTOP signal on axis 0 to HIGH.
                AxmSignalSetStop (nAxisNo, 0, HIGH);

                // Make ordered value on axis 0 to be mm unit.
                AxmMotSetMoveUnitPerPulse(nAxisNo, 10, 10000);
                // Set initial velocity on axis 0 to 1. Default : 1
                AxmMotSetMinVel(nAxisNo, 1);
                // Make pulse output method on axis 0 to TwoCwCcwHigh.
                AxmMotSetPulseOutMethod (nAxisNo, TwoCcwCwHigh);
                // Set encoder input method on designated axis to 4 multiplication.
                AxmMotSetEnclnputMethod (AXIS_0, ObverseSqr4Mode);

                //Use relative S-curve velocity profile
                DWORD uAbsRelMode = 1;
                DWORD uProfileMode = 3;
                AxmMotSetAbsRelMode (AXIS_0, uAbsRelMode); // Set to relative position drive
            }
        }
    }
}

```

```

AxmMotSetProfileMode (AXIS_0, uProfileMode); // S-curve velocity

//Servo On
AxmSignalServoOn(AXIS_0, ENABLE);
//Carry out mapping axis 0 to number 2 axis.
long lRealAxisNo = AXIS_0;
long lVirtualAxisNo = AXIS_2;
AxmVirtualSetAxisNoMap (lRealAxisNo, lVirtualAxisNo);

// help Message
printf("[INFORMATION]*****\n");
printf("Number axis 0 is set to number 2.\nNumber 2 axis drive starts pressing
any key.\n");
printf("*****\n");

//Wait until pressed any key
getch();

//Drive axis 0 as much as 100.
(AXIS_2, 100, 100, 200, 200);
//Verify if in drive.
DWORD uStatus;
AxmStatusReadInMotion (AXIS_2, &uStatus);
while(uStatus) {
    AxmStatusReadInMotion (AXIS_2, &uStatus);
}

//Clear virtual axis mapping.
AxmVirtualResetAxisMap();

//Servo Off
AxmSignalServoOn(AXIS_0, DISABLE);
}

else
printf ("AxmlInfoIsMotionModule () : ERROR ( NOT STATUS_EXIST )
        code 0x%x\n",Code);
}

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

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

// Exit library.
if (AxlClose())
    printf("Library is exited.\n");
else
    printf("Library is not exited normally.\n");
}

```

Interrupt Setting

- [1\) Interrupt Management Method Setting](#)
- [2\) Setting of Interrupt Use or Not](#)

Interrupt can be generated using various events as interrupt source generating in motion drive in AXM, if register of interrupt BAMK composed of 31 bits per axis generates each interrupt, it will be 1. Main interrupt sources are following.

* Mainly used CAMC-IP interrupt source Status

Define	Description
IPINTBANK1_DRIVE_END	When drive exits
IPINTBANK1_DOWN	Deceleration start point
IPINTBANK1_CONT	Constant velocity start point
IPINTBANK1_UP	Acceleration start point
IPINTBANK1_SIGNALDETECTED	If signal assigned to MODE 1 is detected
IPINTBANK1_SLM	If NSLM / PSLM signal is activated
IPINTBANK1_ELM	If NELM / PELM signal is activated
IPINTBANK2_ALARM_ERROR	If alarm signal is activated
IPINTBANK2_INPOSITION	If inposition signal is activated
IPINTBANK2_SSTOP_SIGNAL	If SSTOP signal is inputted
IPINTBANK2_ESTOP_SIGNAL	If ESTOP signal is inputted
IPINTBANK2_L_C_INP_Q_LESS_4	If straight line / circular interpolation data queue is to be less than 4

* Mainly used CAMC-QI interrupt source Status

Define	Description
QIINTBANK1_1	When drive exits
QIINTBANK1_2	When drive starts
QIINTBANK1_4	counter #1 = comparator #1 event occurrence
QIINTBANK2_31	If emergency stop signal is inputted
QIINTBANK2_30	If NSLM / PSLM signal is activated
QIINTBANK2_29	If NELM / PELM signal is activated
QIINTBANK1_29	If alarm signal is activated
QIINTBANK2_9	Drive and exit condition occurrence using inposition function
IPINTBANK2_SSTOP_SIGNAL	If SSTOP signal is inputted
IPINTBANK2_ESTOP_SIGNAL	If ESTOP signal is inputted
IPINTBANK2_L_C_INP_Q_LESS_4	If straight line / circular interpolation data queue is to be less than 4

* In case of CAMC – IP
AXT_MOTION_IPINTERRUPT_BANK1_DEF

#define Bank (0)	Value	Explanation
IPINTBANK1_DRIVE_END	0	Bit 0, Drive end(default value : 1).
IPINTBANK1_ICG	1	Bit 1, INCNT is greater than INCNTCMP.
IPINTBANK1_ICE	2	Bit 2, INCNT is equal with INCNTCMP.
IPINTBANK1_ICL	3	Bit 3, INCNT is less than INCNTCMP.
IPINTBANK1_ECG	4	Bit 4, EXCNT is greater than EXCNTCMP.
IPINTBANK1_ECE	5	Bit 5, EXCNT is equal with EXCNTCMP.
IPINTBANK1_ECL	6	Bit 6, EXCNT is less than EXCNTCMP.
IPINTBANK1_SCRQEMPTY	7	Bit 7, Script control queue is empty.
IPINTBANK1_CAPRQEMPTY	8	Bit 8, Caption result data queue is empty.
IPINTBANK1_SCRREG1EXE	9	Bit 9, Script control register–1 command is executed.
IPINTBANK1_SCRREG2EXE	10	Bit 10, Script control register–2 command is executed.
IPINTBANK1_SCRREG3EXE	11	Bit 11, Script control register–3 command is executed.
IPINTBANK1_CAPREG1EXE	12	Bit 12, Caption control register–1 command is executed.
IPINTBANK1_CAPREG2EXE	13	Bit 13, Caption control register–2 command is executed.
IPINTBANK1_CAPREG3EXE	14	Bit 14, Caption control register–3 command is executed.
IPINTBANK1_INTGGENCMD	15	Bit 15, Interrupt generation command is executed(0xFF)
IPINTBANK1_DOWN	16	Bit 16, At starting point for deceleration drive.
IPINTBANK1_CONT	17	Bit 17, At starting point for constant speed drive.
IPINTBANK1_UP	18	Bit 18, At starting point for acceleration drive.
IPINTBANK1_SIGNALDETECTED	19	Bit 19, Signal assigned in MODE1 is detected.
IPINTBANK1_SP23E	20	Bit 20, Current speed is equal with rate change point RCP23.
IPINTBANK1_SP12E	21	Bit 21, Current speed is equal with rate change point RCP12.
IPINTBANK1_SPE	22	Bit 22, Current speed is equal with speed

		comparison data(SPDCMP).
IPINTBANK1_INCEICM	23	Bit 23, INTCNT(1'st counter) is equal with ICM(1'st count minus limit data)
IPINTBANK1_SCRQEXE	24	Bit 24, Script queue command is executed When SCRCONQ's 30 bit is '1'.
IPINTBANK1_CAPQEXE	25	Bit 25, Caption queue command is executed When CAPCONQ's 30 bit is '1'.
IPINTBANK1_SLM	26	Bit 26, NSLM/PSLM input signal is activated.
IPINTBANK1_ELM	27	Bit 27, NELM/PELM input signal is activated.
IPINTBANK1_USERDEFINE1	28	Bit 28, Selectable interrupt source 0(refer "0xFE" command).
IPINTBANK1_USERDEFINE2	29	Bit 29, Selectable interrupt source 1(refer "0xFE" command).
IPINTBANK1_USERDEFINE3	30	Bit 30, Selectable interrupt source 2(refer "0xFE" command).
IPINTBANK1_USERDEFINE4	31	Bit 31, Selectable interrupt source 3(refer "0xFE" command).

AXT_MOTION_IPINTERRUPT_BANK2_DEF

#define	IBank (1)	Value	Explanation
IPINTBANK2_L_C_INP_Q_EMPTY	0	Bit 0, Linear/Circular interpolation parameter queue is empty.	
IPINTBANK2_P_INP_Q_EMPTY	1	Bit 1, Bit pattern interpolation queue is empty.	
IPINTBANK2_ALARM_ERROR	2	Bit 2, Alarm input signal is activated.	
IPINTBANK2_INPOSITION	3	Bit 3, Inposition input signal is activated.	
IPINTBANK2_MARK_SIGNAL_HIGH	4	Bit 4, Mark input signal is activated.	
IPINTBANK2_SSTOP_SIGNAL	5	Bit 5, SSTOP input signal is activated.	
IPINTBANK2_ESTOP_SIGNAL	6	Bit 6, ESTOP input signal is activated.	
IPINTBANK2_SYNC_ACTIVATED	7	Bit 7, SYNC input signal is activated.	
IPINTBANK2_TRIGGER_ENABLE	8	Bit 8, Trigger output is activated.	
IPINTBANK2_EXCNTCLR	9	Bit 9, External(2'nd) counter is cleared by EXCNTCLR setting.	
IPINTBANK2_FSTCOMPARE_RESULT BIT0	10	Bit 10, ALU1's compare result bit 0 is activated.	
IPINTBANK2_FSTCOMPARE_RESULT	11	Bit 11, ALU1's compare result bit 1 is	

BIT1		activated.
IPINTBANK2_FSTCOMPARE_RESULT_BIT2	12	Bit 12, ALU1's compare result bit 2 is activated.
IPINTBANK2_FSTCOMPARE_RESULT_BIT3	13	Bit 13, ALU1's compare result bit 3 is activated.
IPINTBANK2_FSTCOMPARE_RESULT_BIT4	14	Bit 14, ALU1's compare result bit 4 is activated.
IPINTBANK2_SNDCOMPARE_RESULT_BIT0	15	Bit 15, ALU2's compare result bit 0 is activated.
IPINTBANK2_SNDCOMPARE_RESULT_BIT1	16	Bit 16, ALU2's compare result bit 1 is activated.
IPINTBANK2_SNDCOMPARE_RESULT_BIT2	17	Bit 17, ALU2's compare result bit 2 is activated.
IPINTBANK2_SNDCOMPARE_RESULT_BIT3	18	Bit 18, ALU2's compare result bit 3 is activated.
IPINTBANK2_SNDCOMPARE_RESULT_BIT4	19	Bit 19, ALU2's compare result bit 4 is activated.
IPINTBANK2_L_C_INP_Q_LESS_4	20	Bit 20, Linear/Circular interpolation parameter queue is less than 4.
IPINTBANK2_P_INP_Q_LESS_4	21	Bit 21, Pattern interpolation parameter queue is less than 4
IPINTBANK2_XSYNC_ACTIVATED	22	Bit 22, X axis sync input signal is activated.
IPINTBANK2_YSYNC_ACTIVATED	23	Bit 23, Y axis sync input signal is activated.
IPINTBANK2_P_INP_END_BY_END_PATTERN	24	Bit 24, Bit pattern interpolation is terminated by end pattern.

* In case of CAMC - QI

AXT_MOTION_QI_INTERRUPT_BANK1_DEF

#define	IBank (0)	Value	Explanation
QIINTBANK1_1	1		Bit 1, when drive exits
QIINTBANK1_2	2		Bit 2, when drive starts
QIINTBANK1_3	3		Bit 3, counter #1 < Comparator #1 event occurrence
QIINTBANK1_4	4		Bit 4, counter #1 = Comparator #1 event occurrence
QIINTBANK1_5	5		Bit 5, counter #1 > Comparator #1 event occurrence
QIINTBANK1_6	6		Bit 6, counter #2 < Comparator #2 event occurrence
QIINTBANK1_7	7		Bit 7, counter #2 = Comparator #2 event occurrence

QIINTBANK1_8	8	Bit 8, counter #2 > Comparator #2 event occurrence
QIINTBANK1_9	9	Bit 9, counter #3 < Comparator #3 event occurrence
QIINTBANK1_10	10	Bit 10, counter #3 = Comparator #3 event occurrence
QIINTBANK1_11	11	Bit 11, counter #3 > Comparator #3 event occurrence
QIINTBANK1_12	12	Bit 12, counter #4 < Comparator #4 event occurrence
QIINTBANK1_13	13	Bit 13, counter #4 = Comparator #4 event occurrence
QIINTBANK1_14	14	Bit 14, counter #4 < Comparator #4 event occurrence
QIINTBANK1_15	15	Bit 15, counter #5 < Comparator #5 event occurrence
QIINTBANK1_16	16	Bit 16, counter #5 = Comparator #5 event occurrence
QIINTBANK1_17	17	Bit 17, counter #5 > Comparator #5 event occurrence
QIINTBANK1_18	18	Bit 18, timer #1 event occurrence
QIINTBANK1_19	19	Bit 19, timer #2 event occurrence.
QIINTBANK1_20	20	Bit 20, drive reservation setting Queue cleared.
QIINTBANK1_21	21	Bit 21, drive reservation setting Queue filled
QIINTBANK1_22	22	Bit 22, trigger generation distance cycle/absolute position Queue cleared.
QIINTBANK1_23	23	Bit 23, trigger generation distance cycle / absolute position Queue filled
QIINTBANK1_24	24	Bit 24, trigger signal event occurrence
QIINTBANK1_25	25	Bit 25, script #1 command reservation setting Queue cleared.
QIINTBANK1_26	26	Bit 26, script #2 command reservation setting Queue cleared.
QIINTBANK1_27	27	Bit 27, script #3 command reservation setting Initialized with register operation.
QIINTBANK1_28	28	Bit 28, script #4 command reservation setting Initialized with register operation.
QIINTBANK1_29	29	Bit 29, servo alarm signal permitted
QIINTBANK1_30	30	Bit 30, $ CNT1 - CNT2 \geq CNT4 $ event occurrence.
QIINTBANK1_31	31	Bit 31, interrupt generation command INTGEN operation.

AXT_MOTION_QIINTERRUPT_BANK2_DEF

#define	IBank (1)	Value	Explanation
QIINTBANK2_0	0	Bit 0, script #1 reading command result Queue filled.	
QIINTBANK2_1	1	Bit 1, script #2 reading command result Queue filled	
QIINTBANK2_2	2	Bit 2, script #3 reading command result Register renewed with	

		new data
QIINTBANK2_3	3	Bit 3, script #4 reading command result Register renewed with new data
QIINTBANK2_4	4	Bit 4, during operation among reserved command script #1 set command operated by interrupt generation
QIINTBANK2_5	5	Bit 5, during operation among reserved command script #2 set command operated by interrupt generation.
QIINTBANK2_6	6	Bit 6, during operation of reserved command script #3 set command operated by interrupt generation.
QIINTBANK2_7	7	Bit 7, during operation of reserved command script #4 set command operated by interrupt generation.
QIINTBANK2_8	8	Bit 8, Drive starts
QIINTBANK2_9	9	Bit 9, drive and exit condition occurrence using servo position decision completion (Inposition) function.
QIINTBANK2_10	10	Bit 10, occurrence of event selection #1 condition to use during event counter operation.
QIINTBANK2_11	11	Bit 11, occurrence of event selection #2 condition to use during event counter operation.
QIINTBANK2_12	12	Bit 12, SQSTR1 signal permitted.
QIINTBANK2_13	13	Bit 13, SQSTR2 signal permitted.
QIINTBANK2_14	14	Bit 14, UIO0 terminal signal changed to '1'.
QIINTBANK2_15	15	Bit 15, UIO1 terminal signal changed to '1'.
QIINTBANK2_16	16	Bit 16, UIO2 terminal signal changed to '1'.
QIINTBANK2_17	17	Bit 17, UIO3 terminal signal changed to '1'.
QIINTBANK2_18	18	Bit 18, UIO4 terminal signal changed to '1'.
QIINTBANK2_19	19	Bit 19, UIO5 terminal signal changed to '1'.
QIINTBANK2_20	20	Bit 20, UIO6 terminal signal changed to '1'.
QIINTBANK2_21	21	Bit 21, UIO7 terminal signal changed to '1'.
QIINTBANK2_22	22	Bit 22, UIO8 terminal signal changed to '1'.
QIINTBANK2_23	23	Bit 23, UIO9 terminal signal changed to '1'.
QIINTBANK2_24	24	Bit 24, UIO10 terminal signal changed to '1'.
QIINTBANK2_25	25	Bit 25, UIO11 terminal signal changed to '1'.
QIINTBANK2_26	26	Bit 26, error stop condition (LMT, ESTOP, STOP, ESTOP, CMD, ALARM) occurrence.
QIINTBANK2_27	27	Bit 27, data setting error occurrence in interpolation.
QIINTBANK2_28	28	Bit 28, Don't Care

QIINTBANK2_29	29	Bit 29, limit signal (PELM, NELM) inputted.
QIINTBANK2_30	30	Bit 30, additional limit signal (PSLM, NSLM) inputted.
QIINTBANK2_31	31	Bit 31, emergency stop signal (ESTOP) inputted.

To use interrupt, first set how to manage interrupt on corresponding axis using [AxmlInterruptSetAxis](#) API, and set use of interrupt to ENABLE using [AxmlInterruptSetAxisEnable](#) API.

1) Interrupt Management Method Setting

When an event occurs, to manage this, AXM provides 3 methods; callback API, window message, and event method. Callback API method has an advantage to be reported event first of all by calling callback API immediately at the point of event occurrence, but main process is tied up until callback API exits completely. In short, if there is a work that loads in callback API, need to be careful to use. However, since event method is operated separately with main process using a specific thread watching event occurrence, this method is especially recommended because source can be used most efficiently including MultiProcessor system. To set event management method, use [AxmlInterruptSetAxis](#) API and set as following.

● *Callback API method*

To use callback API method, set as following.

```
//Set event management method on number axis 0 to callback API method.
AxmlInterruptSetAxis (0, NULL, NULL, OnMotionInterruptCallback, NULL);
```

OnMotionInterruptCallback is a pointer of callback API to manage interrupt. Callback API is an all area API and it is set as following. When interrupt is occurred, it calls callback API to manage interrupt automatically.

```
//Callback API to manage interrupt
void __stdcall OnMotionInterruptCallback(long lAxisNo, DWORD dwFlag)
{
    //lAxisNo : Axis number that interrupt is occurred
    //dwFlag : Flag that interrupt is occurred
    printf("[INTERRUPT MESSAGE from CALLBACK method] lAxisNo: %d, dwFlag: %d \n",
    lActiveNo,bwFlag);
}
```

● *Window Message Method*

To use window message method, set as following. Since window message method cannot use in C, for this part, use VC++ to describe. To use window message method, input window handle to receive a message and window message. If it does not need to use window message or needs to use default value, input 0.

```
// Set event management method on number axis 0 to window event method.
AxmlInterruptSetAxis (0, this->m_hWnd, WM_MOTION_INTERRUPT, NULL, NULL);
```

Instruction shows below, OnMotionInterruptMessage API manage interrupt.

```
#define WM_MOTION_INTERRUPT 1011    //Message definition to receive interrupt message
#define AXIS_ 0 0                    //Drive axis number
```

```

...(Skip)...

//Connect message to API
BEGIN_MESSAGE_MAP(CExMotionInterruptDlg, CDialog)
//{{AFX_MSG_MAP(CExMotionInterruptDlg)
ON_MESSAGE(WM_MOTION_INTERRUPT,OnMotionInterruptMessage)

//}}AFX_MSG_MAP
END_MESSAGE_MAP()

...(Skip)...

void CExMotionInterruptDlg::OnBtnStartInterruptMessage()
{
CString Message;
//Set interrupt management method
AxmInterruptSetAxis (AXIS_0, this->m_hWnd, WM_MOTION_INTERRUPT, NULL, NULL);

//Set if interrupt permitted for specific channel
AxmInterruptSetAxisEnable (AXIS_0,ENABLE);
}

void CExMotionInterruptDlg::OnMotionInterruptMessage(long lAxisNo, DWORD dwFlag)
{
//lAxisNo : Axis number that interrupt is occurred
//dwFlag : Flag that interrupt is occurred
printf("[INTERRUPT MESSAGE from CALLBACK method] lAxisNo : %d, bwFlag : %d \n",
lAxisNo, dwFlag);
}

```

● Event Method

To use event method, set as following. Since event method is operated separately with main process using a specific thread watching event occurrence, this method is especially recommended because source can be used most efficiently including MultiProcessor system.

This part also use VC++ to describe. To use event method, HANDLE type member variable m_hInterruptEvent to receive event ID is declared in advance and used.

```

//Set event management method on number 0 module to window event method.
AxmInterruptSetAxis (0,NULL,NULL,NULL,&m_hInterruptEvent);

```

Instruction shows bellow, and ThreadProc API has to manage interrupt. One thing to be careful is [AxmInterruptSetAxis](#) API must be called before thread operation.

If interrupt is occurred, interrupt event will be occurred and thread will detect this event and know that interrupt is occurred. And it verifies on which axis and of which interrupt is occurred verifying axis number that interrupt is occurred and interrupt flag information with calling [AxmInterruptRead](#) API inside thread.

Header File
//Required variable to use event method interrupt

```
HANDLE m_hThreadHandle;
BOOL m_bThread;
HANDLE m_hInterruptEvent;
```

Source File

```
void CExMOTIONInterruptDlg::OnBtnStartInterruptEvent()
{
CString Message;

//Set interrupt management method
AxmInterruptSetAxis_(0,NULL,NULL,NULL,&m_hInterruptEvent); //Event method

//To use event method, initialize after declaration of HANDLE pEvent member variable
//Make thread.
m_bThread = TRUE;
CreateThread(NULL, 0, (LPTHREAD_START_ROUTINE)ThreadProc, this, NULL, NULL);

//Set interrupt permission on axis 0
AxmInterruptSetAxisEnable_ (CHANNEL_NO,ENABLE);
}

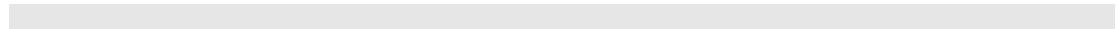
void ThreadProc(LPVOID lpData)
{
CExMOTIONInterruptDlg *pDlg = (CExMOTIONInterruptDlg *)lpData;
while(pDlg->m_bThread)
{
if (WaitForSingleObject(pDlg->m_hInterruptEvent, INFINITE) == WAIT_OBJECT_0)
{
long lAxisNo; // Axis number that interrupt is occurred
DWORD dwFlag; // Flag that interrupt is occurred
AxmInterruptRead_ (lAxisNo, dwFlag);

printf("[INTERRUPT MESSAGE from CALLBACK method] lAxisNo : %d, dwFlag : %d \n",
lAxisNo, dwFlag);
}
}
}
```

2) Interrupt Use Setting

Use of interrupt is set to DISABLE as default. Therefore, interrupt will occur by setting on interrupt mask after setting of ENABLE using [AxmInterruptSetAxisEnable_](#) API.

```
//Set interrupt to ENABLE on number axis 0
long lAxisNo = 0;
DWORD uUse = ENABLE;
AxmInterruptSetAxisEnable_ (lAxisNo, uUse);
```



```
// Ex21_AXM_Interrupt.cpp : Defines the entry point for the console application.  
// Generate interrupt and verify current state.  
  
#include "stdafx.h"  
#include "AXL.h"  
#include <conio.h>  
#include "stdio.h"  
#define AXIS_0 0  
  
//Callback API to manage interrupt  
void __stdcall OnMotionInterruptCallback(long lAxisNo, DWORD dwFlag)  
{  
    //lAxisNo : Axis number that interrupt is occurred  
    //dwFlag : Flag that interrupt is occurred  
    printf("[INTERRUPT MESSAGE from CALLBACK method] lAxisNo : %d, dwFlag : %dx \n",  
        lAxisNo,dwFlag);  
}  
  
void main(void)  
{  
    // Initialize library..  
    // 7 means IRQ. IRQ is set automatically in PCI.  
    DWORD Code = AxlOpen(7);  
    if (Code == AXT_RT_SUCCESS)  
    {  
        printf("Library is initialized.\n");  
  
        // Inspect whether motion module exists or not  
        DWORD dwStatus;  
        Code = AxmInfoIsMotionModule (&dwStatus);  
        if(Code == AXT_RT_SUCCESS)  
        {  
            if(dwStatus == STATUS_EXIST)  
            {  
                printf("Motion module exists.\n");  
  
                // Set Active level of +End limit and -End limit on axis 0 to HIGH.  
                AxmSignalSetLimit(nAxisNo, 0, HIGH, HIGH);  
                // Set input level of inposition signal on axis 0 to HIGH.  
                AxmSignalSetInpos (nAxisNo, HIGH);  
                // Set input level of alarm signal on axis 0 to LOW.  
                AxmSignalSetServoAlarm (nAxisNo, LOW);  
                // Set Active input level of ESTOP signal on axis 0 to HIGH.  
                AxmSignalSetStop (nAxisNo, 0, HIGH);  
  
                // Make ordered value on axis 0 to be mm unit.  
                AxmMotSetMoveUnitPerPulse(nAxisNo, 10, 10000);  
                // Set initial velocity on axis 0 to 1. Default : 1  
                AxmMotSetMinVel(nAxisNo, 1);  
                // Make pulse output method on axis 0 to TwoCwCcwHigh.  
                AxmMotSetPulseOutMethod (nAxisNo, TwoCcwCcwHigh);  
                // Set encoder input method on designated axis to 4 multiplication.
```

```

AxmMotSetEnclInputMethod (AXIS_0, ObverseSqr4Mode);

//Use relative position S-curve velocity profile
DWORD uAbsRelMode = 1;
DWORD uProfileMode = 3;
AxmMotSetAbsRelMode (AXIS_0, uAbsRelMode);
AxmMotSetProfileMode (AXIS_0, uProfileMode);

//Servo On
AxmSignalServoOn(AXIS_0, ENABLE);

long lBank = 1; //CAMC-QI일 경우
//Generate interrupt if motion exits
AxmlInterruptSetUserEnable(AXIS_0, lBank, QIINTBANK2_9);

//Set interrupt to ENABLE on number axis 0
AxmlInterruptSetAxisEnable (AXIS_0, ENABLE);

//Set event management method to callback API method on axis 0.
AxmlInterruptSetAxis (AXIS_0, NULL, NULL, OnMotionInterruptCallback,NULL);

// help Message
printf("[INFORMATION] ****\n");
printf("Generate interrupt starting drive if pressing any key.\n");
printf("*****\n");

//Wait until any key is pressed
getch();

//Drive axis 0 as much as 100.
AxmMoveStartPos (AXIS_0, 100, 100, 200, 200);
//Verify if in drive.
DWORD uStatus;
AxmStatusReadInMotion (AXIS_0, &uStatus);
while(uStatus) {
    AxmStatusReadInMotion (AXIS_0, &uStatus);
}

getch();

//Servo Off
AxmSignalServoOn(AXIS_0, DISABLE);

}

else
printf ("AxmlInfoIsMotionModule () : ERROR ( NOT STATUS_EXIST )
        code 0x%x\n",Code);
}

else
printf ("AxmlInfoIsMotionModule () : ERROR ( Return FALSE )   code 0x%x\n",Code);

}

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

```

```
// Exit library.  
if (AxIClose())  
    printf("Library is exited.\r\n");  
else  
    printf("Library is not exited normally.\r\n");  
}
```

Use of Trigger Signal

AXM may generate interrupt as well as 5V level of Trigger signal based on specific condition during in drive. Trigger signal comes out from TRIG signal pin, and if it is set to ActiveHIGH, it maintains 0V normally and if a signal is occurred, it changed to 5V for the set time then comes back to 0V, and vice versa for ActiveLow. Mechanical part, of course, is to be 24V. To generate trigger signal, first, set trigger signal level and duration using [AxmTriggerSetTimeLevel](#) API, and call APIs generating trigger. There are three trigger generation APIs: [AxmTriggerSetAbsPeriod](#) / [AxmTriggerOnlyAbs](#) to generate trigger on a specific condition, [AxmTriggerSetBlock](#) to generate trigger with constant interval for a specific section, and [AxmTriggerOneShot](#) / [AxmTriggerSetTimerOneshot](#) to generate trigger at once on an arbitrary point.

● [AxmTriggerSetTimeLevel](#)

When generate trigger signal, this API is of trigger output signal time setting, use of trigger output signal and Active Level, trigger datum point value input source, and interrupt output during trigger output. Trigger output signal time can be set to minimum (1 = 1usec) to maximum (40000 = 4msec), and use of trigger output signal and Active Level can be set to LOW(0), HIGH(1), UNUSED(2), and USED(3). Trigger datum position value input source is available to use Command Position(0) and Actual Position(1), and interrupt output can be set to DISABLE(0) or ENABLE(1) during trigger output.

```
//Set trigger signal level and duration on axis 0.
long lAxisNo = 0;
double dTrigTime = 100; // 0.1ms
DWORD uTriggerLevel = HIGH;
DWORD uSelect = 0; // Trigger occurrence in the basis of Command Position
DWORD ulInterrupt = DISABLE;

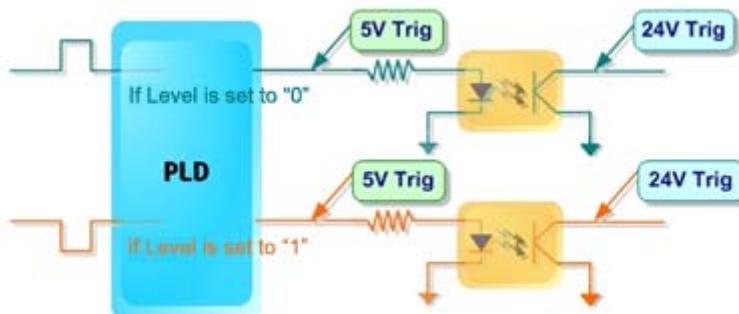
AxmTriggerSetTimeLevel (lAxisNo, dTrigTime, uTriggerLevel, uSelect, ulInterrupt);
```

Low(B contact) : normally Close → if detected, Open

High(A contact): normally Open → if detected, Close

Falling Edge : maintain High state normally, fall to Low at trigger point and rise again to High.

Rising Edge : maintain Low state normally, rise to High at trigger point and fall again to Low.



* Notice : In case of 5V trigger output, if Active Level is set to 0, it will be 5V when trigger is generated at 0V normally, and if case of 24V trigger, if trigger is outputted in Open state, it will be N24V. therefore, if trigger is used with 24V through terminal, it needs to be measured with 2kΩ of pull-up resistance to verify

trigger output.

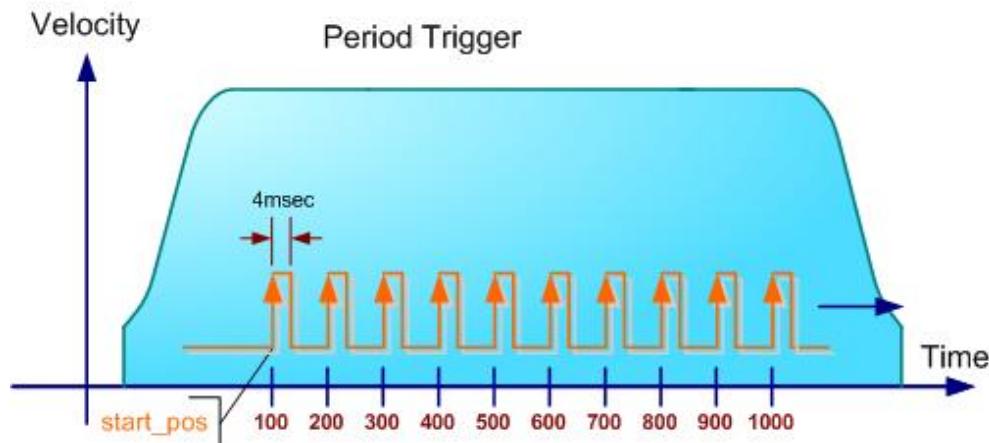
● [AxmTriggerSetAbsPeriod](#)

To generate trigger at a specific position, [AxmTriggerSetAbsPeriod](#) API can be used. This API generates trigger receiving trigger output method and position, and if uMethod value is to be 0, it sets trigger position value to 1000 with periodic trigger method using input trigger position value and when it moves, trigger is generated at every 1000 position (1000,2000,3000,...). And if uMethod value is to be 1, trigger is generated only at 1000 position with absolute trigger generation method and trigger is not generated at 2000, 3000, and so on.

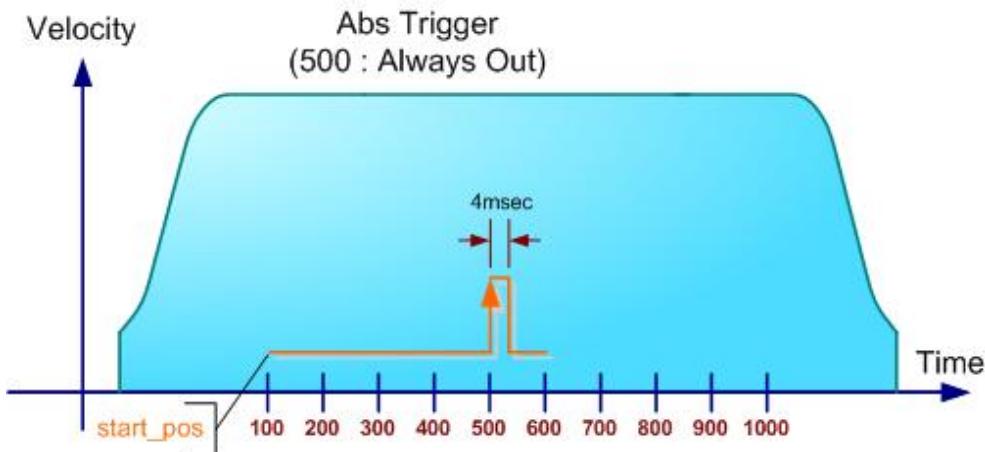
```
//Set to generate trigger at every 1000 position during moving on axis 0.
long lAxisNo = 0;
DWORD uMethod = 0; // [0] : Periodic trigger method, [1] : Absolute position trigger method
double dPosition = 1000;
AxmTriggerSetAbsPeriod (lAxisNo, uMethod, dPosition);
```

Note: When trigger mode is set to periodic mode, trigger at the time of setting is outputted. From the position started differently with [AxmTriggerSetBlock](#) API, trigger is outputted on the basis of counter change volume.

Period mode : If Position is 100, trigger output is generated at every 100 pulse.



Absolute mode : If Position is 500, trigger output is generated at 500 position unconditionally.



● [AxmTriggerOnlyAbs](#)

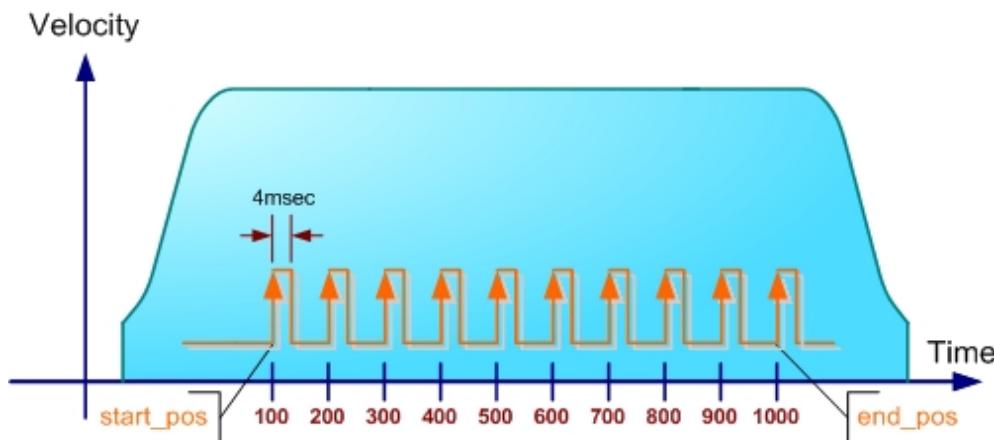
If the position to generate trigger is irregular and a lot, API has to be called repeatedly as much as trigger occurrence position with use of API previously mentioned. But this inconvenience can be reduced using [AxmTriggerOnlyAbs](#) API. This API generates trigger on the corresponding position receiving the position to generate trigger with array.

```
//Generate trigger at the assigned 10 positions during moving on axis 0.
long lAxisNo = 0;
long lTrigNum = 10;
double dTrigPos = {100, 150, 200, 250, 400, 520, 650, 700, 800, 980};
AxmTriggerOnlyAbs (lAxisNo, lTrigNum, dTrigPos);
```

● [AxmTriggerSetBlock](#)

This API sets start point to generate trigger and end point, and generates trigger once on a certain distance within the section.

```
//Generate trigger whenever moved as much as 100 from 300 position to 1000 position on axis 0
long lAxisNo = 0;
double dStartPosition = 100;
double dEndPosition = 1000;
double dPeriodPosition = 100;
AxmTriggerSetBlock (lAxisNo, dStartPosition, dEndPosition, dPeriodPosition);
```

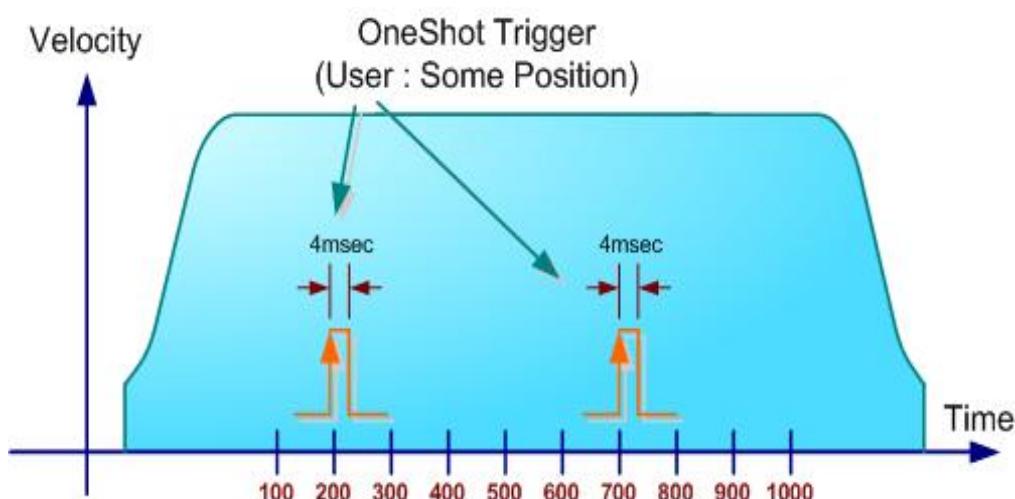


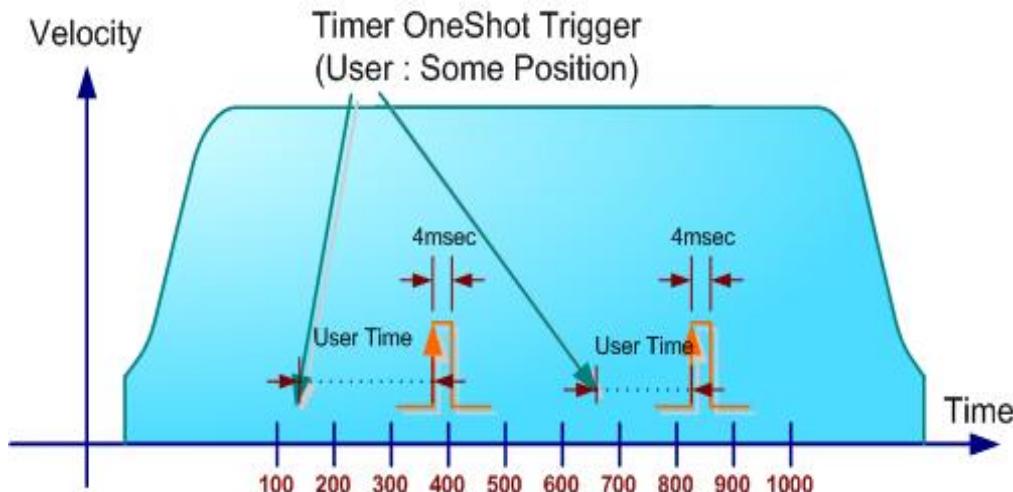
● [AxmTriggerOneShot](#) / [AxmTriggerSetTimerOneshot](#)

This API is that user outputs one trigger signal at an arbitrary point. [AxmTriggerOneShot](#) API generates trigger as soon as it is called, and [AxmTriggerSetTimerOneshot](#) API generates trigger going by a certain time after API call by input time.

```
//Generate trigger at the current point.
long lAxisNo = 0;
AxmTriggerOneShot (lAxisNo);

// Generate trigger after 1000ms(1sec).
long lAxisNo = 0;
long lSec = 1000;
AxmTriggerSetTimerOneshot (lAxisNo, lSec);
```





```
// Ex22_AXM_Trigger.cpp : Defines the entry point for the console application.
// Generate trigger signal with driving.
```

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

#define AXIS_0 0

void main(void)
{
    // Initialize library..
    // 7 means IRQ. IRQ is set automatically in PCI.
    DWORD Code = AxOpen(7);
    if (Code == AXT_RT_SUCCESS)
    {
        printf("Library is initialized.\n");

        // Inspect whether motion module exists or not
        DWORD dwStatus;
        Code = AxmInfoMotionModule(&dwStatus);
        if(Code == AXT_RT_SUCCESS)
        {
            if(dwStatus == STATUS_EXIST)
            {
                printf("Motion module exists.\n");

                // Set Active level of +End limit and -End limit on axis 0 to HIGH.
                AxmSignalSetLimit(nAxisNo, 0, HIGH, HIGH);
                // Set input level of inposition signal on axis 0 to HIGH.
                AxmSignalSetInpos (nAxisNo, HIGH);
                // Set input level of alarm signal on axis 0 to LOW.
                AxmSignalSetServoAlarm (nAxisNo, LOW);
                // Set Active input level of ESTOP signal on axis 0 to HIGH.
            }
        }
    }
}
```

```

AxmSignalSetStop (nAxisNo, 0, HIGH);

// Make ordered value on axis 0 to be mm unit.
AxmMotSetMoveUnitPerPulse(nAxisNo, 10, 10000);
// Set initial velocity on axis 0 to 1. Default : 1
AxmMotSetMinVel(nAxisNo, 1);
// Make pulse output method on axis 0 to TwoCwCcwHigh.
AxmMotSetPulseOutMethod (nAxisNo, TwoCcwcwHigh);
// Set encoder input method on designated axis to 4 multiplication.
AxmMotSetEnclnputMethod (AXIS_0, ObverseSqr4Mode);

//Set to absolute position drive.
AxmMotSetAbsRelMode (AXIS_0, 0);
//Set to symmetric S-curve velocity file.
AxmMotSetProfileMode (AXIS_0, 4);

// Set home search method
AxmHomeSetMethod (AXIS_0,-1,HomeSensor,1,2000.0,0.0);
// Set velocity that will be used during home search
AxmHomeSetVel(AXIS_0, 100, 20, 10, 10, 200, 40);

//Servo On
AxmSignalServoOn(AXIS_0, ENABLE);

// Search home.
AxmHomeSetStart (AXIS_0);

// Verify current home search progress rate
DWORD uHomeMainStepNumber , uHomeStepNumber_0;
AxmHomeGetRate(AXIS_0, &uHomeMainStepNumber ,
&uHomeStepNumber_0);
while(uHomeStepNumber_0 < 100)
{
    AxmHomeGetRate(AXIS_0, &uHomeMainStepNumber ,
&uHomeStepNumber_0);
    printf("Wrln home search : %d% ", uHomeStepNumber_0);
}

// help Message
printf("Wn[INFORMATION]***** Wn");
printf("[ESC] : Exit Wn");
printf("[1] : Generate trigger at every 50 going to 200 with (+) direction Wn");
printf("[2] : Generate trigger at assigned 10 positions going to 200 with (+)
direction Wn");
printf("[3] : Generate trigger from 50 to 150 position with 10 interval going to
200 with (+) direction Wn");
printf("***** Wn");

//Set trigger signal level and duration on axis 0.
double dTrigTime = 4000; // 4ms
DWORD uTriggerLevel = HIGH;
DWORD uSelect = 0; // Trigger generation on the basis of Command Position
DWORD ulnterrupt = DISABLE;
AxmTriggerSetTimeLevel (AXIS_0, dTrigTime, uTriggerLevel, uSelect,

```

```

ulInterrupt);

        DWORD uStatus;
        BOOL fExit = FALSE;
        while (!fExit) // Infinite loop
        {
            if (kbhit()) // If pressing any key
            {
                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': // Generate trigger at every 50 position during
                               movement
                        uMethod = 0; // [0] : Periodic trigger method,
                                     // [1] : Absolute position trigger method
                        dPosition = 50;
                        AxmTriggerSetAbsPeriod(AXIS_0, uMethod, dPosition);

                        //Drive up to 200 position.
                        AxmMoveStartPos(AXIS_0, 200, 50, 100, 100);
                        AxmStatusReadInMotion(AXIS_0, &uStatus);
                        while(uStatus) {
                            AxmStatusReadInMotion(AXIS_0, &uStatus);
                        }

                        //Return again to home.
                        AxmMoveStartPos(AXIS_0, 0, 100, 200, 200);
                        AxmStatusReadInMotion(AXIS_0, &uStatus);
                        while(uStatus) {
                            AxmStatusReadInMotion(AXIS_0, &uStatus);
                        }
                        break;

                    case '2': // Generate trigger at assigned 10 positions during
                               movement
                        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);

                        //Drive up to 200 position.
                }
            }
        }
    }
}

```

```

        AxmMoveStartPos (AXIS_0, 200, 50, 100, 100);
        AxmStatusReadInMotion (AXIS_0, &uStatus);
        while(uStatus) {
            AxmStatusReadInMotion (AXIS_0, &uStatus);
        }

        //Return again to home.
        AxmMoveStartPos (AXIS_0, 0, 100, 200, 200);
        AxmStatusReadInMotion (AXIS_0, &uStatus);
        while(uStatus) {
            AxmStatusReadInMotion (AXIS_0, &uStatus);
        }
        break;

    case '3': //Generate trigger from 50 to 150 position whenever
               moves every 10 on axis 0
        dStartPosition = 50;
        dEndPosition = 150;
        dPeriodPosition = 10;
        AxmTriggerSetBlock (AXIS_0, dStartPosition, dEndPosition,
                             dPeriodPosition);

        //Drive up to 200 position.
        AxmMoveStartPos (AXIS_0, 200, 50, 100, 100);
        AxmStatusReadInMotion (AXIS_0, &uStatus);
        while(uStatus) {
            AxmStatusReadInMotion (AXIS_0, &uStatus);
        }

        //Return again to home.
        AxmMoveStartPos (AXIS_0, 0, 100, 200, 200);
        AxmStatusReadInMotion (AXIS_0, &uStatus);
        while(uStatus) {
            AxmStatusReadInMotion (AXIS_0, &uStatus);
        }
        break;
    }

} //End of While()
}

else
    printf ("AxmlInfoIsMotionModule () : ERROR ( NOT STATUS_EXIST )
           code 0x%x\n",Code);
}

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

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

// Exit library.
if (AxlClose())
    printf("Library is exited.\n");

```

```

    else
        printf("Library is not exited normally.\n");
}

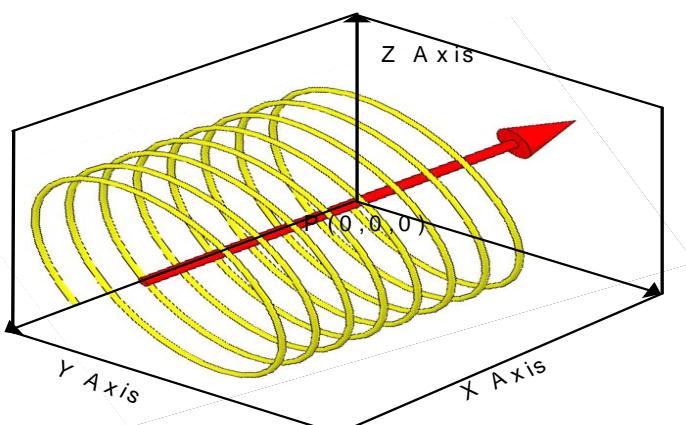
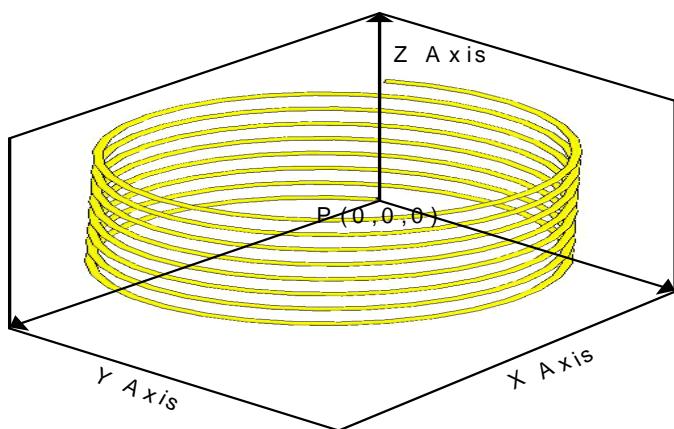
```

Advanced Motion Drive(PCI-N804/404 Board Exclusive Use API)

So far, various APIs for general motion control are discussed. In this chapter, several functions that are must required for APIs for advanced motion drive only on CAMC-QI chip will be discussed.

Helical Interpolation Motion

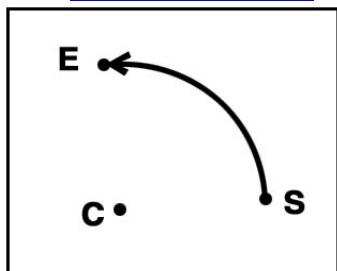
Helical interpolation control is a function of driving with complex 4 axes of 2 axes circular interpolation and straight line interpolation. With this, 2 axes straight line interpolation and circular interpolation can be carried out with synchronization. It provides two APIs: API that can drive one at once, and API used together with continuous interpolation function. To drive helical interpolation, circular interpolation movement must be move with synchronization velocity and moving amount with U axis (number 4 axis). Always last axis is set to U axis, and since U axis moves with synchronization with circular interpolation, moving amount follows Z axis and velocity cannot be used differently from circular interpolation. In case of 4 axis motion board, X(0),Y(1),Z(2),U(3) are used, and for 8 axis motion board, it can be used setting X(0),Y(1),Z(2),U(3) and X2(4),Y2(5),Z2(6),U(7).



● Helical Interpolation Drive

Circular interpolation drive is to be interpolation drive with circular path based on the corresponding setting from the current position to the assigned position. And this circular interpolation drive APIs provide [AxmHelixCenterMove](#), [AxmHelixRadiusMove](#), [AxmHelixAngleMove](#), and [AxmHelixPointMove](#) API.

● [AxmHelixCenterMove](#)

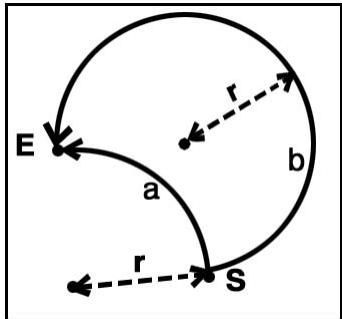


[AxmHelixCenterMove](#) API carries out circular interpolation drive from the current position (S) with input of exit point (E) and center point (C). Since this is a method to calculate circular arc using the position of three points, circle is composed of that accurate center point and exit point are inputted, and API is operated. If a wrong coordination is inputted, API will not be operated and returns FALSE. If circular interpolation drive is needed to carry out counterclockwise with center of (0, 100) from the current position (0,0) to (100,100) position, do the following.

```
// Carry out circular interpolation with center of (0, 100) from the current position (0,0) to (100,100)
position.
long lAxisNo1 = 0;
long lAxisNo2 = 1;
DWORD uAbsRelMode = 1;
DWORD uProfileMode = 3;
AxmMotSetAbsRelMode (lAxisNo1, uAbsRelMode); // Set to relative position drive
AxmMotSetProfileMode (lAxisNo1, uProfileMode); // Use S-curve velocity profile

double dCenterXPosition = 0;
double dCenterYPosition = 100;
double dEndXPosition = 100;
double dEndYPosition = 100;
double dMaxVelocity = 100;
double dMaxAccel = 200;
double dMaxDecel = 200;
DWORD uCWDDir = 1; // [0] Ccw direction, [1] Cw direction
AxmHelixCenterMove(lCoordinate, dCenterXPosition, dCenterYPosition, dEndXPosition,
dEndYPosition, dMaxVelocity, dMaxAccel, dMaxDecel, uCWDDir);
```

● [AxmHelixRadiusMove](#)



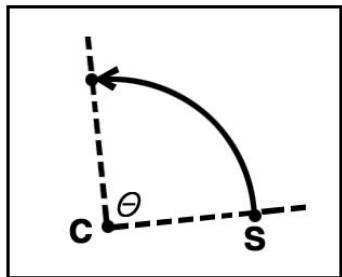
[AxmHelixRadiusMove](#) API carries out circular interpolation drive from the current position (S) with input of exit point (E) and radius (r). Since this is a method to calculate arc that passes two points and has constant radius, two circles are to be structured. Therefore, if uShortDistance value is 0 for the input radius, circle with short distance (a) is selected, and if 1, circle with long distance (b) is selected.

If circular interpolation is needed to carry out clockwise with long distance that has a radius of 120 from the current position to (300,200) position, do the following.

```
// Carry out circular interpolation from the current position to (300,200) position with long distance
// that has a radius of 120
long lAxisNo1 = 0;
long lAxisNo2 = 1;
DWORD uAbsRelMode = 1;
DWORD uProfileMode = 3;
AxmMotSetAbsRelMode (lAxisNo1, uAbsRelMode); // Set to relative position drive
AxmMotSetProfileMode (lAxisNo1, uProfileMode); // Use S-curve velocity profile

double dRadius = 120;
double dEndXPosition = 300
double dEndYPosition = 200;
double dMaxVelocity = 100;
double dMaxAccel = 200;
double dMaxDecel = 200;
DWORD uCWDdir = 0; // [0] Cw direction, [1] Ccw direction
DWORD uShortDistance = 1; // [0]: Small (circular) distance , [1] Big (circular) distance
AxmHelixRadiusMove (lCoordinate, dRadius, dEndXPosition, dEndYPosition, dMaxVelocity,
dMaxAccel, dMaxDecel, uCWDdir, uShortDistance);
```

● [AxmHelixAngleMove](#)



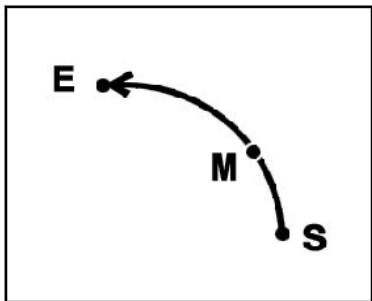
[AxmHelixAngleMove](#) API carries out circular interpolation drive from the current position (S) with input of center point (C) and rotational angle (θ). This is a method to calculate circular arc using radius and angle of arc, and a circle is structured.

Circular interpolation is carried out as much as 200 degrees with center of (200,200) from the current position, and if interpolation drive is needed to carry out clockwise, do the following.

```
// Carried out circular interpolation as much as 200 degrees with center of (200,200) from the
// current position.
long lAxisNo1 = 0;
long lAxisNo2 = 1;
DWORD uAbsRelMode = 1;
DWORD uProfileMode = 3;
AxmMotSetAbsRelMode (lAxisNo1, uAbsRelMode); // Set to relative position drive
AxmMotSetProfileMode (lAxisNo1, uProfileMode); // Use S-curve velocity profile

double dCenterXPosition = 200;
double dCenterYPosition = 200;
double dAngle = 200;
double dMaxVelocity = 100;
double dMaxAccel = 200;
double dMaxDecel = 200;
DWORD uCWDdir = 0; // [0] Cw direction, [1] Ccw direction
AxmHelixAngleMove(lCoordinate, dCenterXPosition, dCenterYPosition, dAngle, dMaxVelocity,
dMaxAccel, dMaxDecel, uCWDdir);
```

● [AxmHelixPointMove](#)



[AxmHelixPointMove](#) API carries out circular interpolation drive from the current position (S) with input of middle point (M) and exit point (E). This is a method to calculate circular arc that passes three points, and a circle is structured.

If interpolation drive is needed to carry out from the current position to (150,150) passing (100,150), do the following.

```
// Carry out circular interpolation from the current position to (150,150) passing (100,150).
long lAxisNo1 = 0;
long lAxisNo2 = 1;
DWORD uAbsRelMode = 1;
DWORD uProfileMode = 3;
AxmMotSetAbsRelMode (lAxisNo1, uAbsRelMode); // Set to relative position drive
AxmMotSetProfileMode (lAxisNo1, uProfileMode); // Use S-curve velocity profile

double dMidXPosition = 100;
double dMidYPosition = 50;
double dEndXPosition = 150;
double dEndYPosition = 150;
double dMaxVelocity = 100;
double dMaxAccel = 200;
double dMaxDecel = 200;
AxmHelixPointMove (lCoordinate, dMidXPosition, dMidYPosition, dEndXPosition, dEndYPosition,
dMaxVelocity, dMaxAccel, dMaxDecel);
```

[1\) Continuous Helical Interpolation Motion](#)

1) Continuous Helical Interpolation Motion

This API is what stores on internal queue for helical continuous interpolation drive on the assigned coordination system when use [AxmContiBeginNode](#) and [AxmContiEndNode](#). (Use with continuous interpolation API)

Continuous interpolation drive API is what is to drive interpolation drive mentioned before continuously, and has a drive that accelerates after starting drive, passes assigned positions with constant velocity, and decelerates and stops.

One unique thing is that, after axes to use for continuous interpolation takes mapping procedure to the new coordination system, stores commands to carry out interpolation drive continuously in the memory area of “continuous interpolation queue,” if continuous interpolation start command is given later, value goes into motion control chip after coming out one by one from the stored continuous interpolation data, and carries out interpolation drive continuously. Of course, interpolation data can be inputted after interpolation drive is started.

In order to carry out continuous interpolation drive, first, axes to be used for continuous interpolation must take a process of mapping. Interpolation drive APIs that are used for continuous interpolation are to be a combination of 2 axes interpolation drives, but they have to take mapping to one coordination system that includes all axes to be used after mapping of axes. For example, if it is carried out interpolation drive of 1 and 2 axis after interpolation drive of 0 and 1 axis continuously using three axes of 0, 1, and 2, they have to take mapping to one coordination system. After mapping, set velocity profile to use in corresponding coordination system, and set absolute/relative position drive.

```
// Ex24_AXM_Helix.cpp : Defines the entry point for the console application.
// Start helical continuous interpolation drive after home setting of 3 axes mechanical part .
#include "stdafx.h"
#include "AXL.h"
#include <conio.h>
#include "stdio.h"

#define AXIS_0 0
#define AXIS_1 1
#define AXIS_2 2
#define AXIS_3 3

void main(void)
{
    // Initialize library.
    // 7 means IRQ. IRQ is set automatically in PCI.
    DWORD Code = AxIOpen(7);
    if (Code == AXT_RT_SUCCESS)
    {
        printf("Library is initialized.\n");

        // Inspect whether motion module exists or not
        DWORD dwStatus;
        Code = AxmInfoIsMotionModule(&dwStatus);
        if(Code == AXT_RT_SUCCESS)
        {
            if(dwStatus == STATUS_EXIST)
```

```

{
    printf("Motion module exists.\n");

    for(int nAxisNo = 0; nAxisNo<3; nAxisNo++)
    {
        // Set Active level of +End limit and -End limit on axis 0 to HIGH and
        // emergency stop.
        AxmSignalSetLimit(nAxisNo, 0, HIGH, HIGH);
        // Set input level of inposition signal on axis 0 to HIGH.
        AxmSignalSetInpos (nAxisNo, HIGH);
        // Set input level of alarm signal on axis 0 to LOW.
        AxmSignalSetServoAlarm (nAxisNo, LOW);
        // Set Active input level of ESTOP signal on axis 0 to HIGH.
        AxmSignalSetStop (nAxisNo, 0, HIGH);

        // Make ordered value on axis 0 to be mm unit.
        AxmMotSetMoveUnitPerPulse(nAxisNo, 10, 10000);
        // Set initial velocity on axis 0 to 1. Default : 1
        AxmMotSetMinVel(nAxisNo, 1);
        // Make pulse output method on axis 0 to TwoCwCcwHigh.
        AxmMotSetPulseOutMethod (nAxisNo, TwoCwCcwHigh);
        // Set encoder input method on designated axis to 4 multiplication.
        AxmMotSetEnclnputMethod (nAxisNo, ObverseSqr4Mode);

        //Set to relative coordination drive
        AxmMotSetAbsRelMode (nAxisNo, 1);
        //Set to symmetrical S-curve velocity profile
        AxmMotSetProfileMode (nAxisNo, 3);

        // Set home search method
        AxmHomeSetMethod (nAxisNo,-1,HomeSensor,1,2000.0,0.0);
        // Set velocity that will be used during home search
        AxmHomeSetVel(nAxisNo, 100, 20, 10, 10, 200, 40);

        AxmStatusSetActPos(nAxisNo, 0.0);
        AxmStatusSetCmdPos(nAxisNo, 0.0);

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

    // Search home
    printf("Start home search pressing any key.\n");
    getch();
    AxmHomeSetStart (AXIS_0);
    AxmHomeSetStart (AXIS_1);
    AxmHomeSetStart (AXIS_2);

    // Verify current home search progress rate
    DWORD uHomeResult_0, uHomeResult_1, uHomeResult_2;
    DWORD uHomeMainStepNumber , uHomeStepNumber_0,
    uHomeStepNumber_1, uHomeStepNumber_2;
    AxmHomeGetResult (AXIS_0, &uHomeResult_0);
    AxmHomeGetResult (AXIS_1, &uHomeResult_1);
}

```

```

AxmHomeGetResult (AXIS_2, &uHomeResult_2);

while(uHomeResult_0 == HOME_SEARCHING || uHomeResult_1 ==
      HOME_SEARCHING || uHomeResult_2 == HOME_SEARCHING )
{
    AxmHomeGetResult (AXIS_0, &uHomeResult_0);
    AxmHomeGetResult (AXIS_1, &uHomeResult_1);
    AxmHomeGetResult (AXIS_2, &uHomeResult_2);

    AxmHomeGetRate(AXIS_0, &uHomeMainStepNumber ,
                      &uHomeStepNumber_0);
    AxmHomeGetRate(AXIS_1, &uHomeMainStepNumber ,
                      &uHomeStepNumber_1);
    AxmHomeGetRate(AXIS_2, &uHomeMainStepNumber ,
                      &uHomeStepNumber_2);

    printf("Wr In home search... axis 0 : %d%, 1 axis : %d, 2 axis : %d% % ", 
           uHomeStepNumber_0, uHomeStepNumber_1, uHomeStepNumber_2);
}

long   lAxis[4];
long lCoordinate = 0;
double dVelocity = 200;
double dAccel = 400;

for(int i=0; i<4; i++)
{
    lAxis[i]      = i;
}

AxmContiWriteClear(lCoordinate);
AxmContiSetAxisMap(lCoordinate, 4, lAxis);
AxmContiSetAbsRelMode(lCoordinate, 1);

//Input helical interpolation drive data.
AxmContiBeginNode (lCoordinate);
AxmHelixCenterMove(lCoordinate, 50, 50, 0, 0, 100, dVelocity, dAccel, dAccel, 1);
AxmHelixCenterMove(lCoordinate, 50, 50, 0, 0, 100, dVelocity, dAccel, dAccel, 1);
AxmHelixCenterMove(lCoordinate, 50, 50, 0, 0, 100, dVelocity, dAccel, dAccel, 1);
AxmHelixCenterMove(lCoordinate, 50, 50, 0, 0, 100, dVelocity, dAccel, dAccel, 1);
AxmHelixCenterMove(lCoordinate, 50, 50, 0, 0, 100, dVelocity, dAccel, dAccel, 1);
AxmHelixCenterMove(lCoordinate, 50, 50, 0, 0, 100, dVelocity, dAccel, dAccel, 1);
AxmContiEndNode (lCoordinate);

AxmContiStart (lCoordinate , 0, 0);

printf("WnExit pressing any key.Wn");
getch();

}
else
printf ("AxmlInfoMotionModule () : ERROR ( NOT STATUS_EXIST )
        code 0x%xWn",Code);
}

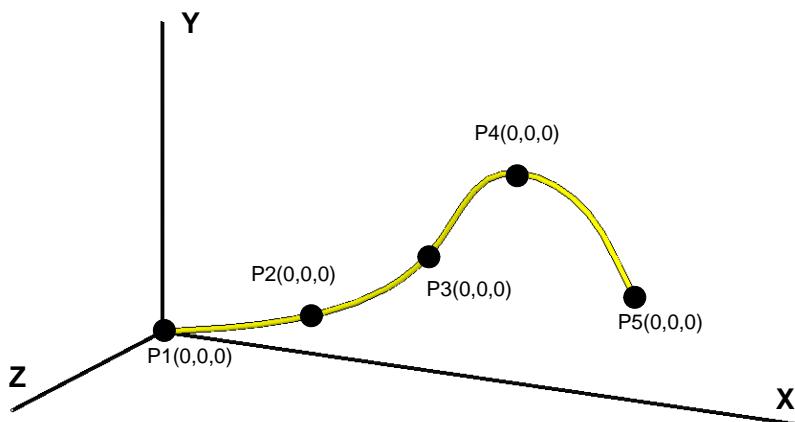
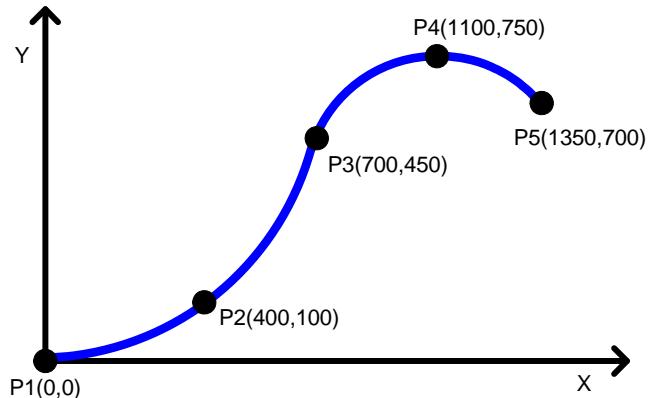
```

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

// Exit library.
if (AxlClose())
    printf("Library is exited.\n");
else
    printf("Library is not exited normally.\n");
}
```

Spline Interpolation Motion

Spline interpolation control provides B spline interpolation function. B spline interpolation shows below, and it changes data points assigned by user to free curve and interpolates. To drive this interpolation with direct motion, it needs to be used with continuous interpolation motion function. It provides basic X, Y, and Z axis. In case of 4 axis motion board, 3 axes out of X(0),Y(1),Z(2),U(3) can be selected and used. And, for 8 axis motion board, 3 axes out of X(0),Y(1),Z(2),U(3) in one chip and 3 axes out of X2(4),Y2(5),Z2(6),U(7) in other chip can be selected and used.



```
// Ex25_AXM_Spline.cpp : Defines the entry point for the console application.
// Start continuous 2 dimension spline interpolation drive after home setting of 2 axes mechanical part.
#include "stdafx.h"
#include "AXL.h"
#include <conio.h>
#include "stdio.h"

#define AXIS_0 0
#define AXIS_1 1

void main(void)
{
    // Initialize library..
```

```

// 7 means IRQ. IRQ is set automatically in PCI.

DWORD Code = AxIOpen(7);
if (Code == AXT_RT_SUCCESS)
{
    printf("Library is initialized.\n");

    // Inspect whether motion module exists or not
    DWORD dwStatus;
    Code = AxmInfoIsMotionModule (&dwStatus);
    if(Code == AXT_RT_SUCCESS)
    {
        if(dwStatus == STATUS_EXIST)
        {
            printf("Motion module exists.\n");

            for(int nAxisNo = 0; nAxisNo<2; nAxisNo++)
            {
                // Set Active level of +End limit and -End limit on axis 0 to HIGH and emergency stop.

                AxmSignalSetLimit(nAxisNo, 0, HIGH, HIGH);
                // Set input level of inposition signal on axis 0 to HIGH.
                AxmSignalSetInpos (nAxisNo, HIGH);
                // Set input level of alarm signal on axis 0 to LOW.
                AxmSignalSetServoAlarm (nAxisNo, LOW);
                // Set Active input level of ESTOP signal on axis 0 to HIGH.
                AxmSignalSetStop (nAxisNo, 0, HIGH);

                // Make ordered value on axis 0 to be mm unit.
                AxmMotSetMoveUnitPerPulse(nAxisNo, 10, 10000);
                // Set initial velocity on axis 0 to 1. Default : 1
                AxmMotSetMinVel(nAxisNo, 1);
                // Make pulse output method on axis 0 to TwoCwCcwHigh.
                AxmMotSetPulseOutMethod (nAxisNo, TwoCwCcwHigh);
                // Set encoder input method on designated axis to 4 multiplication.
                AxmMotSetEncInputMethod (nAxisNo, ObverseSqr4Mode);

                //Set to relative coordination drive
                AxmMotSetAbsRelMode (nAxisNo, 1);
                //Set to symmetrical S-curve velocity profile
                AxmMotSetProfileMode (nAxisNo, 3);

                // Set home search method
                AxmHomeSetMethod (nAxisNo,-1,HomeSensor,1,2000.0,0.0);
                // Set velocity that will be used during home search
                AxmHomeSetVel(nAxisNo, 100, 20, 10, 10, 200, 40);

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

            // Search home
            printf("Start home search pressing any key.\n");
            getch();
        }
    }
}

```

```

AxmHomeSetStart (AXIS_0);
AxmHomeSetStart (AXIS_1);

// Verify current home search progress rate
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("Wr In home search... axis 0 : %d%, 1 axis : %d % ",
           uHomeStepNumber_0, uHomeStepNumber_1);
}

printf("WnMove to relative position (50,25) with straight line interpolation drive.");

long  lAxesNo[2] = {0,1};
double dpPosition[2] = {50,25};
double dMaxVelocity = 100;
double dMaxAccel = 200;
double dMaxDecel = 200;
long  lCoordinate = 0;
double dVelocity = 200;
double dAccel = 400;

AxmContiWriteClear(lCoordinate);
AxmContiSetAxisMap(lCoordinate, 2, lAxesNo);
AxmContiSetAbsRelMode (lCoordinate, 1); // Set to relative position drive
AxmLineMove (lCoordinate, dpPosition, dMaxVelocity, dMaxAccel, dMaxDecel);

//Wait until all motions are exited.
BOOL WaitForDone = TRUE;
DWORD uStatus0, uStatus1;
while(WaitForDone) {
    AxmStatusReadInMotion (AXIS_0, &uStatus0);
    AxmStatusReadInMotion (AXIS_1, &uStatus1);
    if(uStatus0 == 0 && uStatus1 == 0)
        WaitForDone = FALSE;
}

double dPosX[5];
double dPosY[5];
AxmContiWriteClear(lCoordinate);
AxmContiSetAxisMap(lCoordinate, 2, lAxesNo);
AxmContiSetAbsRelMode(lCoordinate, 0);

dPosX[0] = 0 , dPosY[0] = 0;
dPosX[1] = 450, dPosY[1] = 100;

```

```

dPosX[2] = 750, dPosY[2] = 450;
dPosX[3] = 1080, dPosY[3] = 750;
dPosX[4] = 1300, dPosY[4] = 700;

AxmContiBeginNode (ICoordinate);
AxmSplineWrite(ICoordinate, 5, dPosX, dPosY, dVelocity, dAccel, dAccel, 1, 100);
AxmContiEndNode (ICoordinate);

printf("WnExit pressing any key.");
getch();

}

else
printf ("AxmlInfoIsMotionModule() : ERROR ( NOT STATUS_EXIST )
        code 0x%xWn",Code);
}

else
printf ("AxmlInfoIsMotionModule() : ERROR ( Return FALSE )   code 0x%xWn",Code);

}

else
printf ("AxlOpen() : ERROR   code 0x%xWn",Code);

// Exit library.
if (AxlClose())
printf("Library is exited.Wn");
else
printf("Library is not exited normally.Wn");
}

```

Example of 3 dimension spline interpolation

```

// Ex26_AXM_Spline2.cpp : Defines the entry point for the console application.
// Start continuous 3 dimension spline interpolation drive after home setting of 3 axes mechanical part.
#include "stdafx.h"
#include "AXL.h"
#include <conio.h>
#include "stdio.h"

#define AXIS_0 0
#define AXIS_1 1
#define AXIS_2 2

void main(void)
{
    // Initialize library..
    // 7 means IRQ. IRQ is set automatically in PCI.
    DWORD Code = AxlOpen(7);
    if (Code == AXT_RT_SUCCESS)
    {
        printf("Library is initialized.Wn");

```

```

// Inspect whether motion module exists or not
DWORD dwStatus;
Code = AxmlInfoIsMotionModule (&dwStatus);
if(Code == AXT_RT_SUCCESS)
{
    if(dwStatus == STATUS_EXIST)
    {
        printf("Motion module exists.\n");

        for(int nAxisNo = 0; nAxisNo<3; nAxisNo++)
        {
            // Set Active level of +End limit and -End limit on axis 0 to HIGH and emergency stop.

            AxmSignalSetLimit(nAxisNo, 0, HIGH, HIGH);
            // Set input level of inposition signal on axis 0 to HIGH.
            AxmSignalSetInpos (nAxisNo, HIGH);
            // Set input level of alarm signal on axis 0 to LOW.
            AxmSignalSetServoAlarm (nAxisNo, LOW);
            // Set Active input level of ESTOP signal on axis 0 to HIGH.
            AxmSignalSetStop (nAxisNo, 0, HIGH);

            // Make ordered value on axis 0 to be mm unit.
            AxmMotSetMoveUnitPerPulse(nAxisNo, 10, 10000);
            // Set initial velocity on axis 0 to 1. Default : 1
            AxmMotSetMinVel(nAxisNo, 1);
            // Make pulse output method on axis 0 to TwoCwCcwHigh.
            AxmMotSetPulseOutMethod (nAxisNo, TwoCwCcwHigh);
            // Set encoder input method on designated axis to 4 multiplication.
            AxmMotSetEnclInputMethod (nAxisNo, ObverseSqr4Mode);

            // Set home search method
            AxmHomeSetMethod (nAxisNo,-1,HomeSensor,1,2000.0,0.0);
            // Set velocity that will be used during home search
            AxmHomeSetVel(nAxisNo, 100, 20, 10, 10, 200, 40);

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

        // Search home
        printf("Start home search pressing any key.\n");
        getch();
        AxmHomeSetStart (AXIS_0);
        AxmHomeSetStart (AXIS_1);
        AxmHomeSetStart (AXIS_2);

        // Verify current home search progress rate
        DWORD uHomeResult_0, uHomeResult_1, uHomeResult_2;
        DWORD uHomeMainStepNumber , uHomeStepNumber_0,
        DWORD uHomeStepNumber_1, uHomeStepNumber_2;
        AxmHomeGetResult (AXIS_0, &uHomeResult_0);
        AxmHomeGetResult (AXIS_1, &uHomeResult_1);
        AxmHomeGetResult (AXIS_2, &uHomeResult_2);
    }
}

```

```

while(uHomeResult_0 == HOME_SEARCHING || uHomeResult_1 ==  

      HOME_SEARCHING || uHomeResult_2 == HOME_SEARCHING)  

{  

    AxmHomeGetResult (AXIS_0, &uHomeResult_0);  

    AxmHomeGetResult (AXIS_1, &uHomeResult_1);  

    AxmHomeGetResult (AXIS_2, &uHomeResult_2);  

AxmHomeGetRate(AXIS_0, &uHomeMainStepNumber , &uHomeStepNumber_0);  

AxmHomeGetRate(AXIS_1, &uHomeMainStepNumber , &uHomeStepNumber_1);  

AxmHomeGetRate(AXIS_2, &uHomeMainStepNumber , &uHomeStepNumber_2);  

printf("Wr In home search... axis 0 : %d%, 1 axis : %d, 2 axis : %d % ",  

      uHomeStepNumber_0, uHomeStepNumber_1, uHomeStepNumber_2);  

}  

printf("WnMove to relative position (100,100,100) with straight line interpolation drive.");  

long lAxesNo[3] = {0,1,2};  

double dpPosition[3] = {100,100,100};  

double dMaxVelocity = 100;  

double dMaxAccel = 200;  

double dMaxDecel = 200;  

long lCoordinate = 0;  

double dVelocity = 200;  

double dAccel = 400;  

AxmContiWriteClear(lCoordinate);  

AxmContiSetAxisMap(lCoordinate, 3, lAxesNo);  

AxmContiSetAbsRelMode (lCoordinate, 1); // // Set to relative position drive  

AxmLineMove (lCoordinate, dpPosition, dMaxVelocity, dMaxAccel, dMaxDecel);  

//Wait until all motions are exited.  

BOOL WaitForDone = TRUE;  

DWORD uStatus0, uStatus1, uStatus2;  

while(WaitForDone) {  

    AxmStatusReadInMotion (AXIS_0, &uStatus0);  

    AxmStatusReadInMotion (AXIS_1, &uStatus1);  

    AxmStatusReadInMotion (AXIS_2, &uStatus2);  

    if(uStatus0 == 0 && uStatus1 == 0 && uStatus2 == 0)  

        WaitForDone = FALSE;  

}  

double dPosX[5];  

double dPosY[5];  

AxmContiWriteClear(lCoordinate);  

AxmContiSetAxisMap(lCoordinate, 3, lAxesNo);  

AxmContiSetAbsRelMode(lCoordinate, 0);  

dPosX[0] = 0 , dPosY[0] = 0;  

dPosX[1] = 450, dPosY[1] = 100;  

dPosX[2] = 750, dPosY[2] = 450;  

dPosX[3] = 1080, dPosY[3] = 750;  

dPosX[4] = 1300, dPosY[4] = 700;

```

```
AxmContiBeginNode (ICoordinate);
AxmSplineWrite (ICoordinate, 5, dPosX, dPosY, dVelocity, dAccel, dAccel, 1, 100);
AxmContiEndNode (ICoordinate);

printf("WnExit pressing any key.");
getch();

}

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

else
printf ("AxmInfoIsMotionModule () : ERROR ( Return FALSE )   code 0x%xWn",Code);

}

else
printf ("AxlOpen() : ERROR   code 0x%xWn",Code);

// Exit library.
if (AxlClose())
    printf("Library is exited.Wn");
else
    printf("Library is not exited normally.Wn");
}
```

Motion Command Manual information

This manual is required for super ordinate motion modules including SMC-2V03 to drive with Microsoft VC++6.0, Visual Basic, Borland C-Builder, Delphi language in OS environment of Windows 98, Windows NT, Windows 2000 or Windows XP, and describes library API classified by function.

[Header File](#)

[Name of API in This Manual](#)

Header File

C
AXM.h
Visual Basic
AXM.bas
Delphi
AXM.pas

Name of API in This Manual

Names of API in this manual can classify motion by prefix.

※ Library API Prefix

Axm_ : shows that API is a motion module and library exclusive use. All APIs starting with Axm are defined as AXM.h.

Set_ : sets registers or variables of chip.

Get_ : verifies variable values set by Set API in pairs with Set or reads state of chip register.

Is_ : reads state of motion drive, continuous interpolation Queue, or Servo-On signal, etc.

_Move : is a drive start command, and if an API that includes this command is carried out, pulse is outputted immediately from motion module, but at the point of that pulse output is exited, it gets out of operation API.

Start_Move : is same with move command, but at the point of that pulse is outputted from motion module after API is started, it gets out of API.

_Line : sets an argument that is required for straight line interpolation drive and starts drive.

_Circle : sets an argument that is required for circular interpolation drive and starts drive.

Conti_ : sets an argument that is required for continuous interpolation drive and starts drive.

Vel_ : carries out continuous drive with velocity mode or operates velocity override.

Accel_ : sets acceleration of accelerating section (**unit of acceleration is PPS[Pulses/sec]** if Unit/pulse is 1/1) in drive or jerk value.

Decel_ : sets acceleration of decelerating section (**unit of acceleration is PPS[Pulses/sec]** if Unit/pulse is 1/1) in drive or jerk value.

Sensor_ : sets and inspects parameter used for sensor position search drive.

mpg_ : sets and inspects parameter that carries out MPG drive or is required.

CRC_ : sets and inspects parameter that carries out clear of remaining pulse or is required.

Common arguments of APIs used in this manual have following meaning.

long lAxis : means a specific axis number, and is a channel (axis) number (starting from 0) obtained by initialization API. Start of channel (axis) number (starting from 0) is “0.”

DWORD uLevel : means output or input level of signal in signal setting, and has HIGH(logic '1') or LOW(logic '0') value electrically. Or it also has Enable/ Disable value of signal.

double dPos : means a position value at absolute mode, and means a distance to move at relative mode. Unit is unit distance.

double dVelocity : means moving velocity, and unit is unit distance per second. That is, if unit distance is set by 1mm, velocity value as an argument of API is to be mm/sec.

double dAcceleration : means moving acceleration (**unit of acceleration is PPS[Pulses/sec] if Unit/pulse is 1/1**) or accelerating time. In case of moving acceleration (**unit of acceleration is PPS[Pulses/sec] if Unit/pulse is 1/1**), unit is change amount of velocity per second. Here, velocity is velocity used as an argument of API. If acceleration value is to be twice of velocity value, it means that acceleration time to maximum velocity is 0.5 sec. In case of acceleration time, unit is second. Once acceleration time is set, it drives with acceleration ratio value that is set automatically. If acceleration time is 2 sec, acceleration ratio is 1/2 times value of maximum velocity, and it means acceleration time from the drive start point to maximum velocity is 2 sec.

Terminology List

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	Mot
Motion Profile	
Continuous	Conti
Absolute	Abs
Relative	Rel
Manual Pulse Generation	MPG
Current Residual Clear	Crc
Number	Num

Motion Command Quick List

API List

[Board and Module Verification API \(Info\) – Information](#)
[Virtual Axis API \(Virtual\)](#)
[Interrupt API \(Interrupt\)](#)
[Motion Parameter Setting API \(Mot\) -> Motor Parameter](#)
[Input and Output Setting API \(Signal\)](#)
[State Verification API after Motion Drive \(Status\)](#)
[Home Search API \(Home\)](#)
[Position Drive API \(Move\)](#)
[Position and Velocity Override Drive API \(Override\)](#)
[MASTER, SLAVE Drive API\(Link\)](#)
[Gantry Drive API \(Gantry\)](#)
[Interpolation Drive API \(Line, Circle\)](#)
[Continuous Interpolation Setting and Drive API \(Conti\) -> Continue](#)
[Trigger API \(Trigger\)](#)
[CRC\(Remaining Pulse Clear\) Drive API](#)
[MPG\(Manul Pulse Generator\) Drive API](#)
[Helical Interpolation Setting and Drive API \(PCI-N804/404 Exclusive Use API\) -> Unusable in SMC-2V03](#)
[Spline Interpolation Setting and Drive API \(PCI-N804/404 Exclusive Use API\) -> Unusable in SMC-2V03](#)

[Define Sentence](#)

API List

[Board and Module Verification API \(Info\) – Information](#)
[Virtual Axis API \(Virtual\)](#)
[Interrupt API \(Interrupt\)](#)
[Motion Parameter Setting API \(Mot\) -> Motor Parameter](#)
[Input and Output Setting API \(Signal\)](#)
[State Verification API after Motion Drive \(Status\)](#)
[Home Search API \(Home\)](#)
[Position Drive API \(Move\)](#)
[Position and Velocity Override Drive API \(Override\)](#)
[MASTER, SLAVE Drive API\(Link\)](#)
[Gantry Drive API \(Gantry\)](#)
[Interpolation Drive API \(Line, Circle\)](#)
[Continuous Interpolation Setting and Drive API \(Conti\) -> Continue](#)
[Trigger API \(Trigger\)](#)
[CRC\(Remaining Pulse Clear\) Drive API](#)
[MPG\(Manul Pulse Generator\) Drive API](#)
[Helical Interpolation Setting and Drive API \(PCI-N804/404 Exclusive Use API\) -> Unusable in SMC-2V03](#)
[Spline Interpolation Setting and Drive API \(PCI-N804/404 Exclusive Use API\) -> Unusable in SMC-2V03](#)

Board and Module Verification API (Info) – Information

Function	Description
AxmInfoGetAxis	Returns board number, module position, module ID of corresponding axis.
AxmInfoIsMotionModule	Returns whether motion module exists.
AxmInfoIsInvalidAxisNo	Returns whether corresponding axis is effective.
AxmInfoGetAxisCount	Returns efftentive motion axis number installed in system.
AxmInfoGetFirstAxisNo	Returns the first axis number of corresponding board and module.

Virtual Axis API (Virtual)

Function	Description
AxmVirtualSetAxisNoMap	Sets virtual axis.
AxmVirtualGetAxisNoMap	Returns virtual channel (axis) number which is set.

AxmVirtualSetMultiAxisNoMap	Sets multi-virtual axis.
AxmVirtualGetMultiAxisNoMap	Returns multi-virtual channel (axis) which is set.
AxmVirtualResetAxisMap	Clears virtual axis setting.

Interrupt API (Interrupt)

Function	Description
AxIInterruptEnable	In order to use the interrupt, other interrupt functions are to be used after calling this function..
AxIInterruptDisable	This function is used to disable the interrupt
AxMIInterruptSetAxis	Uses window message or callback API to receive interrupt message.
AxMIInterruptSetAxisEnable	Sets whether interrupt on setting axis is used
AxMIInterruptGetAxisEnable	Returns whether interrupt on setting axis is used
AxMIInterruptRead	Reads corresponding interrupt information if interrupt is used by event method.
AxMIInterruptReadAxisFlag	Returns interrupt flag value on corresponding axis.
AxMIInterruptSetUserEnable	Sets whether interrupt set by user of specific axis is occurred.
AxMIInterrupt GetUserEnable	Verifies whether interrupt set by user of specific axis is occurred.

Motion Parameter Setting API (Mot) -> Motor Parameter

Function	Description
AxmMotLoadParaAll	Calls 33 parameter files stored with AxmMotSaveParaAll . Arbitrary change is available by user's calling.
AxmMotSaveParaAll	Stores all 33 parameters by axis for the current all axes. Store as .mot file. Call a file using AxmMotLoadParaAll.
AxmMotSetParaLoad	Use AxmMotSaveParaAll API to store information which is set on register of CAMC-IP, QI as a file. The API stores information of all axes as a file. Set 4 parameters of dInitPos, dInitVel, dInitAccel and dInitDecel only from 28 to 31 among parameters. The rest parameters are stored using AxmMotSaveParaAll API reading

	current state when library is closed.
<u>AxmMotGetParaLoad</u>	Use <u>AxmMotSaveParaAll</u> API to store information which is set on register of CAMC-IP, QI as a file. The API stores information of all axes as a file. The API is used to call only 4 parameters of dInitPos, dInitVel, dInitAccel and dInitDecel from 28 to 31 among parameters that information of all axes is stored.
<u>AxmMotSetPulseOutMethod</u>	Sets pulse output method on a specific axis.
<u>AxmMotGetPulseOutMethod</u>	Returns pulse output method setting on a specific axis.
<u>AxmMotSetEnclInputMethod</u>	Sets encoder input method on a specific axis including increasing direction setting of external (actual) count on a specific axis.
<u>AxmMotGetEnclInputMethod</u>	Returns encoder input method on a specific axis including increasing direction setting of external (actual) count on a specific axis.
<u>AxmMotSetMoveUnitPerPulse</u>	Sets moving distance per pulse on a specific axis.
<u>AxmMotGetMoveUnitPerPulse</u>	Returns moving distance per pulse on a specific axis.
<u>AxmMotSetDecelMode</u>	Sets deceleration start point search method on a specific axis.
<u>AxmMotGetDecelMode</u>	Returns deceleration start point search method on a specific axis.
<u>AxmMotSetRemainPulse</u>	Sets remaining pulse in manual deceleration mode on a specific axis.
<u>AxmMotGetRemainPulse</u>	Returns remaining pulse in manual deceleration mode on a specific axis.
<u>AxmMotSetMaxVel</u>	Sets maximum velocity in constant speed drive API on a specific axis.
<u>AxmMotGetMaxVel</u>	Returns maximum velocity in constant speed drive API on a specific axis.
<u>AxmMotSetAbsRelMode</u>	Sets moving distance calculation mode on a specific axis. If absolute position mode (0) is set, position value used in moving API is to be absolute position value. For example, if current position value is “10” and moves from absolute position mode to “100” position, it will stop at “100” position moving as much as “+90” from the current position. If relative position mode (1) is set, position value used moving

	API is to be moving amount to move from the current position. For example, if current position value is “10” and moves from relative position mode to “100” position, it will stop at “110” position moving as much as “+100” from the current position.
AxmMotGetAbsRelMode	Returns moving distance calculation mode which is set on a specific axis.
AxmMotSetProfileMode	Sets drive velocity profile mode on a specific axis. Velocity profile mode has symmetrical or asymmetrical Trapezoide / S-Curve mode and Quasi-S mode.
AxmMotGetProfileMode	Returns drive velocity profile mode which is set on a specific axis.
AxmMotSetAccelUnit	Sets acceleration unit on a specific axis. Acceleration unit can be selected among u/s ² , sec, rev/s ² .
AxmMotGetAccelUnit	Returns acceleration unit which is set on a specific axis.
AxmMotSetMinVel	Sets initial velocity on a specific axis.
AxmMotGetMinVel	Returns initial velocity on a specific axis.
AxmMotSetAccelJerk	Sets acceleration jerk value on a specific axis.
AxmMotGetAccelJerk	Returns acceleration jerk value on a specific axis.
AxmMotSetDecelJerk	Sets deceleration jerk value on a specific axis.
AxmMotGetDecelJerk	Returns deceleration jerk value on a specific axis.

Input and Output Setting API (Signal)

Function	Description
AxmSignalSetZphaseLevel	Sets Z phase level on a specific axis.
AxmSignalGetZphaseLevel	Returns Z phase level on a specific axis.
AxmSignalSetServoOnLevel	Sets output level of Servo-On signal on a specific axis.
AxmSignalGetServoOnLevel	Returns output level of Servo-On signal on a specific axis.
AxmSignalSetServoAlarmResetLevel	Sets output level of Servo-Alarm Reset signal on a specific axis.
AxmSignalGetServoAlarmResetLevel	Returns output level of Servo-Alarm Reset signal on a specific axis.

<u>AxmSignalSetInpos</u>	Sets whether Inpositon signal is used and signal input level on a specific axis.
<u>AxmSignalGetInpos</u>	Returns whether Inpositon signal is used and signal input level on a specific axis.
<u>AxmSignalReadInpos</u>	Returns Inpositon signal input state on a specific axis.
<u>AxmSignalSetServoAlarm</u>	Sets whether emergency stop is used and signal input level during alarm signal input on a specific axis.
<u>AxmSignalGetServoAlarm</u>	Returns whether emergency stop is used and signal input level during alarm signal input on a specific axis.
<u>AxmSignalReadServoAlarm</u>	Returns input level of alarm signal on a specific axis.
<u>AxmSignalSetLimit</u>	Sets whether end limit sensor is used and input level of signal on a specific axis. Available to set for decelerate stop or emergency stop during end limit sensor signal input.
<u>AxmSignalGetLimit</u>	Returns use of end limit sensor and signal input level and stop mode during signal input on a specific axis.
<u>AxmSignalReadLimit</u>	Returns input state of end limit sensor on a specific axis.
<u>AxmSignalSetSoftLimit</u>	Sets use of Software limit, count to use, and stop method on a specific axis.
<u>AxmSignalGetSoftLimit</u>	Returns use of Software limit, count to use, and stop method on a specific axis.
<u>AxmSignalSetStop</u>	Sets stop method (emergency stop/decelerate stop) or use of emergency stop signal.
<u>AxmSignalGetStop</u>	Returns stop method (emergency stop/decelerate stop) or use of emergency stop signal.
<u>AxmSignalReadStop</u>	Returns input state of emergency stop signal.
<u>AxmSignalServoOn</u>	Outputs Servo-On signal on a specific axis.
<u>AxmSignalIsServoOn</u>	Returns output state of Servo-On signal on a specific axis.
<u>AxmSignalServoAlarmReset</u>	Outputs Servo-Alarm Reset signal on a specific axis.
<u>AxmSignalWriteOutput</u>	Sets universal output value.
<u>AxmSignalReadOutput</u>	Returns universal output value.
<u>AxmSignalWriteOutputBit</u>	Sets universal output value by bit.
<u>AxmSignalReadOutputBit</u>	Returns universal output value by bit.

AxmSignalReadInput	Returns universal input value.
AxmSignalReadInputBit	Returns universal input value by bit.

State Verification API after Motion Drive (**Status**)

Function	Description
AxmStatusReadInMotion	Returns pulse output state on a specific axis.
AxmStatusReadDrivePulseCount	Returns drive pulse counter value on a specific axis.
AxmStatusReadMotion	Returns DriveStatus register on a specific axis.
AxmStatusReadStop	Returns EndStatus register on a specific axis.
AxmStatusReadMechanical	Returns Mechanical Signal Data on a specific axis.
AxmStatusReadVel	Reads current drive velocity on a specific axis.
AxmStatusReadPosError	Returns difference between Command Position and Actual Position on a specific axis.
AxmStatusReadDriveDistance	Verifies moving (moved) distance to final drive.
AxmStatusSetActPos	Sets actual position on a specific axis.
AxmStatusGetActPos	Returns Actual position on a specific axis.
AxmStatusSetCmdPos	Sets Command position on a specific axis.
AxmStatusGetCmdPos	Returns Command position on a specific axis.

Home Search API(Home)

Function	Description
AxmHomeSetSignalLevel	Sets Home sensor Level on a specific axis.
AxmHomeGetSignalLevel	Returns Home sensor Level on a specific axis.
AxmHomeReadSignal	Returns input state of Home signal on a specific axis.

<u>AxmHomeSetMethod</u>	To execute home search on corresponding axis, parameters related to home search must be set. Every home search parameters can be stored and called using <u>AxmMotSaveParaAll</u> and <u>AxmMotLoadParaAll</u> . For home search method setting, set search progressing direction, signal to be used as home, home sensor active level, whether encoder Z phase is detected, and so on.
<u>AxmHomeGetMethod</u>	To execute home search on corresponding axis, parameters related to home search must be set. Return parameters related to already set home.
<u>AxmHomeSetVel</u>	For fast and accurate home search, detect with several steps. Set velocity to be used for each step. Home search time and accuracy are determined by the setting value of velocity. Need to set home search velocity of each axis changing velocity properly by step.
<u>AxmHomeGetVel</u>	For fast and accurate home search, detect with several steps. Set velocity to be used for each step. Home search time and accuracy are determined by the setting value of velocity. Get each set velocity.
<u>AxmHomeSetStart</u>	Once home search start API is executed, be exited automatically after execution of home search in turn with automatic generation of thread that executes home search of corresponding axis inside library.
<u>AxmHomeSetResult</u>	Search result is set to HOME_SUCCESS after successful execution of home search using home search API. The API can be set the result arbitrary without execution of home search by user. Home search can be executed again setting home search result to HOME_ERR_USER _BRE AK when abnormal condition occurred in normal drive after actual home search completion.
<u>AxmHomeGetResult</u>	Verify search result of home search API. Once home search is started, be set to HOME_SEARCHING, and if fail to home search, failure cause is set. Execute home search again after removal of failure cause.
<u>AxmHomeGetRate</u>	Progress rate can be verified after home search is started. If home search is completed, be returned 100 regardless of success. Home search success can be verified using GetHome Result API.

Position Drive API (Move)

Function	Description
<u>AxmMoveStartPos</u>	Drive with set velocity and acceleration rate up to set position with absolute/relative coordination on a specific axis. Set velocity profile in <u>AxmMotSetProfileMode</u> API. Get out of API at the point that pulse is outputted.
<u>AxmMovePos</u>	Drive with set velocity and acceleration rate up to set position with absolute/relative coordination on a specific axis. Set velocity profile in <u>AxmMotSetProfileMode</u> API. Get out of API at the point that pulse output is exited.
<u>AxmMoveVel</u>	Carry out velocity mode drive with set velocity and acceleration rate continuously. Get out of API at the point that pulse is outputted.
<u>AxmMoveStartMultiVel</u>	Carry out velocity mode drive with set velocity and acceleration rate for the specific multi-axis. Get out of API at the point that pulse output is started.
<u>AxmMoveSignalSearch</u>	Drive with set velocity and acceleration rate until set signal is detected. Get out of API at the point that pulse is outputted.
<u>AxmMoveSignalCapture</u>	API that moves to search the set signal and store the position on a specific axis.
<u>AxmMoveSignalCapture</u>	API that verifies stored position value in ' <u>AxmMoveSignalCapture</u> ' API.
<u>AxmMoveStartMultiPos</u>	Drive with set velocity and acceleration rate up to set position with absolute coordination of specific axes. Velocity profile is set in <u>AxmMotSetProfileMode</u> API. Get out of API at the point that pulse output is exited.
<u>AxmMoveMultiPos</u>	Drive with set velocity and acceleration rate up to set position with absolute coordination of specific axes. Velocity profile is set in <u>AxmMotSetProfileMode</u> API. Get out of API at the point that pulse is outputted.
<u>AxmMoveStop</u>	Make a specific axis decelerate and stop with set deceleration.
<u>AxmMoveEStop</u>	Make a specific axis to emergency stop.
<u>AxmMoveSStop</u>	Make a specific axis to decelerate and stop.

Position and Velocity Override Drive API (Override)

Function	Description
AxmOverridePos	Adjust a specific output pulse number before drive exit on a specific axis.
AxmOverrideSetMaxVel	Set maximum velocity to override before velocity override on a specific axis.
AxmOverrideVel	Make variable setting of velocity during drive on a specific axis.
AxmOverrideVelAtPos	Drive velocity mode continuously with set velocity and acceleration rate for the specific axis. Execute velocity override at the position which is set during driving. Get out of API at the point of that pulse output is started.

MASTER, SLAVE Drive API (Link)

Function	Description
AxmLinkSetMode	Set electric gear ratio between Master and Slave axis.
AxmLinkGetMode	Return electric gear ratio between Master and Slave axis.
AxmLinkResetMode	Clear setting of electric gear ratio between Master and Slave axis.

Gantry Drive API (Gantry)

Function	Description
AxmGantrySetEnable	Motion module supports gantry drive system control mechanically linked with two axes. Using this API, if master axis is set to gantry control, corresponding slave axis is synchronized with master axis and driven. Even if drive or stop command is given on slave axis after gantry setting, all commands are ignored.
AxmGantryGetEnable	Return set parameter when master axis is set to gantry control using this API.
AxmGantrySetDisable	If home search is executed after gantry control setting, slave axis executes home search after master axis executes home search. This API is used to verify error value between home on master

	axis and home on slave axis after home search.
--	--

Interpolation Drive API (Line, Circle)

Function	Description
<u>AxmLineMove</u>	This API executes multi-axis straight line interpolation drive assigning start point and end point. Get out of API after drive start. Using with <u>AxmContiBeginNode</u> and <u>AxmContiEndNode</u> , it is to be stored API in queue driving straight line interpolation assigning start point and end point on the specific coordination system. For straight line profile continuous interpolation, it stores in internal queue and starts using <u>AxmContiStart</u> API.
<u>AxmCircleCenterMove</u>	This API executes circular interpolation drive assigning start point, end point and middle point. Get out of API after drive start. Using with <u>AxmContiBeginNode</u> and <u>AxmContiEndNode</u> , it is to be stored API in queue driving circular interpolation assigning start point, end point and middle point on the specific coordination system. For profile circular continuous interpolation drive, it stores in internal queue and starts using <u>AxmContiStart</u> API.
<u>AxmCirclePointMove</u>	This API executes circular interpolation drive assigning middle point and end point. Get out of API after drive start. Using with <u>AxmContiBeginNode</u> and <u>AxmContiEndNode</u> , it is to be stored API in queue driving circular interpolation assigning middle point and end point on the specific coordination system. For profile circular continuous interpolation drive, it stores in internal queue and starts using <u>AxmContiStart</u> API.
<u>AxmCircleRadiusMove</u>	This API executes circular interpolation drive assigning start point, end point and radius. Get out of API after drive start. Using with <u>AxmContiBeginNode</u> and <u>AxmContiEndNode</u> , it is to be stored API in queue driving circular interpolation assigning start point, end point and radius on the specific coordination system. For profile circular continuous interpolation drive, it stores in internal queue and starts using <u>AxmContiStart</u> API.
<u>AxmCircleAngleMove</u>	This API executes circular interpolation drive assigning start point, rotational angle and radius. Get out of API after drive start. Using with <u>AxmContiBeginNode</u> and <u>AxmContiEndNode</u> , it is to be stored API in queue driving circular interpolation assigning start point, rotational angle and radius on the specific coordination

	system. For profile circular continuous interpolation drive, it stores in internal queue and starts using AxmContiStart API.
--	--

Continuous Interpolation Setting and Drive API (Conti) -> Continue

Function	Description
AxmContiSetAxisMap	Set continuous interpolation axis mapping on the specific coordination system.
AxmContiGetAxisMap	Return continuous interpolation axis mapping on the specific coordination system.
AxmContiSetAbsRelMode	Set continuous interpolation axis absolute/relative mode on the specific coordination system.
AxmContiGetAbsRelMode	Return continuous interpolation axis absolute/relative mode on the specific coordination system.
AxmContiBeginNode	Start registration of works to execute continuous interpolation on the specific coordination system. After calling this API, until AxmContiEndNode API is called, all motion work executed is not executed actual motion but registered to continuous interpolation motion. And when AxmContiStart API is called, registered motion is executed actually.
AxmContiEndNode	Exit registration of works to execute continuous interpolation on the specific coordination system.
AxmContiReadFree	To verify if internal queue is cleared for interpolation drive on the specific coordination system.
AxmContiReadIndex	To verify interpolation drive number stored in internal queue for interpolation drive on the specific coordination system.
AxmContiWriteClear	To clear all stored internal queue for continuous interpolation drive on the specific coordination system.
AxmContiStart	To start drive of internal continuous interpolation queue stored on the specific coordination system.
AxmContiIsMotion	To verify whether continuous interpolation drive is in drive on the specific coordination system.
AxmContiGetNodeNum	To verify continuous interpolation index number which is currently in drive during continuous interpolation drive on the specific coordination system.
AxmContiGetTotalNodeNum	To verify total index number of continuous interpolation drive set

	on the specific coordination system.
--	--------------------------------------

Trigger API (Trigger)

Function	Description
<u>AxmTriggerSetTimeLevel</u>	Set trigger signal duration, output level, and output method on a specific axis.
<u>AxmTriggerGetTimeLevel</u>	Return trigger signal duration, output level, and output method on a specific axis.
<u>AxmTriggerSetAbsPeriod</u>	Set use of trigger function, output level, position comparator, trigger signal duration and trigger output mode on a specific axis.
<u>AxmTriggerGetAbsPeriod</u>	Return use of trigger function, output level, position comparator, trigger signal duration and trigger output mode on a specific axis.
<u>AxmTriggerSetBlock</u>	Output trigger at every certain section from the start position to end position by user assignment.
<u>AxmTriggerGetBlock</u>	Read trigger setting value in ' <u>AxmTriggerSetBlock</u> ' API.
<u>AxmTriggerOneShot</u>	Output one trigger pulse by user.
<u>AxmTriggerSetTimerOneshot</u>	Output one trigger pulse after seconds by user.
<u>AxmTriggerOnlyAbs</u>	Output infinite absolute position of absolute position trigger.
<u>AxmTriggerSetReset</u>	Reset trigger setting.

CRC(Remaining Pulse Clear) Drive API

Function	Description
<u>AxmCrcSetMaskLevel</u>	Set use of CRC signal and output level on a specific axis.
<u>AxmCrcGetMaskLevel</u>	Return use of CRC signal and output level on a specific axis.
<u>AxmCrcSetOutput</u>	Generate forced CRC signal on a specific axis.
<u>AxmCrcGetOutput</u>	Return whether forced CRC signal is generated on a specific axis.
<u>AxmCrcSetEndLimit</u>	Set use of CRC signal and output level for limit on a specific axis.

AxmCrcGetEndLimit	Return use of CRC signal and output level for limit on a specific axis.
-----------------------------------	---

MPG(Manul pulse Generator) Drive API

Function	Description.
AxmMPGSetEnable	Set MPG input method, drive move mode, moving distance, velocity, etc on assigend axis.
AxmMPGGetEnable	Return MPG input method, drive move mode, moving distance, velocity, etc on assigend axis.
AxmMPGSetRatio	Set pulse ratio to move per pulse in MPG drive move mode on a specific axis.
AxmMPGGetRatio	Return pulse ratio to move per pulse in MPG drive move mode on a specific axis.
AxmMPGReset	Clear MPG drive setting on a specific axis.

Helical Interpolation Setting and Drive API

Function	Description
AxmHelixCenterMove	This API executes helical interpolation drive assigning start point, end point and middle point. Using with AxmContiBeginNode and AxmContiEndNode , it drives helical continuous interpolation assigning start point, end point and middle point on the specific coordination system. For circular continuous interpolation drive, it stores in internal queue. Start using AxmContiStart API. (use with continuous interpolation API)
AxmHelixPointMove	This API executes helical interpolation drive assigning start point, end point and radius. Using with AxmContiBeginNode and AxmContiEndNode , it drives helical continuous interpolation assigning start point and end point on the specific coordination system. For circular continuous interpolation drive, it stores in internal queue. Start using AxmContiStart API. (use with continuous interpolation API)
AxmHelixRadiusMove	This API executes helical interpolation drive assigning start point,

	<p>end point and radius.</p> <p>Using with AxmContiBeginNode and AxmContiEndNode, it drives helical continuous interpolation assigning start point, end point and radius on the specific coordination system.</p> <p>For circular continuous interpolation drive, it stores in internal queue. Start using AxmContiStart API. (use with continuous interpolation API)</p>
AxmHelixAngleMove	<p>This API executes helical interpolation drive assigning start point, rotational angle and radius.</p> <p>Using with AxmContiBeginNode and AxmContiEndNode, it drives helical continuous interpolation assigning start point, rotational angle and radius on the specific coordination system.</p> <p>For circular continuous interpolation drive, it stores in internal queue. Start using AxmContiStart API. (use with continuous interpolation API)</p>

Spline Interpolation Setting and Drive API

Function	Description
AxmSplineWrite	<p>Use with AxmContiBeginNode and AxmContiEndNode. This API executes spline continuous interpolation drive. For circular continuous interpolation drive, it stores in internal queue. Start using AxmContiStart API. (use with continuous interpolation API)</p>

Define Sentence

It is used in AXL Motion Library, and list of define sentence is following
 Header File : #include "AXHS.h"

FALSE & TRUE

define	Value	Explanation
FALSE	00h	False
TRUE	01h	True

DISABLE & ENABLE

define	Value	Explanation
DISABLE	00h	Not -used
ENABLE	01h	Used

AXT_MOTION_STOPMODE_DEF

#define	Value	Explanation
EMERGENCY_STOP	00h	Emergency stop
SLOWDOWN_STOP	01h	Decelerate stop

AXT_MOTION_EDGE_DEF

#define	Value	Explanation
SIGNAL_DOWN_EDGE	00h	Falling edge
SIGNAL_UP_EDGE	01h	Rising edge

AXT_MOTION_SELECTION_DEF

#define	Value	Explanation
COMMAND	00h	Command position
ACTUAL	01h	Actual position

AXT_MOTION_TRIGGER_MODE_DEF

#define	Value	Explanation
PERIOD_MODE	00h	Trigger periodic mode
ABS_POS_MODE	01h	Trigger absolute mode

AXT_MOTION_LEVEL_MODE_DEF

#define	Value	Explanation
LOW	00h	B Contact (NORMAL CLOSE)
HIGH	01h	A Contact (NORMAL OPEN)
UNUSED	02h	Not-used
USED	03h	Maintaine current state

AXT_MOTION_ABSREL_MODE_DEF

#define	Value	Explanation
POS_ABS_MODE	00h	Position drive absolute mode
POS_REL_MODE	01h	Position drive relative mode

AXT_MOTION_PROFILE_MODE_DEF

#define	Value	Explanation
SYM_TRAPEZOIDE_MODE	00h	Symmetrical Trapezode
ASYM_TRAPEZOIDE_MODE	01h	Asymmetrical Trapezode
QUASI_S_CURVE_MODE	02h	Symmetrical Quasi-S Curve
SYM_S_CURVE_MODE	03h	Symmetrical S Curve
ASYM_S_CURVE_MODE	04h	Asymmetrical S Curve

AXT_MOTION_SELECTION_DEF

#define	Value	Explanation
INACTIVE	00h	No-activation
ACTIVE	01h	Activation

AXT_MOTION_HOME_RESULT_DEF

#define	Value	Explanation
HOME_SUCCESS	00h	If home search is exited successfully
HOME_SEARCHING	02h	If home search is in progress currently
HOME_ERR_GNT_RANGE	10h	If home search result of master and slave axis on gantry drive axis is off the set OffsetRange
HOME_ERR_USER_BREAK	11h	If user executes a stop command during home search
HOME_ERR_VELOCITY	12h	If home search velocity is not set
HOME_ERR_AMP_FAULT	13h	If servo pack alarm occurred during home search
HOME_ERR_NEG_LIMIT	14h	If (-) limit sensor is detected during home sensor search with (+) direction
HOME_ERR_POS_LIMIT	15h	If (+) limit sensor is detected during home sensor search with (-) direction
HOME_ERR_NOT_DETECT	16h	If home sensor is not detected
HOME_ERR_UNKNOWN	FFh	If tried home search from unknown channel (axis) number (starting from 0)

AXT_MOTION_UNIV_INPUT_DEF

#define	Value	Explanation

UIO_INP0	00h	Universal input 0
UIO_INP1	01h	Universal input 1
UIO_INP2	02h	Universal input 2
UIO_INP3	03h	Universal input 3
UIO_INP4	04h	Universal input 4

AXT_MOTION_UNIV_OUTPUT_DEF

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

AXT_MOTION_DETECT_DOWN_START_POINT_DEF

Assignment of deceleration start point search method

#define	Value	Explanation
AutoDetect	00h	Automatic search method
RestPulse	01h	Remaining search method (manual acceleration and deceleration)

AXT_MOTION_PULSE_OUTPUT_DEF

#define	Value	Explanation
OneHighLowHigh	00h	1 pulse method, PULSE(Active High), Forward direction (DIR=Low) / Reverse direction (DIR=High)
OneHighHighLow	01h	1 pulse method, PULSE(Active High), Forward direction (DIR=High) / Reverse direction (DIR=Low)
OneLowLowHigh	02h	1 pulse method, PULSE(Active Low), Forward direction (DIR=Low) / Reverse direction (DIR=High)
OneLowHighLow	03h	1 pulse method, PULSE(Active Low), Forward direction (DIR=High) / Reverse direction (DIR=Low)
TwoCcwCwHigh	04h	2 pulse method, PULSE(CCW: Reverse direction), DIR(CW: Forward direction), Active High
TwoCcwCwLow	05h	2 pulse method, PULSE(CCW: Reverse direction), DIR(CW: Forward direction), Active Low
TwoCwCcwHigh	06h	2 pulse method, PULSE(CW: Forward direction), DIR(CCW: Reverse direction), Active High
TwoCwCcwLow	07h	2 pulse method, PULSE(CW: Forward direction),

		DIR(CCW: Reverse direction), Active Low
TwoPhase	08h	2 phase(90' phase difference), PULSE lead DIR(CW: Forward direktion), PULSE lag DIR(CCW: Reverse direction)
TwoPhaseReverse	09h	2 phase(90' phase difference), PULSE lead DIR(CCW: Forward direktion), PULSE lag DIR(CW: Reverse direction)

AXT_MOTION_EXTERNAL_COUNTER_INPUT_DEF

Encoder Input Method

#define	Value	Explanation
ObverseUpDownMode	00h	Forward direction Up / Down
ObverseSqr1Mode	01h	Forward direction 1 multiplication
ObverseSqr2Mode	02h	Forward direction 2 multiplication
ObverseSqr4Mode	03h	Forward direction 4 multiplication
ReverseUpDownMode	04h	Reverse direction Up / Down
ReverseSqr1Mode	05h	Reverse direction 1 multiplication
ReverseSqr2Mode	06h	Reverse direction 2 multiplication
ReverseSqr4Mode	07h	Reverse direction 4 multiplication

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	Clockwise direction
DIR_CW	01h	Counterclockwise direction

AXT_MOTION_RADIUS_DISTANCE_DEF

#define	Value	Explanation
SHORT_DISTANCE	00h	Short distance circular movement
LONG_DISTANCE	01h	Lone distance circular movement

AXT_MOTION_INTERPOLATION_AXIS_DEF

#define	Value	Explanation
INTERPOLATION_AXIS2	02h	When 2 axis is used for interpolation
INTERPOLATION_AXIS3	03h	When 3 axis is used for interpolation

INTERPOLATION_AXIS4	04h	When 4 axis is used for interpolation
---------------------	-----	---------------------------------------

AXT_MOTION_CONTISTART_NODE_DEF

#define	Value	Explanation
CONTI_NODE_VELOCITY	00h	Velocity assigned interpolation mode
CONTI_NODE_MANUAL	01h	Node acceleration and deceleration interpolation mode
CONTI_NODE_AUTO	02h	Automatic acceleration and deceleration interpolation mode

AXT_MOTION_HOME_DETECT_SIGNAL_DEF

#define	Value	Explanation
PosEndLimit	00h	+Elm(End limit) + direction limit sensor signal
NegEndLimit	01h	-Elm(End limit) - direction limit sensor signal
PosSloLimit	02h	+Slm(Decelerate limit) signal – Not-used
NegSloLimit	03h	-Slm(Decelerate limit) signal – Not-used
HomeSensor	04h	IN0(ORG) home sensor signal
EncodZPhase	05h	IN1(Z phase) Encoder Z phase signal
Unilnput02	06h	IN2(universal) universal input number 2 signal
Unilnput03	07h	IN3(universal) universal input number 3 signal

AXT_MOTION_IPDETECT_DESTINATION_SIGNAL

#define	Value	Explanation
PElmNegativeEdge	00h	+Elm(End limit) falling edge
NElmNegativeEdge	01h	-Elm(End limit) falling edge
PSlmNegativeEdge	02h	+Slm(Decelerate limit) signal – Not-used
NSlmNegativeEdge	03h	-Slm(Decelerate limit) signal – Not-used
In0DownEdge	04h	IN0(ORG) falling edge
In1DownEdge	05h	IN1(Z phase) falling edge
In2DownEdge	06h	IN2(universal) falling edge
In3DownEdge	07h	IN3(universal) falling edge
PElmPositiveEdge	00h	+Elm(End limit) rising edge
NElmPositiveEdge	01h	-Elm(End limit) rising edge
PSlmPositiveEdge	02h	+Slm(Decelerate limit) signal – Not-used
NSlmPositiveEdge	03h	-Slm(Decelerate limit) signal – Not-used
In0UpEdge	04h	IN0(ORG) rising edge
In1UpEdge	05h	IN1(Z phase) rising edge
In2UpEdge	06h	IN2(universal) rising edge
In3UpEdge	07h	IN3(universal) rising edge

AXT_MOTION_QIDECTECT_DESTINATION_SIGNAL

#define	Value	Explanation
Signal_PosEndLimit	00h	+Elm(End limit) + direction limit sensor signal
Signal_NegEndLimit	01h	-Elm(End limit) - direction limit sensor signal
Signal_PosSloLimit	02h	+Slm(Decelerate limit) signal – Not-used
Signal_NegSloLimit	03h	-Slm(Decelerate limit) signal – Not-used
Signal_HomeSensor	04h	IN0(ORG) home sensor signal
Signal_EncodZPhase	05h	IN1(Z phase) Encoder Z phase signal
Signal_Unilnput02	06h	IN2(universal) universal input number 2 signal
Signal_Unilnput03	07h	IN3(universal) universal input number 3 signal

AXT_MOTION MPG_INPUT_METHOD

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

AXT_MOTION_SENSOR_INPUT_METHOD

#define	Value	Explanation
SENSOR_METHOD1	00h	General drive
SENSOR_METHOD2	01h	Slow speed drive before sensor signal detection. General drive after signal detection
SENSOR_METHOD3	02h	Slow speed drive

AXT_MOTION_HOME_CRC_SELECT

#define	Value	Explanation
CRC_SELECT1	00h	No use of position clear, no use of remaining pulse clear
CRC_SELECT2	01h	Use of position clear, no use of remaining pulse clear
CRC_SELECT3	02h	No use of position clear, use of remaining pulse clear
CRC_SELECT4	03h	Use of position clear, use of remaining pulse clear

SMC-2V03 exclusive use

AXT_MOTION_IPMECHANICAL_SIGNAL_DEF

Mechanical Signal register (SMC-2V03)

#define	Value	Explanation
IPMECHANICAL_PELM_LEVEL	0001h	Bit 0, +Limit emergency stop signal input Level
IPMECHANICAL_NELM_LEVEL	0002h	Bit 1, -Limit emergency stop signal input Level
IPMECHANICAL_PSLM_LEVEL	0004h	Bit 2, +limit decelerate stop signal input Level
IPMECHANICAL_NSLSM_LEVEL	0008h	Bit 3, -limit decelerate stop signal input Level
IPMECHANICAL_ALARM_LEVEL	0010h	Bit 4, Alarm signal input Level
IPMECHANICAL_INP_LEVEL	0020h	Bit 5, InPos signal input Level
IPMECHANICAL_ENC_DOWN_LEVEL	0040h	Bit 6, encoder DOWN(B phase) signal input Level
IPMECHANICAL_ENC_UP_LEVEL	0080h	Bit 7, encoder UP(A phase) signal input Level
IPMECHANICAL_EXMP_LEVEL	0100h	Bit 8, EXMP signal input Level
IPMECHANICAL_EXPP_LEVEL	0200h	Bit 9, EXPP signal input Level
IPMECHANICAL_MARK_LEVEL	0400h	Bit 10, MARK# signal input Level
IPMECHANICAL_SSTOP_LEVEL	0800h	Bit 11, SSTOP signal input Level
IPMECHANICAL_ESTOP_LEVEL	1000h	Bit 12, ESTOP signal input Level
IPMECHANICAL_SYNC_LEVEL	2000h	Bit 13, SYNC signal input Level
IPMECHANICAL_MODE8_16_LEVEL	4000h	Bit 14, MODE8_16 signal input Level

AXT_MOTION_IPEND_STATUS_DEF

Exit state of axis (SMC-2V03)

#define	Value	Explanation
	0000h	No drive or initial state
IPEND_STATUS_SLM	0001h	Bit 0, exit by limit decelerate stop signal input
IPEND_STATUS_ELM	0002h	Bit 1, exit by limit emergency stop signal input
IPEND_STATUS_SSTOP_SIGNAL	0004h	Bit 2, exit by decelerate stop signal input
IPEND_STATUS_ESTOP_SIGNAL	0008h	Bit 3, exit by emergency stop signal input
IPEND_STATUS_SSTOP_COMMAND	0010h	Bit 4, exit by decelerate stop command
IPEND_STATUS_ESTOP_COMMAND	0020h	Bit 5, exit by emergency stop command
IPEND_STATUS_ALARM_SIGNAL	0040h	Bit 6, exit by Alarm signal input
IPEND_STATUS_DATA_ERROR	0080h	Bit 7, exit by data setting error
IPEND_STATUS_DEVIATION_ERROR	0100h	Bit 8, exit by deviation error
IPEND_STATUS_ORIGIN_DETECT	0200h	Bit 9, exit by home search
IPEND_STATUS_SIGNAL_DETECT	0400h	Bit 10, exit by signal search

		(Signal search-1/2 drive exit)
IPEND_STATUS_PRESET_PULSE_DRIVE	0800h	Bit 11, Preset pulse drive exit
IPEND_STATUS_SENSOR_PULSE_DRIVE	1000h	Bit 12, Sensor pulse drive exit
IPEND_STATUS_LIMIT	2000h	Bit 13, exit by Limit complete stop
IPEND_STATUS_SOFTLIMIT	4000h	Bit 14, exit by Soft limit
IPEND_STATUS_INTERPOLATION_DRIVE	8000h	Bit 15, Interpolation drive exit

AXT_MOTION_IPDRIVE_STATUS_DEF

Drive state of axis (SMC-2V03)

#define	Value	Explanation
IPDRIVE_STATUS_BUSY	00000001h	Bit 0, BUSY (in DRIVE)
IPDRIVE_STATUS_DOWN	00000002h	Bit 1, DOWN (in deceleration DRIVE)
IPDRIVE_STATUS_CONST	00000004h	Bit 2, CONST (in constant speed DRIVE)
IPDRIVE_STATUS_UP	00000008h	Bit 3, Up (in acceleration DRIVE)
IPDRIVE_STATUS_ICL	00000010h	Bit 4, ICL (ICM < INCNT)
IPDRIVE_STATUS_ICG	00000020h	Bit 5, ICG (ICM < INCNT)
IPDRIVE_STATUS_ECL	00000040h	Bit 6, ECL (ECM > EXCNT)
IPDRIVE_STATUS_ECG	00000080h	Bit 7, ECG (ECM < EXCNT)
IPDRIVE_STATUS_DRIVE_DIRECTION	00000100h	Bit 8, drive direction signal (0=CW/1=CCW)
IPDRIVE_STATUS_COMMAND_BUSY	00000200h	In execution of command
IPDRIVE_STATUS_PRESET_DRIVING	00000400h	In drive of a specific pulse number
IPDRIVE_STATUS_CONTINUOUS_DRIVING	00000800h	In continuous drive
IPDRIVE_STATUS_SIGNAL_SEARCH_DRIVING	00001000h	In signal search drive
IPDRIVE_STATUS_ORG_SEARCH_DRIVING	00002000h	In home search drive
IPDRIVE_STATUS MPG_DRIVING	00004000h	In MPG drive
IPDRIVE_STATUS_SENSOR_DRIVING	00008000h	In sensor position drive
IPDRIVE_STATUS_L_C_INTERPOLATION	00010000h	In straight/circular interpolation drive
IPDRIVE_STATUS_PATTERN_INTERP	00020000h	In pattern interpolation drive

OLATION		
IPDRIVE_STATUS_INTERRUPT_BANK1	00040000h	Interrupt occurrence in BANK1
IPDRIVE_STATUS_INTERRUPT_BANK2	00080000h	Interrupt occurrence in BANK2

AXT_MOTION_IPINTERRUPT_BANK1_DEF

#define	Value	Explanation
IPINTBANK1_DONTUSE	00000000h	INTERRUT DISABLED
IPINTBANK1_DRIVE_END	00000001h	Bit 0, Drive end(default value : 1).
IPINTBANK1_ICG	00000002h	Bit 1, INCNT is greater than INCNTCMP.
IPINTBANK1_ICE	00000004h	Bit 2, INCNT is equal with INCNTCMP.
IPINTBANK1_ICL	00000008h	Bit 3, INCNT is less than INCNTCMP.
IPINTBANK1_ECG	00000010h	Bit 4, EXCNT is greater than EXCNTCMP.
IPINTBANK1_ECE	00000020h	Bit 5, EXCNT is equal with EXCNTCMP.
IPINTBANK1_ECL	00000040h	Bit 6, EXCNT is less than EXCNTCMP.
IPINTBANK1_SCRQEMPTY	00000080h	Bit 7, Script control queue is empty.
IPINTBANK1_CAPRQEMPTY	00000100h	Bit 8, Caption result data queue is empty.
IPINTBANK1_SCRREG1EXE	00000200h	Bit 9, Script control register-1 Cmd is executed.
IPINTBANK1_SCRREG2EXE	00000400h	Bit 10, Script control register-2 Cmd is executed.
IPINTBANK1_SCRREG3EXE	00000800h	Bit 11, Script control register-3 Cmd is executed.
IPINTBANK1_CAPREG1EXE	00001000h	Bit 12, Caption control register-1 Cmd is executed.
IPINTBANK1_CAPREG2EXE	00002000h	Bit 13, Caption control register-2 Cmd is executed.
IPINTBANK1_CAPREG3EXE	00004000h	Bit 14, Caption control register-3 Cmd is executed.
IPINTBANK1_INTGGENCMD	00008000h	Bit 15, Interrupt generation Cmd is executed(0xFF)
IPINTBANK1_DOWN	00010000h	Bit 16, At starting point for deceleration drive.
IPINTBANK1_CONT	00020000h	Bit 17, At starting point for constant speed drive.
IPINTBANK1_UP	00040000h	Bit 18, At starting point for acceleration drive.

IPINTBANK1_SIGNALDETECTED	00080000h	Bit 19, Signal assigned in MODE1 is detected.
IPINTBANK1_SP23E	00100000h	Bit 20, Current speed is equal with rate change point RCP23.
IPINTBANK1_SP12E	00200000h	Bit 21, Current speed is equal with rate change point RCP12.
IPINTBANK1_SPE	00400000h	Bit 22, Current speed is equal with speed comparison data(SPDCMP).
IPINTBANK1_INCEICM	00800000h	Bit 23, INTCNT(1'st counter) is equal with ICM(1'st count minus limit data)
IPINTBANK1_SCRQEXE	01000000h	Bit 24, Script queue Cmd is executed When SCRCONQ's 30 bit is '1'.
IPINTBANK1_CAPQEXE	02000000h	Bit 25, Caption queue Cmd is executed When CAPCONQ's 30 bit is '1'.
IPINTBANK1_SLM	04000000h	Bit 26, NSLM/PSLM input signal is activated.
IPINTBANK1_ELM	08000000h	Bit 27, NELM/PELM input signal is activated.
IPINTBANK1_USERDEFINE1	10000000h	Bit 28, Selectable interrupt source 0(refer "0xFE" Cmd).
IPINTBANK1_USERDEFINE2	20000000h	Bit 29, Selectable interrupt source 1(refer "0xFE" Cmd).
IPINTBANK1_USERDEFINE3	40000000h	Bit 30, Selectable interrupt source 2(refer "0xFE" Cmd).
IPINTBANK1_USERDEFINE4	80000000h	Bit 31, Selectable interrupt source 3(refer "0xFE" Cmd).

AXT_MOTION_IPINTERRUPT_BANK2_DEF

#define	Value	Explanation
IPINTBANK2_DONTUSE	00000000h	INTERRUPT DISABLED
IPINTBANK2_L_C_INP_Q_EMPTY	00000001h	Bit 0, Linear/Circular interpolation parameter queue is empty.
IPINTBANK2_P_INP_Q_EMPTY	00000002h	Bit 1, Bit pattern interpolation queue is empty.
IPINTBANK2_ALARM_ERROR	00000004h	Bit 2, Alarm input signal is activated.
IPINTBANK2_INPOS	00000008h	Bit 3, InPos input signal is activated.
IPINTBANK2_MARK_SIGNAL_HIGH	00000010h	Bit 4, Mark input signal is activated.
IPINTBANK2_SSTOP_SIGNAL	00000020h	Bit 5, SSTOP input signal is activated.
IPINTBANK2_ESTOP_SIGNAL	00000040h	Bit 6, ESTOP input signal is activated.
IPINTBANK2_SYNC_ACTIVATED	00000080h	Bit 7, SYNC input signal is activated.

IPINTBANK2_TRIGGER_ENABLE	00000100h	Bit 8, Trigger output is activated.
IPINTBANK2_EXCNTCLR	00000200h	Bit 9, External(2'nd) counter is cleared by EXCNTCLR setting.
IPINTBANK2_FSTCOMPARE_RESULT_BIT0	00000400h	Bit 10, ALU1's compare result bit 0 is activated.
IPINTBANK2_FSTCOMPARE_RESULT_BIT1	00000800h	Bit 11, ALU1's compare result bit 1 is activated.
IPINTBANK2_FSTCOMPARE_RESULT_BIT2	00001000h	Bit 12, ALU1's compare result bit 2 is activated.
IPINTBANK2_FSTCOMPARE_RESULT_BIT3	00002000h	Bit 13, ALU1's compare result bit 3 is activated.
IPINTBANK2_FSTCOMPARE_RESULT_BIT4	00004000h	Bit 14, ALU1's compare result bit 4 is activated.
IPINTBANK2_SNDCOMPARE_RESULT_BIT0	00008000h	Bit 15, ALU2's compare result bit 0 is activated.
IPINTBANK2_SNDCOMPARE_RESULT_BIT1	00010000h	Bit 16, ALU2's compare result bit 1 is activated.
IPINTBANK2_SNDCOMPARE_RESULT_BIT2	00020000h	Bit 17, ALU2's compare result bit 2 is activated.
IPINTBANK2_SNDCOMPARE_RESULT_BIT3	00040000h	Bit 18, ALU2's compare result bit 3 is activated.
IPINTBANK2_SNDCOMPARE_RESULT_BIT4	00080000h	Bit 19, ALU2's compare result bit 4 is activated.
IPINTBANK2_L_C_INP_Q_LESS_4	00100000h	Bit 20, Linear/Circular interpolation parameter queue is less than 4.
IPINTBANK2_P_INP_Q_LESS_4	00200000h	Bit 21, Pattern interpolation parameter queue is less than 4
IPINTBANK2_XSYNC_ACTIVATED	00400000h	Bit 22, Axis X sync input signal is activated.
IPINTBANK2_YSYNC_ACTIVATED	00800000h	Bit 23, Axis Y sync input signal is activated.
IPINTBANK2_P_INP_END_BY_END_PATTERN	01000000h	Bit 24, Bit pattern interpolation is terminated by end pattern.

**PCIN404,PCIN804 (PCI-N804/404) exclusive use
AXT_MOTION_QIMECHANICAL_SIGNAL_DEF**

Mechanical Signal register (PCI-N804/404)

#define	Value	Explanation
QIMECHANICAL_PELM_LEVEL	0001h	Bit 0, +Limit emergency stop signal current

		state
QIMECHANICAL_NELM_LEVEL	0002h	Bit 1, -Limit emergency stop signal current state
QIMECHANICAL_PSLM_LEVEL	0004h	Bit 2, +limit decelerate stop signal current state.
QIMECHANICAL_NSML_LEVEL	0008h	Bit 3, -limit decelerate stop signal current state
QIMECHANICAL_ALARM_LEVEL	0010h	Bit 4, Alarm signal current state
QIMECHANICAL_INP_LEVEL	0020h	Bit 5, InPos signal current state
QIMECHANICAL_ESTOP_LEVEL	0040h	Bit 6, emergency stop signal(ESTOP) current state
QIMECHANICAL_ORG_LEVEL	0080h	Bit 7, home signal current state
QIMECHANICAL_ZPHASE_LEVEL	0100h	Bit 8, Z phase input signal current state
QIMECHANICAL_ECUP_LEVEL	0200h	Bit 9, ECUP terminal signal state.
QIMECHANICAL_ECDN_LEVEL	0400h	Bit 10, ECDN terminal signal state.
QIMECHANICAL_EXPP_LEVEL	0800h	Bit 11, EXPP terminal signal state.
QIMECHANICAL_EXMP_LEVEL	1000h	Bit 12, EXMP terminal signal state.
QIMECHANICAL_SQSTR1_LEVEL	2000h	Bit 13, SQSTR1 terminal signal state.
QIMECHANICAL_SQSTR2_LEVEL	4000h	Bit 14, SQSTR2 terminal signal state.
QIMECHANICAL_SQSTP1_LEVEL	8000h	Bit 15, SQSTP1 terminal signal state.
QIMECHANICAL_SQSTP2_LEVEL	10000h	Bit 16, SQSTP2 terminal signal state.
QIMECHANICAL_MODE_LEVEL	20000h	Bit 17, MODE terminal signal state.

AXT_MOTION_QIEND_STATUS_DEF

Exit state of axis (PCI-N804/404)

#define	Value	Explanation
	00000000h	No drive or initial state
QIEND_STATUS_0	00000001h	Bit 0, exit by forward direction limit signal (PELM)
QIEND_STATUS_1	00000002h	Bit 1, exit by reverse direction limit signal (NELM)
QIEND_STATUS_2	00000004h	Bit 2, exit by forward direction additional limit signal (PSLM)
QIEND_STATUS_3	00000008h	Bit 3, exit by reverse direction additional limit signal (NSLM)
QIEND_STATUS_4	00000010h	Bit 4, exit by forward direction soft limit emergency stop function

QIEND_STATUS_5	00000020h	Bit 5, exit by reverse direction soft limit emergency stop function
QIEND_STATUS_6	00000040h	Bit 6, exit by forward direction soft limit decelerate stop function
QIEND_STATUS_7	00000080h	Bit 7, exit by reverse direction soft limit decelerate stop function
QIEND_STATUS_8	00000100h	Bit 8, drive exit by servo alarm function
QIEND_STATUS_9	00000200h	Bit 9, drive exit by emergency stop signal input
QIEND_STATUS_10	00000400h	Bit 10, drive exit by emergency stop command
QIEND_STATUS_11	00000800h	Bit 11, drive exit by decelerate stop command.
QIEND_STATUS_12	00001000h	Bit 12, drive exit by entire axes emergency stop command
QIEND_STATUS_13	00002000h	Bit 13, drive exit by sync stop function #1(SQSTP1)
QIEND_STATUS_14	00004000h	Bit 14, drive exit by sync stop function #2(SQSTP2).
QIEND_STATUS_15	00008000h	Bit 15, encoder input (ECUP,ECDN) error occurrence
QIEND_STATUS_16	00010000h	Bit 16, MPG input (EXPP,EXMP) error occurrence
QIEND_STATUS_17	00020000h	Bit 17, exit with successful home search
QIEND_STATUS_18	00040000h	Bit 18, exit with successful signal search
QIEND_STATUS_19	00080000h	Bit 19, drive exit by interpolation data abnormality.
QIEND_STATUS_20	00100000h	Bit 20, abnormal drive stop occurrence.
QIEND_STATUS_21	00200000h	Bit 21, MPG function block pulse buffer overflow occurrence
QIEND_STATUS_22 – 27		Bit 22,- Bit27 DON'CARE
QIEND_STATUS_28	10000000h	Bit 28, current/last move drive direction
QIEND_STATUS_29	20000000h	Bit 29, in output of remaining pulse clear

QIEND_STATUS_30	40000000h	Bit 30, abnormal drive stop cause state
QIEND_STATUS_31	80000000h	Bit 31, interpolation drive data error state.

AXT_MOTION_QIDRIVE_STATUS_DEF

Drive state of axis (PCI-N804/404)

#define	Value	Explanation
QIDRIVE_STATUS_0	00000001h	Bit 0, BUSY(in drive move)
QIDRIVE_STATUS_1	00000002h	Bit 1, DOWN(in deceleration)
QIDRIVE_STATUS_2	00000004h	Bit 2, CONST(in constant speed)
QIDRIVE_STATUS_3	00000008h	Bit 3, UP(in acceleration)
QIDRIVE_STATUS_4	00000010h	Bit 4, in move of continuous drive
QIDRIVE_STATUS_5	00000020h	Bit 5, in move of assigned distance drive
QIDRIVE_STATUS_6	00000040h	Bit 6, in move of MPG drive
QIDRIVE_STATUS_7	00000080h	Bit 7, in move of home search drive
QIDRIVE_STATUS_8	00000100h	Bit 8, in move of signal search drive
QIDRIVE_STATUS_9	00000200h	Bit 9, in move of interpolation drive
QIDRIVE_STATUS_10	00000400h	Bit 10, in move of Slave drive
QIDRIVE_STATUS_11	00000800h	Bit 11, current move drive direction (different indication information in interpolation drive)
QIDRIVE_STATUS_12	00001000h	Bit 12, in waiting for servo position completion after pulse output.
QIDRIVE_STATUS_13	00002000h	Bit 13, in move of straight line interpolation drive.
QIDRIVE_STATUS_14	00004000h	Bit 14, in move of circular interpolation drive.
QIDRIVE_STATUS_15	00008000h	Bit 15, in pulse output.
QIDRIVE_STATUS_16	00010000h	Bit 16, drive reserved data number(start)(0~7)
QIDRIVE_STATUS_17	00020000h	Bit 17, drive reserved data number (middle)(0~7)
QIDRIVE_STATUS_18	00040000h	Bit 18, drive reserved data number (end)(0~7)
QIDRIVE_STATUS_19	00080000h	Bit 19, drive reversed Queue is cleared
QIDRIVE_STATUS_20	00100000h	Bit 20, drive reversed Queue is full
QIDRIVE_STATUS_21	00200000h	Bit 21, velocity mode of current move drive (start)
QIDRIVE_STATUS_22	00400000h	Bit 22, velocity mode of current move drive (end)
QIDRIVE_STATUS_23	00800000h	Bit 23, MPG buffer #1 Full
QIDRIVE_STATUS_24	01000000h	Bit 24, MPG buffer #2 Full

QIDRIVE_STATUS_25	02000000h	Bit 25, MPG buffer #3 Full
QIDRIVE_STATUS_26	04000000h	Bit 26, MPG buffer data OverFlow

AXT_MOTION_QIINTERRUPT_BANK1_DEF

#define	Value	Explanation
QIINTBANK1_DISABLE	00000000h	INTERRUT DISABLED
QIINTBANK1_0	00000001h	Bit 0, when drive set interrupt occurrence use is exit
QIINTBANK1_1	00000002h	Bit 1, when drive is exited
QIINTBANK1_2	00000004h	Bit 2, when drive is started
QIINTBANK1_3	00000008h	Bit 3, counter #1 < comparator #1 event occurrence
QIINTBANK1_4	00000010h	Bit 4, counter #1 = comparator #1 event occurrence
QIINTBANK1_5	00000020h	Bit 5, counter #1 > comparator #1 event occurrence
QIINTBANK1_6	00000040h	Bit 6, counter #2 < comparator #2 event occurrence
QIINTBANK1_7	00000080h	Bit 7, counter #2 = comparator #2 event occurrence
QIINTBANK1_8	00000100h	Bit 8, counter #2 > comparator #2 event occurrence
QIINTBANK1_9	00000200h	Bit 9, counter #3 < comparator #3 event occurrence
QIINTBANK1_10	00000400h	Bit 10, counter #3 = comparator #3 event occurrence
QIINTBANK1_11	00000800h	Bit 11, counter #3 > comparator #3 event occurrence
QIINTBANK1_12	00001000h	Bit 12, counter #4 < comparator #4 event occurrence
QIINTBANK1_13	00002000h	Bit 13, counter #4 = comparator #4 event occurrence
QIINTBANK1_14	00004000h	Bit 14, counter #4 < comparator #4 event occurrence
QIINTBANK1_15	00008000h	Bit 15, counter #5 < comparator #5 event occurrence

QIINTBANK1_16	00010000h	Bit 16, counter #5 = comparator #5 event occurrence
QIINTBANK1_17	00020000h	Bit 17, counter #5 > comparator #5 event occurrence
QIINTBANK1_18	00040000h	Bit 18, timer #1 event occurrence
QIINTBANK1_19	00080000h	Bit 19, timer #2 event occurrence.
QIINTBANK1_20	00100000h	Bit 20, drive reservation setting Queue is cleared.
QIINTBANK1_21	00200000h	Bit 21, drive reservation setting Queue is full
QIINTBANK1_22	00400000h	Bit 22, trigger occurring distance period/absolute position Queue is cleared.
QIINTBANK1_23	00800000h	Bit 23, trigger occurring distance period/absolute position Queue is full
QIINTBANK1_24	01000000h	Bit 24, trigger signal occurrence event
QIINTBANK1_25	02000000h	Bit 25, script #1 command reservation setting Queue is cleared.
QIINTBANK1_26	04000000h	Bit 26, script #2 command reservation setting Queue is cleared.
QIINTBANK1_27	08000000h	Bit 27, script #3 initialized with execution of command reservation setting register.
QIINTBANK1_28	10000000h	Bit 28, script #4 initialized with execution of command reservation setting register.
QIINTBANK1_29	20000000h	Bit 29, servo alarm signal is permitted
QIINTBANK1_30	40000000h	Bit 30, $ CNT1 - CNT2 \geq CNT4 $ event occurrence.
QIINTBANK1_31	80000000h	Bit 31, interrupt occurrence command INTGEN execution.

AXT_MOTION_QIINTERRUPT_BANK2_DEF

#define	Value	Explanation
QIINTBANK2_DISABLE	00000000h	INTERRUT DISABLED
QIINTBANK2_0	00000001h	Bit 0, script #1 reading command result Queue is full.
QIINTBANK2_1	00000002h	Bit 1, script #2 reading command result Queue is full.
QIINTBANK2_2	00000004h	Bit 2, script #3 reading command result register is renewed with new data

QIINTBANK2_3	00000008h	Bit 3, script #4 reading command result register is renewed with new data
QIINTBANK2_4	00000010h	Bit 4, when reservation command of script #1 is executed, command set by interrupt occurrence is executed
QIINTBANK2_5	00000020h	Bit 5, when reservation command of script #2 is executed, command set by interrupt occurrence is executed.
QIINTBANK2_6	00000040h	Bit 6, when reservation command of script #3 is executed, command set by interrupt occurrence is executed.
QIINTBANK2_7	00000080h	Bit 7, when reservation command of script #4 is executed, command set by interrupt occurrence is executed.
QIINTBANK2_8	00000100h	Bit 8, start drive
QIINTBANK2_9	00000200h	Bit 9, drive using servo position determination completion(InPos) function, exit condition occurrence.
QIINTBANK2_10	00000400h	Bit 10, event selection #1 condition occurrence to use during event counter operation.
QIINTBANK2_11	00000800h	Bit 11, event selection #2 condition occurrence to use during event counter operation.
QIINTBANK2_12	00001000h	Bit 12, SQSTR1 signal is permitted.
QIINTBANK2_13	00002000h	Bit 13, SQSTR2 signal is permitted.
QIINTBANK2_14	00004000h	Bit 14, UIO0 terminal signal is changed to '1'.
QIINTBANK2_15	00008000h	Bit 15, UIO1 terminal signal is changed to '1'.
QIINTBANK2_16	00010000h	Bit 16, UIO2 terminal signal is changed to '1'.
QIINTBANK2_17	00020000h	Bit 17, UIO3 terminal signal is changed to '1'.
QIINTBANK2_18	00040000h	Bit 18, UIO4 terminal signal is changed to '1'.
QIINTBANK2_19	00080000h	Bit 19, UIO5 terminal signal is changed to '1'.
QIINTBANK2_20	00100000h	Bit 20, UIO6 terminal signal is changed to '1'.
QIINTBANK2_21	00200000h	Bit 21, UIO7 terminal signal is changed to '1'.
QIINTBANK2_22	00400000h	Bit 22, UIO8 terminal signal is changed to '1'.
QIINTBANK2_23	00800000h	Bit 23, UIO9 terminal signal is changed to '1'.
QIINTBANK2_24	01000000h	Bit 24, UIO10 terminal signal is changed to '1'.
QIINTBANK2_25	02000000h	Bit 25, UIO11 terminal signal is changed to '1'.

QIINTBANK2_26	04000000h	Bit 26, error stop condition (LMT, ESTOP, STOP, ESTOP, CMD, ALARM) occurrence.
QIINTBANK2_27	08000000h	Bit 27, data setting error is occurred during interpolation.
QIINTBANK2_28	10000000h	Bit 28, Don't Care
QIINTBANK2_29	20000000h	Bit 29, limit signal (PELM, NELM) is inputted.
QIINTBANK2_30	40000000h	Bit 30, additional limit signal (PSLM, NSLM) is inputted.
QIINTBANK2_31	80000000h	Bit 31, emergency stop signal (ESTOP) is input.

Motion Command Function List

[Board and Module Initialization](#)

[Virtual Axis API](#)

[Interrupt API](#)

[Motion Parameter Setting API](#)

[Input and Output Level Setting API](#)

[State Verification API after Motion Drive](#)

[Home Search API](#)

[Position Move API](#)

[Position and Velocity Override Drive API](#)

[MASTER, SLAVE Drive API](#)

[Interpolation Drive API \(Line, Circle\)](#)

[Continuous Interpolation Setting and Drive API](#)

[Trigger API](#)

[Gantry Drive API](#)

[CRC\(Remaining Pulse Clear\) API](#)

[MPG Drive API](#)

[Helical Interpolation Setting and Drive API \(PCI-N804/404 Exclusive Use API\)](#)

[Spline Interpolation Setting and Drive API \(PCI-N804/404 Exclusive Use API\)](#)

Board and Module Initialization

In this unit, APIs related to load or initialization required to initialize library will be introduced.

Function	Description
<u>AxmlInfoGetAxis</u>	Returns board number, module position and module ID of corresponding axis.
<u>AxmlInfoIsMotionModule</u>	Returns whether motion module exists.
<u>AxmlInfoIsInvalidAxisNo</u>	Returns whether corresponding axis is effective.
<u>AxmlInfoGetAxisCount</u>	Returns effective motion axis number installed in system.
<u>AxmlInfoGetFirstAxisNo</u>	Returns the first axis number of corresponding board and module.

AxmInfoGetAxis

Purpose

To verify board number, module position and module ID of a specific axis.

Format

C

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

Visual Basic

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

Delphi

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

Input / Output

Name	in/out	Format	Init Value	Explanation
AxisNo	in	long		Channel (axis) number(start from 0)
BoardNo	out	long *		Base board position value included setting axis
ModulePos	out	long*		Module position value included setting axis
ModuleID	out	DWORD*		Module ID value included setting axis

ModuleID

#define	Value	Explanation
AXT_SMC_2V03	05h	CAMC-IP, 2 Axis
AXT_SMC_4V04	06h	CAMC-QI, 4 Axis
AXT_SIO_DI32	97h	Digital IN 32 points
AXT_SIO_DO32P	98h	Digital OUT 32 points
AXT_SIO_DB32P	99h	Digital IN 16 points / OUT 16 points
AXT_SIO_DO32T	9Eh	Digital OUT 16 points, Power TR output
AXT_SIO_DB32T	9Fh	Digital IN 16 points / OUT 16 points, Power TR output
AXT_SIO_AI4RB	A1h	A1h(161) : AI 4Ch, 12 bit
AXT_SIO_AO4RB	A2h	A2h(162) : AO 4Ch, 12 bit
AXT_SIO_AI16H	A3h	A3h(163) : AI 4Ch, 16 bit
AXT_SIO_AO8H	A4h	A4h(164) : AO 4Ch, 16 bit
AXT_SIO_AI16HB	A5h	A5h(165) : AI 16Ch, 16 bit (SIO-AI16HR(input

		module))
AXT_SIO_AO2HB	A6h	A6h(166) : AO 2Ch, 16 bit (SIO-AI16HR(output module))

Return AXT_RT_SUCCESS(0000) : Successful execution of API.

[* See error code Table for more information on status error codes](#)

Description

Note : When only required data are needed to be called, it can be used with input of NULL on the unrequired variable.

Since board and module ID value are identical to product model number, it can be used to verify actual board and module information.

C Example

```
// Return board number, module position and module ID on axis 0.
Long BoardNo, ModulePos;
DWORD ModuleID;
AxmInfoGetAxis (0, &BoardNo, &ModulePos, &ModuleID);
```

VB Example

```
' Return board number, module position and module ID on axis 0.
Dim BoardNo , ModulePos, ModuleID As Long
AxmInfoGetAxis 0, BoardNo, ModulePos, ModuleID
```

Delphi Example

```
{Return board number, module position and module ID on axis 0. }
var
ModuleID: DWord;
BoardNo, ModulePos: LongInt;

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

See Also

[AxmInfoIsMotionModule](#), [AxmInfoIsInvalidAxisNo](#), [AxmInfoGetAxisCount](#), [AxmInfoGetFirstAxisNo](#)

AxmInfoIsMotionModule

Purpose

To return if motion module is installed.

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*		Value of whether motion module is installed or not

Status

#define	Value	Explanation
FALSE	00h	Motion module is not installed
TRUE	01h	Motion module is installed

Return AXT_RT_SUCCESS(0000) : Successful execution of API.

[* See error code Table for more information on status error codes](#)

Description

Whether motion module is installed or not returns to *upStatus value.

C Example

```
// Verify whether motion module is installed
DWORD Status;
AxmInfoIsMotionModule (&Status);
```

VB Example

```
' Verify whether motion module is installed
Dim Status As Long
AxmInfoIsMotionModule Status
```

Delphi Example

```
{ Verify whether motion module is installed }
var
Status: DWord;
begin
AxmInfoIsMotionModule (@Status);
end;
```

See Also

[AxmInfoGetAxis](#), [AxmInfoIsInvalidAxisNo](#), [AxmInfoGetAxisCount](#), [AxmInfoGetFirstAxisNo](#)

AxmlInfoIsInvalidAxisNo

Purpose

To return if corresponding axis is available to use.

Format

C

```
DWORD AxmlInfoIsInvalidAxisNo(long lAxisNo);
```

Visual Basic

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

Delphi

```
function AxmlInfoIsInvalidAxisNo (lAxisNo : LongInt) : DWord; stdcall;
```

Input / Output

Name	in/out	Format	Init Value	Explanation
AxisNo	in	long		Axis number to test if available to use

Return AXT_RT_SUCCESS(0000) : Successful execution of API.

[* See error code Table for more information on status error codes](#)

Description

Verify if corresponding axis can be used. If not, generate error code.

C Example

```
// Verify whether it can be used
DWORD dwReturn;
dwReturn = AxmlInfoIsInvalidAxisNo(9);
```

VB Example

```
' Verify whether it can be used
Dim dwReturn As Long
dwReturn = AxmlInfoIsInvalidAxisNo 9
```

Delphi Example

```
{ Verify whether it can be used }
var
dwReturn: DWord
Begin
dwReturn := AxmlInfoIsInvalidAxisNo(9);
end;
```

See Also

[AxmlInfoGetAxis](#), [AxmlInfoMotionModule](#), [AxmlInfoGetAxisCount](#), [AxmlInfoGetFirstAxisNo](#)

AxmInfoGetAxisCount

Purpose

To return total axis number of installed board.

Format

C

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

Visual Basic

```
Function AxmInfoGetAxisCount (ByRef IpAxisCount As Long) As Long
```

Delphi

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

Input / Output

Name	in/out	Format	Init Value	Explanation
AxisCount	out	long *		Total axis number of installed board

None

Return AXT_RT_SUCCESS(0000) : Successful execution of API.

[* See error code Table for more information on status error codes](#)

Description

Axis number of installed board returns to * lpAxisCount value.

C Example

```
// Verify axis number of installed motion module
long      AxisCount;
AxmInfoGetAxisCount (&AxisCount);
```

VB Example

```
' Verify axis number of installed motion module
Dim AxisCount As Long
AxmInfoGetAxisCount AxisCount
```

Delphi Example

```
{ Verify axis number of installed motion module }
var
AxisCount: LongInt;

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

See Also

[AxmInfoGetAxis](#), [AxmInfoIsMotionModule](#), [AxmInfoIsInvalidAxisNo](#), [AxmInfoGetFirstAxisNo](#)

AxmlInfoGetFirstAxisNo

Purpose

To return the earliest axis number in a specific position module of corresponding base board.

Format

C

```
DWORD AxmlInfoGetFirstAxisNo (long lBoardNo, long lModulePos, long *lpAxisNo);
```

Visual Basic

```
Function AxmlInfoGetFirstAxisNo (ByVal lBoardNo As Long, ByVal lModulePos As Long, ByRef lpAxisNo As Long) As Long
```

Delphi

```
function AxmlInfoGetFirstAxisNo (lBoardNo: LongInt; lModulePos : LongInt; lpAxisNo : PLongInt) : DWord; stdcall;
```

Input / Output

Name	in/out	Format	Init Value	Explanation
BoardNo	in	long		Setting base board number. Number is assigned in order of slot from 0.
ModulePos	In	long		Module number located within base board. In case of BPFR, module number from 0 to 3 are assigned, and in case of BPHR base board, module number from 0 to 1 are assigned.
AxisNo	out	long *		Axis number value of corresponding board/module

Return AXT_RT_SUCCESS(0000) : Successful execution of API.

[* See error code Table for more information on status error codes](#)

Description

Since the case of N804, N404 CAMC-QI is not the module type of board itself, return the first starting axis number on board.

Base board number is assigned in order of slot from 0. In case of BPFR, module number from 0 to 3 are assigned, and in case of BPHR base board, module number from 0 to 1 are assigned. Since motion module has 2 axes, *lpAxisNo value is to be the earliest axis number.

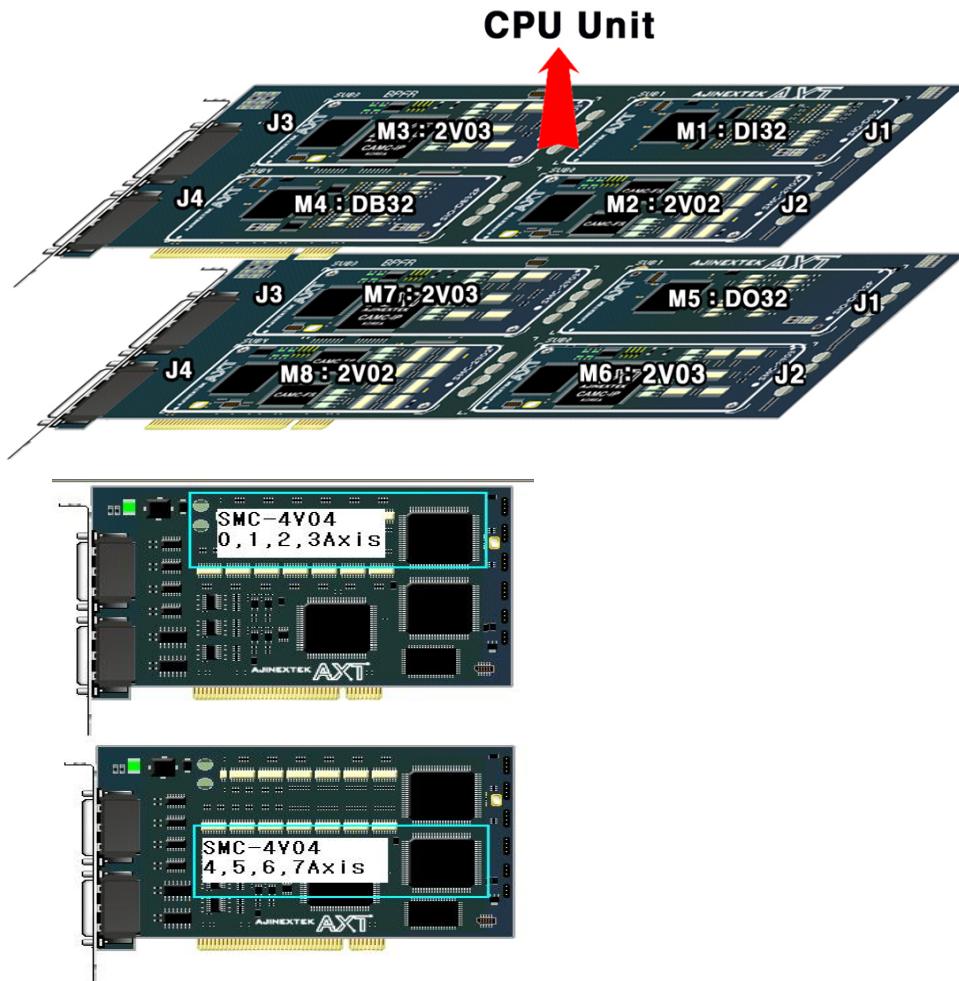
1. Numbering by same module is done individually.

AXL provides only version of N404, N804, and 2V03 module above.

2. If there are various base boards, numbering is done from the close base board to CPU.

In general PC or IPC system, order of PCI slot is set from the close one to CPU. However, based on system setting, there are systems that start number from a distant one on CPU module, or that set each slot number arbitrary.

3. Numbering is done in order of J1,J2,J3, and J4 on the same base board.



Assignment of axis number and channel number

Motion axis number connected to 2V03 module

Axis number	0	1	2	3	4	5
Corresponding module	M 3		M6		M7	

Motion axis number connected to 2V03 module

Channel Number	00	01	02	03	...	29	30	31	32	33	34	...	45	46	47
Corresponding module	M1						M4								

Digital Output channel number

Channel number	0	0	...	1	1	1	1	1	1	2	...	4	4	4	4
Corresponding module	M4			M5											

* Channel number of AIO module is determined in the same type with DIO.

C Example

```
// Verify the earliest axis number on a specific 0 module
long      AxisNo;
AxmInfoGetFirstAxisNo (0, 0, &AxisNo);
```

VB Example

```
' Verify the earliest axis number on a specific 0 module
Dim AxisNo As Long
AxmInfoGetFirstAxisNo 0, 0, AxisNo
```

Delphi Example

```
{ Verify the earliest axis number on a specific 0 module }
var
AxisNo: LongInt;

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

See Also

[AxmInfoGetAxis](#), [AxmInfoIsMotionModule](#), [AxmInfoIsInvalidAxisNo](#), [AxmInfoGetAxisCount](#)

Virtual Axis API

In this unit, when drive command is given from library, APIs that are to give command using arbitrary axis number not an actual corresponding axis number, will be introduced. For example, when given a command on axis 0, give a command as axis 2 during command. Let us suppose there are axes from 0 to 10 in the system, and axes 3, 4 in the middle are removed somehow. In this case, axis number will be run moved up from 0 to 8 in sequence. Therefore, axis numbers on the program have to be changed one by one, and in this case, it can be solved by mapping with original axis number without toughing program source. In this case, axes 3 to 8 just need to change to axes 5 to 10 .

Function	Description.
<u>AxmVirtualSetAxisNoMap</u>	Sets virtual axis.
<u>AxmVirtualGetAxisNoMap</u>	Returns virtual channel (axis) number which is set.
<u>AxmVirtualSetMultiAxisNoMap</u>	Sets multi–virtual axis.
<u>AxmVirtualGetMultiAxisNoMap</u>	Returns multi–virtual channel (axis) which is set.
<u>AxmVirtualResetAxisMap</u>	Clears virtual axis setting.

AxmVirtualSetAxisNoMap

Purpose

API that executes mapped actual axis number and virtual axis number.

Format

C

```
DWORD AxmVirtualSetAxisNoMap (long IRealAxisNo, long IVirtualAxisNo);
```

Visual Basic

```
Function AxmVirtualSetAxisNoMap (ByVal IRealAxisNo As Long, ByVal IVirtualAxisNo As Long ) As Long
```

Delphi

```
function AxmVirtualSetAxisNoMap (IRealAxisNo: LongInt ; IVirtualAxisNo : LongInt) : DWord; stdcall;
```

Input / Output

Name	in/out	Format	Init Value	Explanation
RealAxisNo	In	long		Actual axis number installed in system
VirtualAxisNo	In	long		Virtual axis number for mapping

Return AXT_RT_SUCCESS(0000) : Successful execution of API.

[* See error code Table for more information on status error codes](#)

Description

This function maps duplication of “actual” axes numbers into an array of virtual axes numbers.

After initialization, axes numbers in the range of 0 ~ (number of axis installed in the system – 1) are valid for all AXM functions, but arbitrary axes numbers can be used instead of actual installed axes numbers by using this function.

This function is made for the purpose of keeping the axes numbers assigned to the existing program unchanged and changing physical position of the actual control axes when changing the control system hardware.

Note: No more than one axis should be mapped to one virtual axis number. When there is an overlap in mapping, only the axis with lowest axis number can be controlled by the mapped virtual axis number, and other axes mapped to the same virtual axis cannot be controlled.

C Example

```
// Set actual axis 0 to virtual axis 5
AxmVirtualSetAxisNoMap (0, 5);

// Verify setting virtual axis
long      VirtualAxisNo;
AxmVirtualGetAxisNoMap (0, &VirtualAxisNo);
```

VB Example

```
'Set actual axis 0 to virtual axis 5
AxmVirtualSetAxisNoMap 0, 5

'Verify setting virtual axis
Dim VirtualAxisNo As Long
AxmVirtualGetAxisNoMap 0, VirtualAxisNo
```

Delphi Example

```
{ Verify setting virtual axis }
var
VirtualAxisNo: LongInt;

Begin
{ Set actual axis 0 to virtual 5 axis }
AxmVirtualSetAxisNoMap (0, 5);
AxmVirtualGetAxisNoMap (0, @VirtualAxisNo);
end;
;
```

See Also

[AxmVirtualGetAxisNoMap](#), [AxmVirtualSetMultiAxisNoMap](#), [AxmVirtualGetMultiAxisNoMap](#),
[AxmVirtualResetAxisMap](#)

AxmVirtualGetAxisNoMap

Purpose

API that verifies actual axis number and mapped virtual axis number.

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		Actual axis number installed in system
VirtualAxisNo	out	long *		Virtual axis number for mapping

Return AXT_RT_SUCCESS(0000) : Successful execution of API.

[* See error code Table for more information on status error codes](#)

Description

This function is to check the virtual axis number of the corresponding actual axis number mapped by the [AxmVirtualSetAxisNoMap](#) function. The return value will be -1 if the axis is not mapped, and the mapped virtual axis number will be returned when the axis is mapped.

C Example

```
// Set actual axis 0 to virtual axis 5
AxmVirtualSetAxisNoMap (0, 5);

// Verify setting virtual axis
long VirtualAxisNo;
AxmVirtualGetAxisNoMap (0, &VirtualAxisNo);
```

VB Example

```
' Set actual axis 0 to virtual axis 5
AxmVirtualSetAxisNoMap 0, 5

' Verify setting virtual axis
Dim VirtualAxisNo As Long
AxmVirtualGetAxisNoMap 0, VirtualAxisNo
```

Delphi Example

```
{ Verify setting virtual axis }
var
VirtualAxisNo: LongInt;

Begin
```

```
{ Set actual axis 0 to virtual axis 5 }
AxmVirtualSetAxisNoMap (0, 5);
AxmVirtualGetAxisNoMap (0, @VirtualAxisNo);
end;
```

See Also

[AxmVirtualSetAxisNoMap](#), [AxmVirtualGetAxisNoMap](#), [AxmVirtualSetMultiAxisNoMap](#),
[AxmVirtualGetMultiAxisNoMap](#), [AxmVirtualResetAxisMap](#)

AxmVirtualSetMultiAxisNoMap

Purpose

API that executes mapping more than one axis of actual axis number and mapped virtual axis number.

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		Axis number to set
RealAxisNo	In	long *		Actual axis number array installed in system
VirtualAxisNo	In	long *		Virtual axis number array for mapping

Return AXT_RT_SUCCESS(0000) : Successful execution of API.

[* See error code Table for more information on status error codes](#)

Description

This function maps actual axes numbers into an array of virtual axes numbers.

After initialization, axes numbers in the range of 0 ~ (number of axis installed in the system - 1) are valid for all AXM functions, but arbitrary axes numbers can be used instead of actual installed axes numbers by using this function.

This function is made for the purpose of keeping the axes numbers assigned to the existing program unchanged and changing physical position of the actual control axes when changing the control system hardware.

Note: No more than one axis should be mapped to one virtual axis number. When there is an overlap in mapping, only the axis with lowest axis number can be controlled by the mapped virtual axis number, and other axes mapped to the same virtual axis cannot be controlled.

C Example

```
// Set virtual axis
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;
// Set axis 0 to axis 4, axis 1 to axis 5, axis 2 to axis 6, and axis 3
// to axis 7.
AxmVirtualSetMultiAxisNoMap(4,Realaxis,Virtualaxis);

// Verify virtual axis setting
long lReadRealAxisNo[4];
AxmVirtualGetMultiAxisNoMap (4, Realaxis, lReadRealAxisNo);

```

VB Example

```

' Set virtual axis
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)

' Verify virtual axis setting
Dim RealAxisNo (0 To 3) As Long
AxmVirtualGetMultiAxisNoMap 4, VirtualAxisNo(0), RealAxisNo(0)

```

Delphi Example

```

{ Set virtual axis }
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;

  AxmVirtualSetMultiAxisNoMap (4, @Realaxis, @Virtualaxis);
  { Verify virtual axis setting }

  AxmVirtualGetMultiAxisNoMap (4, @Realaxis, @RealAxisNo);
end;

```

See Also

[AxmVirtualSetAxisNoMap](#), [AxmVirtualGetAxisNoMap](#), [AxmVirtualGetMultiAxisNoMap](#), [AxmVirtualResetAxisMap](#)

AxmVirtualGetMultiAxisNoMap

Purpose

API that verifies more than one axis of actual axis number and mapped virtual axis number.

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		Axis number to set
RealAxisNo	In	long *		Actual axis number array installed in system
VirtualAxisNo	Out	long *		Virtual axis number array for mapping

Return AXT_RT_SUCCESS(0000) : Successful execution of API.

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

Description

Verify various virtual axis number actually mapped using [AxmVirtualSetMultiAxisNoMap](#) .

C Example

```
// Set virtual axis
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;
// Set axis 0 to axis 4, axis 1 to axis 5, axis 2 to axis 6, and axis 3 to
// axis 7.
AxmVirtualSetMultiAxisNoMap (4,Realaxis,Virtualaxis);

// Verify virtual axis setting
long lReadRealAxisNo[4];
AxmVirtualGetMultiAxisNoMap (4, Realaxis, lReadRealAxisNo);
```

VB Example

```

'// Set virtual axis
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)

' Verify virtual axis setting
Dim RealAxisNo (0 To 3) As Long
AxmVirtualGetMultiAxisNoMap 4, Realaxis(0), RealAxisNo(0)

```

Delphi Example

```

{ Set virtual axis }
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;

  AxmVirtualSetMultiAxisNoMap (4, @Realaxis, @Virtualaxis);
  { Verify virtual axis setting }

  AxmVirtualGetMultiAxisNoMap (4, @Realaxis, @RealAxisNo);
end;

```

See Also

[AxmVirtualResetAxisMap](#), [AxmVirtualSetAxisNoMap](#), [AxmVirtualGetAxisNoMap](#), [AxmVirtualSetMultiAxisNoMap](#)

AxmVirtualResetAxisMap

Purpose

Clear virtual axis setting that is currently set.

Format

C

```
DWORD AxmVirtualResetAxisMap ();
```

Visual Basic

```
Function AxmVirtualResetAxisMap () As Long
```

Delphi

```
function AxmVirtualResetAxisMap () : DWord; stdcall;
```

Return AXT_RT_SUCCESS(0000) : Successful execution of API.

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

Description

If virtual axis setting is cleared, axis numbers used in API are to be same with axis number installed in actual system.

C Example

```
// Clear virtual axis setting  
AxmVirtualResetAxisMap ();
```

VB Example

```
' Clear virtual axis setting  
AxmVirtualResetAxisMap
```

Delphi Example

```
{ Clear virtual axis setting }  
begin  
AxmVirtualResetAxisMap ();  
End;
```

See Also

[AxmVirtualResetAxisMap](#), [AxmVirtualSetAxisNoMap](#), [AxmVirtualGetAxisNoMap](#), [AxmVirtualSetMultiAxisNoMap](#),
[AxmVirtualGetMultiAxisNoMap](#)

Interrupt API

In this unit, interrupt related APIs are introduced. Interrupt is to inform to user when a specific condition occurs. Unlike falling method, interrupt has an advantage that does not generate load on CPU. In the environment of window, interrupt is not managed at general Application level, and it informs by event.

For example, if a user wants to use interrupt for comparison function that compares deviation of Command and Actual counter, user can make to generate interrupt when greater deviation occurred than assigned value after deviation limit assignment if user sets interrupt. AXM can generate interrupt using various events as interrupt source occurring during motion drive, and if register of interrupt BANK composed of 31-bit generates each interrupt, it becomes TRUE. When interrupt occurs, to manage this, AXM provides 3 methods; callback API, window message, and event method. Especially, since event method uses a specified thread watching if interrupt occurred or uses while sentence, it takes more system resources, but has an advantage that manages interrupt most fast.

Function	Description
AxIInterruptEnable	In order to use the interrupt, other interrupt functions are to be used after calling this function..
AxIInterruptDisable	This function is used to disable the interrupt
AxMInterruptSetAxis	Uses window message or callback API to receive interrupt message.
AxMInterruptSetAxisEnable	Read internal flag variable which is set in ReadInterruptFlag.
AxMInterruptGetAxisEnable	Sets whether interrupt is used of motion module on corresponding axis
AxMInterruptRead	Returns whether interrupt is used of motion module on corresponding axis.
AxMInterruptReadAxisFlag	Returns interrupt flag value on corresponding axis.
AxMInterruptSetUserEnable	Sets whether interrupt set by user of a specific axis occurred.
AxMInterrupt GetUserEnable	Verifies whether interrupt set by user of a specific axis occurred.

AxIInterruptEnable

Purpose

To enable the on-board interrupt

Format

C

```
DWORD AxIInterruptEnable();
```

Visual Basic

```
Function AxIInterruptEnable () As Long
```

Delphi

```
function AxIInterruptEnable () : DWord; stdcall;
```

Return AXT_RT_SUCCESS(0000) : Successful execution of API.

[* See error code Table for more information on status error codes](#)

Description

This function must be called in order to use the on-board interrupt.

C Example

```
// Use window message to bring interrupt message
BEGIN_MESSAGE_MAP(CCAMCIPDlg, CDialog)
// } }AFX_MSG_MAP
// Declaration
ON_MESSAGE(WM_CAMCIP_INTERRUPT,OnTriggerInterrupt)
END_MESSAGE_MAP()

AxIInterruptEnable();
// When interrupt occurs, 'OnTriggerInterrupt()' is called
AxmInterruptSetAxis(0, m_hWnd, NULL, NULL);
```

Basic Example

```
'Use window message to bring interrupt message
' Declaration
Private Sub AxtMsg1_OnMessage1(ByVal wParam As Long, ByVal lParam As Long)
    ' Interrupt message process script
End Sub

    ' 'AxtMsg1_OnMessage1()' function will be called When interrupt occurs
AxtMsg1.Message1 = WM_CAMCIP_INTERRUPT

AxIInterruptEnable
    ' 'AxtMsg1_OnMessage1()' function will be called When interrupt occurs
AxmInterruptSetAxis 0, Me.hWnd, WM_CAMCIP_INTERRUPT, 0
```

Delphi Example

```
{ Use window message to bring interrupt message }
{ Declaration }
procedure TForm1.OnInterruptMessage(var Msg : TMessage);

begin
{ Interrupt message process script }
{ 'AxtMsg1_OnMessage1()' function will be called When interrupt occurs }
Ax1InterruptEnable();
AxmInterruptSetAxis(Form1.Handle, WM_CAMCIP_INTERRUPT, nil);
end;
```

See Also

[AxmInterruptSetAxisEnable](#), [AxmInterruptGetAxisEnable](#), [AxmInterruptRead](#), [AxmInterruptReadAxisFlag](#)

AxIInterruptDisable

Purpose

To disable the on-board interrupt

Format

C

```
DWORD AxIInterruptDisable();
```

Visual Basic

```
Function AxIInterruptDisable () As Long
```

Delphi

```
function AxIInterruptDisable () : DWord; stdcall;
```

Return AXT_RT_SUCCESS(0000) : Successful execution of API.

[* See error code Table for more information on status error codes](#)

Description

Call this function to disable the interrupt if the [AxIInterruptEnable](#) function was used to enable the interrupt.

C Example

```
// Disable after interrupt use.  
AxIInterruptDisable();
```

Basic Example

```
' Disable after interrupt use.  
AxIInterruptDisable
```

Delphi Example

```
{ Disable after interrupt use.}  
AxIInterruptDisable();
```

See Also

[AxmInterruptSetAxisEnable](#), [AxmInterruptGetAxisEnable](#), [AxmInterruptRead](#), [AxmInterruptReadAxisFlag](#)

AxmInterruptSetAxis

Purpose

To use window message or callback API to bring interrupt message.

Format

C

```
DWORD AxmInterruptSetAxis (long lAxisNo, HWND hWnd, DWORD uMessage, AXT_NTERRUPT_PROC pProc,
                           HANDLE *pEvent);
```

Visual Basic

```
Function AxmInterruptSetAxis (ByVal lAxisNo As Long, ByVal hWnd As Long, ByVal uMessage As Long, ByVal pProc
                           As Long, ByRef pEvent As Long) As Long
```

Delphi

```
function AxmInterruptSetAxis (lAxisNo : LongInt; hWnd : HWND; uMessage : DWord; pProc : AXT_INTERRUPT_PROC; pEvent : PHWND) : DWord; stdcall;
```

Input / Output

Name	in/out	Format	Init Value	Explanation
AxisNo	in	long		Axis number
Wnd	In	HWND		Window handle
Message	In	DWORD		Window message
Proc	In	AXT_I NTERRUPT_PROC		Callback API
Event	In	HANDLE*		Use event

Return AXT_RT_SUCCESS(0000) : Successful execution of API.

[* See error code Table for more information on status error codes](#)

Description

Send window handle to bring interrupt message, and send API pointer when use callback. See following table for argument.

C Example

```
// Use window message to bring interrupt message

BEGIN_MESSAGE_MAP(CCAMCIPDlg, CDialog)
// } }AFX_MSG_MAP
// Declared
ON_MESSAGE(WM_CAMCIP_INTERRUPT,OnTriggerInterrupt)
END_MESSAGE_MAP()

// When interrupt occurs, called 'OnTriggerInterrupt()'
AxmInterruptSetAxis(0, m_hWnd, NULL, NULL);
```

Basic Example

```
'Use window message to bring interrupt message
'Declaration
Private Sub AxtMsg1_OnMessage1(ByVal wParam As Long, ByVal lParam As Long)
    ' Interrupt message management sentence
End Sub

'When interrupt occurs, called 'AxtMsg1_OnMessage1()
AxtMsg1.Message1 = WM_CAMCIP_INTERRUPT

'When interrupt occurs, called 'OnTriggerInterrupt()
AxmInterruptSetAxis 0, Me.hWnd, WM_CAMCIP_INTERRUPT, 0
```

Delphi Example

```
{ Use window message to bring interrupt message }
{ Declaration }
procedure TForm1.OnInterruptMessage(var Msg : TMessage);

begin
{ Interrupt message management sentence }
{ When interrupt occurs, called 'OnInterruptMessage()' }
AxmInterruptSetAxis(Form1.Handle, WM_CAMCIP_INTERRUPT, nil);
end;
```

See Also

[AxmInterruptSetAxisEnable](#), [AxmInterruptGetAxisEnable](#), [AxmInterruptRead](#), [AxmInterruptReadAxisFlag](#)

AxmInterruptSetAxisEnable

Purpose

Set whether interrupt is used on set axis.

Format

C

```
DWORD AxmInterruptSetAxisEnable(long lAxisNo, DWORD uUse);
```

Visual Basic

```
Function AxmInterruptSetAxisEnable(ByVal lAxisNo As Long, ByVal uUse As Long) As Long
```

Delphi

```
function AxmInterruptSetAxisEnable (lAxisNo : LongInt; uUse : DWord) : DWord; stdcall;
```

Input / Output

Name	in/out	Format	Init Value	Explanation
AxisNo	in	long		Channel (axis) number(starting from 0)
Use	In	DWORD		Whether interrupt is set

uUse

#define	Value	Explanation
DISABLE	00h	Interrupt not-used
ENABLE	01h	Interrupt used

Return AXT_RT_SUCCESS(0000) : Successful execution of API.

[* See error code Table for more information on status error codes](#)

Description

This function sets whether the interrupt will be used for the corresponding axis, sets whether the global interrupt condition is applied to the corresponding axis, and sets the global interrupt mask for the corresponding axis to configure.

C Example

```
// Set interrupt use on number axis 0
AxmInterruptSetAxisEnable(0, ENABLE);

// Verify interrupt use setting on axis 0
DWORD uIntUse;
AxmInterruptGetAxisEnable(0, &uIntUse);
```

Basic Example

```
'Set interrupt use on number axis 0
AxmInterruptSetAxisEnable 0, ENABLE

'Verify interrupt use setting on axis 0
Dim uIntUse As Long
```

```
AxmInterruptGetAxisEnable 0, uIntUse
```

Delphi Example

```
{ Verify interrupt use setting on axis 0 }
var
uIntUse : DWord

begin
{ Set interrupt use on axis 0 }
AxmInterruptSetAxisEnable(0, ENABLE);

AxmInterruptGetAxisEnable(0, @uIntUse);
End;
```

See Also

[AxmlInterruptSetAxis](#), [AxmlInterruptGetAxisEnable](#), [AxmlInterruptRead](#), [AxmlInterruptReadAxisFlag](#),
[AxmlInterruptSetMotionDoneEnable](#), [AxmlInterruptGetMotionDoneEnable](#)

AxmInterruptGetAxisEnable

Purpose

Return setting whether interrupt is used on corresponding axis.

Format

C

```
DWORD AxmInterruptGetAxisEnable(long lAxisNo, DWORD *upUse);
```

Visual Basic

```
Function AxmInterruptGetAxisEnable(ByVal lAxisNo As Long, ByRef upUse As Long) As Long
```

Delphi

```
function AxmInterruptGetAxisEnable (lAxisNo : LongInt; upUse : PDWord) : DWord; stdcall;
```

Input / Output

Name	in/out	Format	Init Value	Explanation
AxisNo	in	long		Channel (axis) number(starting from 0)
Use	out	DWORD*		Whether interrupt is set

Use

#define	Value	Explanation
DISABLE	00h	Interrupt not-used
ENABLE	01h	Interrupt used

Return AXT_RT_SUCCESS(0000) : Successful execution of API.

[* See error code Table for more information on status error codes](#)

Description

Return setting whether interrupt is used on corresponding axis that [AxmInterruptSetAxisEnable](#) is set

C Example

```
// Set interrupt use on axis 0
AxmInterruptSetAxisEnable(0, ENABLE);

// Verify interrupt use setting on axis 0
DWORD uIntUse;
AxmInterruptGetAxisEnable(0, &uIntUse);
```

Basic Example

```
'Set interrupt use on axis 0
AxmInterruptSetAxisEnable 0, ENABLE

'Verify interrupt use setting on axis 0
Dim uIntUse As Long
AxmInterruptGetAxisEnable 0, uIntUse
```

Delphi Example

```
{ Verify interrupt use setting on axis 0 }
var
uIntUse : DWord

begin
{ Set interrupt use on axis 0 }
AxmInterruptSetAxisEnable(0, ENABLE);

AxmInterruptGetAxisEnable(0, @uIntUse);
End;
```

See Also

[AxmlInterruptSetAxis](#), [AxmlInterruptSetAxisEnable](#), [AxmlInterruptRead](#), [AxmlInterruptReadAxisFlag](#),

AxmInterruptRead

Purpose

Return whether interrupt is used on corresponding axis.

Format

C

```
DWORD AxmInterruptRead(long *lpAxisNo, DWORD *upFlag);
```

Visual Basic

```
Function AxmInterruptRead(ByRef lpAxisNo As Long, ByRef upFlag As Long) As Long
```

Delphi

```
Function AxmInterruptRead(lpAxisNo : PLongInt; upFlag : PDWord) : DWord; stdcall;
```

Input / Output

Name	in/out	Format	Init Value	Explanation
AxisNo	in	long*		Channel (axis) number(starting from 0)
Flag	In	DWORD*		Whether interrupt is set

Flag

#define	Value	Explanation
FALSE	00h	Interrupt not occurred
TRUE	01h	Interrupt occurred

Return AXT_RT_SUCCESS(0000) : Successful execution of API.

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

Description

Return occurred interrupt source verification to [AxmInterruptReadAxisFlag](#) API.

C Example

```
// Return whether interrupt is used on corresponding axis.
Long upAxisNo;
DWORD upFlag;
AxmInterruptRead(&upAxisNo, &upFlag);
```

Basic Example

```
' Return whether interrupt is used on corresponding axis.
Dim upAxisNo, upFlag As Long
AxmInterruptRead upAxisNo, upFlag
```

Delphi Example

```
{ Return whether interrupt is used on corresponding axis.}
var
upFlag: DWord;
upAxisNo: LongInt;

begin
AxmInterruptRead(@upAxisNo, @upFlag);
End;
```

See Also

[AxmInterruptSetAxis](#), [AxmInterruptSetAxisEnable](#), [AxmInterruptGetAxisEnable](#),
[AxmInterruptReadAxisFlag](#), [AxmInterruptSetMotionDoneEnable](#) [AxmInterruptGetMotionDoneEnable](#)

AxmInterruptReadAxisFlag

Purpose

Return interrupt flag value on corresponding axis.

Format

C

```
DWORD AxmInterruptReadAxisFlag(long IAxisNo, long IBank, DWORD *upFlag);
```

Visual Basic

```
Function AxmInterruptReadAxisFlag(ByVal IAxisNo As Long, ByVal IBank As Long, ByRef upFlag As Long) As Long
```

Delphi

```
Function AxmInterruptReadAxisFlag(IAxisNo : LongInt; IBank : LongInt; upFlag : PDWord) : DWord; stdcall;
```

Input / Output

Name	in/out	Format	Init Value	Explanation
AxisNo	in	long		Channel (axis) number(starting from 0)
Bank	in	long		Interrupt bank number on corresponding axis(0: Bank1 , 1:Bank2)
Flag	out	DWORD*		Interrupt occurrence state value

Flag

#define	Value	Explanation
FALSE	00h	Interrupt not occurred
TRUE	01h	Interrupt occurred

Return AXT_RT_SUCCESS(0000) : Successful execution of API.

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

Description

32-bit interrupt flag register exists as 2 banks on motion chip installed on motion module in use. With verification of interrupt flag value of each bank, it can verify interrupt flag value that is outputted.

In case of SMC-2V03

AXT_MOTION_IPINTERRUPT_BANK1_DEF

#define	Bank (0)	Value	Explanation
IPINTBANK1_DRIVE_END		00000001h	Bit 0, Drive end(default value : 1).
IPINTBANK1_ICG		00000002h	Bit 1, INCNT is greater than INCNTCMP.
IPINTBANK1_ICE		00000004h	Bit 2, INCNT is equal with INCNTCMP.
IPINTBANK1_ICL		00000008h	Bit 3, INCNT is less than INCNTCMP.
IPINTBANK1_ECG		00000010h	Bit 4, EXCNT is greater than EXCNTCMP.
IPINTBANK1_ECE		00000020h	Bit 5, EXCNT is equal with EXCNTCMP.
IPINTBANK1_ECL		00000040h	Bit 6, EXCNT is less than EXCNTCMP.

IPINTBANK1_SCRQEMPTY	00000080h	Bit 7, Script control queue is empty.
IPINTBANK1_CAPRQEMPTY	00000100h	Bit 8, Caption result data queue is empty.
IPINTBANK1_SCRREG1EXE	00000200h	Bit 9, Script control register-1 Cmd is executed.
IPINTBANK1_SCRREG2EXE	00000400h	Bit 10, Script control register-2 Cmd is executed.
IPINTBANK1_SCRREG3EXE	00000800h	Bit 11, Script control register-3 Cmd is executed.
IPINTBANK1_CAPREG1EXE	00001000h	Bit 12, Caption control register-1 Cmd is executed.
IPINTBANK1_CAPREG2EXE	00002000h	Bit 13, Caption control register-2 Cmd is executed.
IPINTBANK1_CAPREG3EXE	00004000h	Bit 14, Caption control register-3 Cmd is executed.
IPINTBANK1_INTGGENCMD	00008000h	Bit 15, Interrupt generation Cmd is executed(0xFF)
IPINTBANK1_DOWN	00010000h	Bit 16, At starting point for deceleration drive.
IPINTBANK1_CONT	00020000h	Bit 17, At starting point for constant speed drive.
IPINTBANK1_UP	00040000h	Bit 18, At starting point for acceleration drive.
IPINTBANK1_SIGNALDETECTED	00080000h	Bit 19, Signal assigned in MODE1 is detected.
IPINTBANK1_SP23E	00100000h	Bit 20, Current speed is equal with rate change point RCP23.
IPINTBANK1_SP12E	00200000h	Bit 21, Current speed is equal with rate change point RCP12.
IPINTBANK1_SPE	00400000h	Bit 22, Current speed is equal with speed comparison data(SPDCMP).
IPINTBANK1_INCEICM	00800000h	Bit 23, INTCNT(1'st counter) is equal with ICM(1'st count minus limit data)
IPINTBANK1_SCRQEXE	01000000h	Bit 24, Script queue Cmd is executed When SCRCONQ's 30 bit is '1'.
IPINTBANK1_CAPQEXE	02000000h	Bit 25, Caption queue Cmd is executed When CAPCONQ's 30 bit is '1'.
IPINTBANK1_SLM	04000000h	Bit 26, NSLM/PSLM input signal is activated.
IPINTBANK1_ELM	08000000h	Bit 27, NELM/PELM input signal is activated.
IPINTBANK1_USERDEFINE1	10000000h	Bit 28, Selectable interrupt source 0(refer

		"0xFE" Cmd).
IPINTBANK1_USERDEFINE2	20000000h	Bit 29, Selectable interrupt source 1(refer "0xFE" Cmd).
IPINTBANK1_USERDEFINE3	40000000h	Bit 30, Selectable interrupt source 2(refer "0xFE" Cmd).
IPINTBANK1_USERDEFINE4	80000000h	Bit 31, Selectable interrupt source 3(refer "0xFE" Cmd).

AXT_MOTION_IPINTERRUPT_BANK2_DEF

#define	IBank (1)	Value	Explanation
IPINTBANK2_L_C_INP_Q_EMPTY	00000001h	Bit 0, Linear/Circular interpolation parameter queue is empty.	
IPINTBANK2_P_INP_Q_EMPTY	00000002h	Bit 1, Bit pattern interpolation queue is empty.	
IPINTBANK2_ALARM_ERROR	00000004h	Bit 2, Alarm input signal is activated.	
IPINTBANK2_INPOS	00000008h	Bit 3, InPos input signal is activated.	
IPINTBANK2_MARK_SIGNAL_HIG H	00000010h	Bit 4, Mark input signal is activated.	
IPINTBANK2_SSTOP_SIGNAL	00000020h	Bit 5, SSTOP input signal is activated.	
IPINTBANK2_ESTOP_SIGNAL	00000040h	Bit 6, ESTOP input signal is activated.	
IPINTBANK2_SYNC_ACTIVATED	00000080h	Bit 7, SYNC input signal is activated.	
IPINTBANK2_TRIGGER_ENABLE	00000100h	Bit 8, Trigger output is activated.	
IPINTBANK2_EXCNTCLR	00000200h	Bit 9, External(2'nd) counter is cleared by EXCNTCLR setting.	
IPINTBANK2_FSTCOMPARE_RE SULT BIT0	00000400h	Bit 10, ALU1's compare result bit 0 is activated.	
IPINTBANK2_FSTCOMPARE_RE SULT BIT1	00000800h	Bit 11, ALU1's compare result bit 1 is activated.	
IPINTBANK2_FSTCOMPARE_RE SULT BIT2	00001000h	Bit 12, ALU1's compare result bit 2 is activated.	
IPINTBANK2_FSTCOMPARE_RE SULT_BIT3	00002000h	Bit 13, ALU1's compare result bit 3 is activated.	
IPINTBANK2_FSTCOMPARE_RE SULT_BIT4	00004000h	Bit 14, ALU1's compare result bit 4 is activated.	
IPINTBANK2_SNDCOMPARE_RE SULT_BIT0	00008000h	Bit 15, ALU2's compare result bit 0 is activated.	
IPINTBANK2_SNDCOMPARE_RE SULT_BIT1	00010000h	Bit 16, ALU2's compare result bit 1 is activated.	

IPINTBANK2_SNDCOMPARE_REL SULT_BIT2	00020000h	Bit 17, ALU2's compare result bit 2 is activated.
IPINTBANK2_SNDCOMPARE_REL SULT_BIT3	00040000h	Bit 18, ALU2's compare result bit 3 is activated.
IPINTBANK2_SNDCOMPARE_REL SULT_BIT4	00080000h	Bit 19, ALU2's compare result bit 4 is activated.
IPINTBANK2_L_C_INP_Q_LESS_4	00100000h	Bit 20, Linear/Circular interpolation parameter queue is less than 4.
IPINTBANK2_P_INP_Q_LESS_4	00200000h	Bit 21, Pattern interpolation parameter queue is less than 4
IPINTBANK2_XSYNC_ACTIVATE_D	00400000h	Bit 22, X axis sync input signal is activated.
IPINTBANK2_YSYNC_ACTIVATE_D	00800000h	Bit 23, Y axis sync input siangl is activated.
IPINTBANK2_P_INP_END_BY_EN_D_PATTERN	01000000h	Bit 24, Bit pattern interpolation is terminated by end pattern.

In case of PCI-N404/804

AXT_MOTION_QI_INTERRUPT_BANK1_DEF

#define	IBank (0)	Value	Explanation
QIINTBANK1_0	00000001h	Bit 0, when drive set interrupt occurrence use exits	
QIINTBANK1_1	00000002h	Bit 1, when drive is exited	
QIINTBANK1_2	00000004h	Bit 2, when drive is started	
QIINTBANK1_3	00000008h	Bit 3, counter #1 < comparator #1 event occurrence	
QIINTBANK1_4	00000010h	Bit 4, counter #1 = comparator #1 event occurrence	
QIINTBANK1_5	00000020h	Bit 5, counter #1 > comparator #1 event occurrence	
QIINTBANK1_6	00000040h	Bit 6, counter #2 < comparator #2 event occurrence	
QIINTBANK1_7	00000080h	Bit 7, counter #2 = comparator #2 event occurrence	
QIINTBANK1_8	00000100h	Bit 8, counter #2 > comparator #2 event occurrence	
QIINTBANK1_9	00000200h	Bit 9, counter #3 < comparator #3 event occurrence	
QIINTBANK1_10	00000400h	Bit 10, counter #3 = comparator #3 event occurrence	
QIINTBANK1_11	00000800h	Bit 11, counter #3 > comparator #3 event occurrence	
QIINTBANK1_12	00001000h	Bit 12, counter #4 < comparator #4 event occurrence	
QIINTBANK1_13	00002000h	Bit 13, counter #4 = comparator #4 event occurrence	
QIINTBANK1_14	00004000h	Bit 14, counter #4 < comparator #4 event occurrence	
QIINTBANK1_15	00008000h	Bit 15, counter #5 < comparator #5 event occurrence	
QIINTBANK1_16	00010000h	Bit 16, counter #5 = comparator #5 event occurrence	

QIINTBANK1_17	00020000h	Bit 17, counter #5 > comparator #5 event occurrence
QIINTBANK1_18	00040000h	Bit 18, timer #1 event occurrence
QIINTBANK1_19	00080000h	Bit 19, timer #2 event occurrence
QIINTBANK1_20	00100000h	Bit 20, drive reservation setting Queue is cleared.
QIINTBANK1_21	00200000h	Bit 21, drive reservation setting Queue is full
QIINTBANK1_22	00400000h	Bit 22, trigger occurring distance period/absolute position Queue is cleared.
QIINTBANK1_23	00800000h	Bit 23, trigger occurring distance period/absolute position Queue is full
QIINTBANK1_24	01000000h	Bit 24, trigger signal occurrence event
QIINTBANK1_25	02000000h	Bit 25, script #1 command reservation setting Queue is cleared.
QIINTBANK1_26	04000000h	Bit 26, script #2 command reservation setting Queue is cleared.
QIINTBANK1_27	08000000h	Bit 27, script #3 initialized with execution of command reservation setting register.
QIINTBANK1_28	10000000h	Bit 28, script #4 initialized with execution of command reservation setting register.
QIINTBANK1_29	20000000h	Bit 29, servo alarm signal is permitted
QIINTBANK1_30	40000000h	Bit 30, $ CNT1 - CNT2 \geq CNT4 $ event occurrence.
QIINTBANK1_31	80000000h	Bit 31, interrupt occurrence command INTGEN execution.

AXT_MOTION_QIINTERRUPT_BANK2_DEF

#define	IBank (1)	Value	Explanation
	QIINTBANK2_0	00000001h	Bit 0, script #1 reading command result Queue is full.
	QIINTBANK2_1	00000002h	Bit 1, script #2 reading command result Queue is full.
	QIINTBANK2_2	00000004h	Bit 2, script #3 reading command result register is renewed with new data
	QIINTBANK2_3	00000008h	Bit 3, script #4 reading command result register is renewed with new data
	QIINTBANK2_4	00000010h	Bit 4, when reservation command of script #1 is executed, command set by interrupt occurrence is executed

QIINTBANK2_5	00000020h	Bit 5, when reservation command of script #2 is executed, command set by interrupt occurrence is executed.
QIINTBANK2_6	00000040h	Bit 6, when reservation command of script #3 is executed, command set by interrupt occurrence is executed.
QIINTBANK2_7	00000080h	Bit 7, when reservation command of script #4 is executed, command set by interrupt occurrence is executed.
QIINTBANK2_8	00000100h	Bit 8, start drive
QIINTBANK2_9	00000200h	Bit 9, drive using servo position determination completion(InPos) function, exit condition occurrence.
QIINTBANK2_10	00000400h	Bit 10, event selection #1 condition occurrence to use during event counter operation.
QIINTBANK2_11	00000800h	Bit 11, event selection #2 condition occurrence to use during event counter operation.
QIINTBANK2_12	00001000h	Bit 12, SQSTR1 signal is permitted.
QIINTBANK2_13	00002000h	Bit 13, SQSTR2 signal is permitted.
QIINTBANK2_14	00004000h	Bit 14, UIO0 terminal signal is changed to '1'.
QIINTBANK2_15	00008000h	Bit 15, UIO1 terminal signal is changed to '1'.
QIINTBANK2_16	00010000h	Bit 16, UIO2 terminal signal is changed to '1'.
QIINTBANK2_17	00020000h	Bit 17, UIO3 terminal signal is changed to '1'.
QIINTBANK2_18	00040000h	Bit 18, UIO4 terminal signal is changed to '1'.
QIINTBANK2_19	00080000h	Bit 19, UIO5 terminal signal is changed to '1'.
QIINTBANK2_20	00100000h	Bit 20, UIO6 terminal signal is changed to '1'.
QIINTBANK2_21	00200000h	Bit 21, UIO7 terminal signal is changed to '1'.
QIINTBANK2_22	00400000h	Bit 22, UIO8 terminal signal is changed to '1'.
QIINTBANK2_23	00800000h	Bit 23, UIO9 terminal signal is changed to '1'.
QIINTBANK2_24	01000000h	Bit 24, UIO10 terminal signal is changed to '1'.
QIINTBANK2_25	02000000h	Bit 25, UIO11 terminal signal is changed to '1'.
QIINTBANK2_26	04000000h	Bit 26, error stop condition (LMT, ESTOP, STOP, ESTOP, CMD, ALARM) occurrence.
QIINTBANK2_27	08000000h	Bit 27, data setting error is occurred during interpolation.
QIINTBANK2_28	10000000h	Bit 28, Don't Care

QIINTBANK2_29	20000000h	Bit 29, limit signal (PELM, NELM) is inputted.
QIINTBANK2_30	40000000h	Bit 30, additional limit signal (PSLM, NSLM) is inputted.
QIINTBANK2_31	80000000h	Bit 31, emergency stop signal (ESTOP) is input.

C Example

```
//Return interrupt flag value of number 1 bank on corresponding axis.
Long upAxisNo;
DWORD upFlag;

AxmInterruptReadAxisFlag(&upAxisNo, 0, &upFlag);
```

Basic Example

```
'Return interrupt flag value of number 1 bank on corresponding axis.
Dim upAxisNo, upFlag As Long
AxmInterruptReadAxisFlag upAxisNo, 0, upFlag
```

Delphi Example

```
{ Return interrupt flag value of number 1 bank on corresponding axis.}
var
  upFlag: DWord;
  upAxisNo: LongInt;

begin
  AxmInterruptReadAxisFlag (@upAxisNo, 0, @upFlag);
End;
```

See Also

[AxmInterruptSetAxis](#), [AxmInterruptSetAxisEnable](#), [AxmInterruptGetAxisEnable](#), [AxmInterruptRead](#),

AxmInterruptSetUserEnable

Purpose

Set whether Interrupt that user wanted occurred on a specific axis.

Format

C

```
DWORD AxmInterruptSetUserEnable (long lAxisNo, long lBank, DWORD ulInterruptNum);
```

Visual Basic

```
Function AxmInterruptSetUserEnable (ByVal lAxisNo As Long, ByVal lBank As Long , ByVal ulInterruptNum As Long )  
As Long
```

Delphi

```
function AxmInterruptSetUserEnable (lAxisNo : LongInt; lBank : LongInt; ulInterruptNum : DWORD) : DWord; stdcall;
```

Input / Output

Name	in/out	Format	Init Value	Explanation
AxisNo	in	long		Channel (axis) number(starting from 0)
Bank	In	long		Interrupt bank number user wanted(0,1) -> refer to Description
InterruptNum	In	DWORD		Interrupt bank number user wanted (0 – 31) –> refer to Description

Return AXT_RT_SUCCESS(0000) : Successful execution of API.

[* See error code Table for more information on status error codes](#)

Description

When interrupt is used, set to ‘ENABLE,’ otherwise, set to ‘DISABLE.’ If interrupt is used, Active Level of interrupt output signal can be set to ‘HIGH’ or ‘LOW’.

In case of SMC-2V03

AXT_MOTION_IPINTERRUPT_BANK1_DEF

#define Bank (0)	Value	Explanation
IPINTBANK1_DONTUSE	00000000h	INTERRUT DISABLED
IPINTBANK1_DRIVE_END	00000001h	Bit 0, Drive end(default value : 1).
IPINTBANK1_ICG	00000002h	Bit 1, INCNT is greater than INCNTCMP.
IPINTBANK1_ICE	00000004h	Bit 2, INCNT is equal with INCNTCMP.
IPINTBANK1_JCL	00000008h	Bit 3, INCNT is less than INCNTCMP.
IPINTBANK1_ECG	00000010h	Bit 4, EXCNT is greater than EXCNTCMP.
IPINTBANK1_ECE	00000020h	Bit 5, EXCNT is equal with EXCNTCMP.
IPINTBANK1_ECL	00000040h	Bit 6, EXCNT is less than EXCNTCMP.
IPINTBANK1_SCRQEMPTY	00000080h	Bit 7, Script control queue is empty.

IPINTBANK1_CAPRQEMPTY	00000100h	Bit 8, Caption result data queue is empty.
IPINTBANK1_SCRREG1EXE	00000200h	Bit 9, Script control register-1 Cmd is executed.
IPINTBANK1_SCRREG2EXE	00000400h	Bit 10, Script control register-2 Cmd is executed.
IPINTBANK1_SCRREG3EXE	00000800h	Bit 11, Script control register-3 Cmd is executed.
IPINTBANK1_CAPREG1EXE	00001000h	Bit 12, Caption control register-1 Cmd is executed.
IPINTBANK1_CAPREG2EXE	00002000h	Bit 13, Caption control register-2 Cmd is executed.
IPINTBANK1_CAPREG3EXE	00004000h	Bit 14, Caption control register-3 Cmd is executed.
IPINTBANK1_INTGGENCMD	00008000h	Bit 15, Interrupt generation Cmd is executed(0xFF)
IPINTBANK1_DOWN	00010000h	Bit 16, At starting point for deceleration drive.
IPINTBANK1_CONT	00020000h	Bit 17, At starting point for constant speed drive.
IPINTBANK1_UP	00040000h	Bit 18, At starting point for acceleration drive.
IPINTBANK1_SIGNALDETECTED	00080000h	Bit 19, Signal assigned in MODE1 is detected.
IPINTBANK1_SP23E	00100000h	Bit 20, Current speed is equal with rate change point RCP23.
IPINTBANK1_SP12E	00200000h	Bit 21, Current speed is equal with rate change point RCP12.
IPINTBANK1_SPE	00400000h	Bit 22, Current speed is equal with speed comparison data(SPDCMP).
IPINTBANK1_INCEICM	00800000h	Bit 23, INTCNT(1'st counter) is equal with ICM(1'st count minus limit data)
IPINTBANK1_SCRQEXE	01000000h	Bit 24, Script queue Cmd is executed When SCRCONQ's 30 bit is '1'.
IPINTBANK1_CAPQEXE	02000000h	Bit 25, Caption queue Cmd is executed When CAPCONQ's 30 bit is '1'.
IPINTBANK1_SLM	04000000h	Bit 26, NSLM/PSLM input signal is activated.

IPINTBANK1_ELM	08000000h	Bit 27, NELM/PELM input signal is activated.
IPINTBANK1_USERDEFINE1	10000000h	Bit 28, Selectable interrupt source 0(refer "0xFE" Cmd).
IPINTBANK1_USERDEFINE2	20000000h	Bit 29, Selectable interrupt source 1(refer "0xFE" Cmd).
IPINTBANK1_USERDEFINE3	40000000h	Bit 30, Selectable interrupt source 2(refer "0xFE" Cmd).
IPINTBANK1_USERDEFINE4	80000000h	Bit 31, Selectable interrupt source 3(refer "0xFE" Cmd).

AXT_MOTION_IPINTERRUPT_BANK2_DEF

#define	IBank (1)	Value	Explanation
IPINTBANK2_DONTUSE	00000000h		INTERRUT DISABLED
IPINTBANK2_L_C_INP_Q_EMPTY	00000001h		Bit 0, Linear/Circular interpolation parameter queue is empty.
IPINTBANK2_P_INP_Q_EMPTY	00000002h		Bit 1, Bit pattern interpolation queue is empty.
IPINTBANK2_ALARM_ERROR	00000004h		Bit 2, Alarm input signal is activated.
IPINTBANK2_INPOS	00000008h		Bit 3, InPos input signal is activated.
IPINTBANK2_MARK_SIGNAL_HIGH	00000010h		Bit 4, Mark input signal is activated.
IPINTBANK2_SSTOP_SIGNAL	00000020h		Bit 5, SSTOP input signal is activated.
IPINTBANK2_ESTOP_SIGNAL	00000040h		Bit 6, ESTOP input signal is activated.
IPINTBANK2_SYNC_ACTIVATED	00000080h		Bit 7, SYNC input signal is activated.
IPINTBANK2_TRIGGER_ENABLE	00000100h		Bit 8, Trigger output is activated.
IPINTBANK2_EXCNTCLR	00000200h		Bit 9, External(2'nd) counter is cleared by EXCNTCLR setting.
IPINTBANK2_FSTCOMPARE_RESULT BIT0	00000400h		Bit 10, ALU1's compare result bit 0 is activated.
IPINTBANK2_FSTCOMPARE_RESULT BIT1	00000800h		Bit 11, ALU1's compare result bit 1 is activated.
IPINTBANK2_FSTCOMPARE_RESULT BIT2	00001000h		Bit 12, ALU1's compare result bit 2 is activated.
IPINTBANK2_FSTCOMPARE_RESULT BIT3	00002000h		Bit 13, ALU1's compare result bit 3 is activated.

IPINTBANK2_FSTCOMPARE_RE SULT_BIT4	00004000h	Bit 14, ALU1's compare result bit 4 is activated.
IPINTBANK2_SNDCOMPARE_RE SULT_BIT0	00008000h	Bit 15, ALU2's compare result bit 0 is activated.
IPINTBANK2_SNDCOMPARE_RE SULT_BIT1	00010000h	Bit 16, ALU2's compare result bit 1 is activated.
IPINTBANK2_SNDCOMPARE_RE SULT_BIT2	00020000h	Bit 17, ALU2's compare result bit 2 is activated.
IPINTBANK2_SNDCOMPARE_RE SULT_BIT3	00040000h	Bit 18, ALU2's compare result bit 3 is activated.
IPINTBANK2_SNDCOMPARE_RE SULT_BIT4	00080000h	Bit 19, ALU2's compare result bit 4 is activated.
IPINTBANK2_L_C_INP_Q_LESS_4	00100000h	Bit 20, Linear/Circular interpolation parameter queue is less than 4.
IPINTBANK2_P_INP_Q_LESS_4	00200000h	Bit 21, Pattern interpolation parameter queue is less than 4
IPINTBANK2_XSYNC_ACTIVATE D	00400000h	Bit 22, X axis sync input signal is activated.
IPINTBANK2_YSYNC_ACTIVATE D	00800000h	Bit 23, Y axis sync input signal is activated.
IPINTBANK2_P_INP_END_BY_EN D_PATTERN	01000000h	Bit 24, Bit pattern interpolation is terminated by end pattern.

In case of PCI-N404/804

AXT_MOTION_QIINTERRUPT_BANK1_DEF

#define	IBank (0)	Value	Explanation
QIINTBANK1_0	00000001h	Bit 0, when drive set interrupt occurrence use exits	
QIINTBANK1_1	00000002h	Bit 1, when drive is exited	
QIINTBANK1_2	00000004h	Bit 2, when drive is started	
QIINTBANK1_3	00000008h	Bit 3, counter #1 < comparator #1 event occurrence	
QIINTBANK1_4	00000010h	Bit 4, counter #1 = comparator #1 event occurrence	
QIINTBANK1_5	00000020h	Bit 5, counter #1 > comparator #1 event occurrence	
QIINTBANK1_6	00000040h	Bit 6, counter #2 < comparator #2 event occurrence	
QIINTBANK1_7	00000080h	Bit 7, counter #2 = comparator #2 event occurrence	
QIINTBANK1_8	00000100h	Bit 8, counter #2 > comparator #2 event occurrence	
QIINTBANK1_9	00000200h	Bit 9, counter #3 < comparator #3 event occurrence	

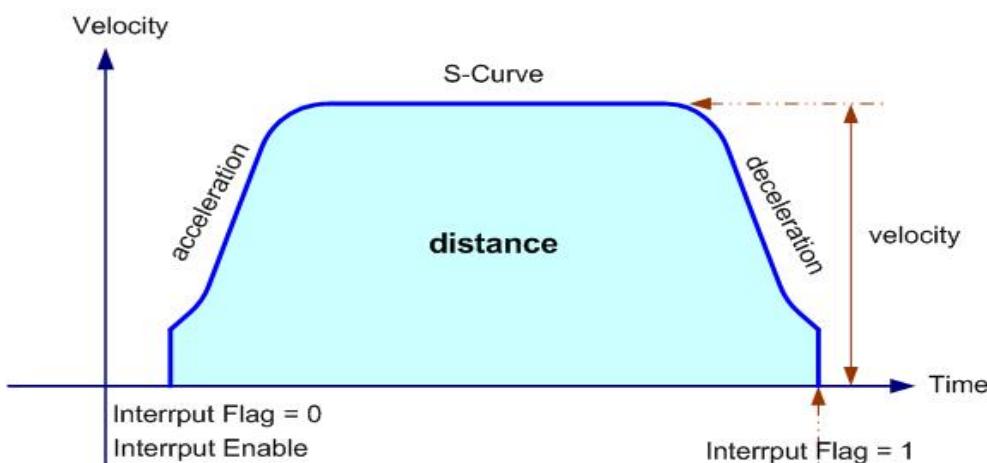
QIINTBANK1_10	00000400h	Bit 10, counter #3 = comparator #3 event occurrence
QIINTBANK1_11	00000800h	Bit 11, counter #3 > comparator #3 event occurrence
QIINTBANK1_12	00001000h	Bit 12, counter #4 < comparator #4 event occurrence
QIINTBANK1_13	00002000h	Bit 13, counter #4 = comparator #4 event occurrence
QIINTBANK1_14	00004000h	Bit 14, counter #4 < comparator #4 event occurrence
QIINTBANK1_15	00008000h	Bit 15, counter #5 < comparator #5 event occurrence
QIINTBANK1_16	00010000h	Bit 16, counter #5 = comparator #5 event occurrence
QIINTBANK1_17	00020000h	Bit 17, counter #5 > comparator #5 event occurrence
QIINTBANK1_18	00040000h	Bit 18, timer #1 event occurrence
QIINTBANK1_19	00080000h	Bit 19, timer #2 event occurrence
QIINTBANK1_20	00100000h	Bit 20, drive reservation setting Queue is cleared.
QIINTBANK1_21	00200000h	Bit 21, drive reservation setting Queue is full
QIINTBANK1_22	00400000h	Bit 22, trigger occurring distance period/absolute position Queue is cleared.
QIINTBANK1_23	00800000h	Bit 23, trigger occurring distance period/absolute position Queue is full
QIINTBANK1_24	01000000h	Bit 24, trigger signal occurrence event
QIINTBANK1_25	02000000h	Bit 25, script #1 command reservation setting Queue is cleared.
QIINTBANK1_26	04000000h	Bit 26, script #2 command reservation setting Queue is cleared.
QIINTBANK1_27	08000000h	Bit 27, script #3 initialized with execution of command reservation setting register.
QIINTBANK1_28	10000000h	Bit 28, script #4 initialized with execution of command reservation setting register.
QIINTBANK1_29	20000000h	Bit 29, servo alarm signal is permitted
QIINTBANK1_30	40000000h	Bit 30, $ CNT1 - CNT2 \geq CNT4 $ event occurrence.
QIINTBANK1_31	80000000h	Bit 31, interrupt occurrence command $ INTGEN $ execution.

AXT_MOTION_QIINTERRUPT_BANK2_DEF

#define	IBank (1)	Value	Explanation
	QIINTBANK2_0	00000001h	Bit 0, script #1 reading command result Queue is full.
	QIINTBANK2_1	00000002h	Bit 1, script #2 reading command result Queue is full.

QIINTBANK2_2	00000004h	Bit 2, script #3 reading command result register is renewed with new data
QIINTBANK2_3	00000008h	Bit 3, script #4 reading command result register is renewed with new data
QIINTBANK2_4	00000010h	Bit 4, when reservation command of script #1 is executed, command set by interrupt occurrence is executed
QIINTBANK2_5	00000020h	Bit 5, when reservation command of script #2 is executed, command set by interrupt occurrence is executed.
QIINTBANK2_6	00000040h	Bit 6, when reservation command of script #3 is executed, command set by interrupt occurrence is executed.
QIINTBANK2_7	00000080h	Bit 7, when reservation command of script #4 is executed, command set by interrupt occurrence is executed.
QIINTBANK2_8	00000100h	Bit 8, start drive
QIINTBANK2_9	00000200h	Bit 9, drive using servo position determination completion(InPos) function, exit condition occurrence.
QIINTBANK2_10	00000400h	Bit 10, event selection #1 condition occurrence to use during event counter operation.
QIINTBANK2_11	00000800h	Bit 11, event selection #2 condition occurrence to use during event counter operation.
QIINTBANK2_12	00001000h	Bit 12, SQSTR1 signal is permitted.
QIINTBANK2_13	00002000h	Bit 13, SQSTR2 signal is permitted.
QIINTBANK2_14	00004000h	Bit 14, UIO0 terminal signal is changed to '1'.
QIINTBANK2_15	00008000h	Bit 15, UIO1 terminal signal is changed to '1'.
QIINTBANK2_16	00010000h	Bit 16, UIO2 terminal signal is changed to '1'.
QIINTBANK2_17	00020000h	Bit 17, UIO3 terminal signal is changed to '1'.
QIINTBANK2_18	00040000h	Bit 18, UIO4 terminal signal is changed to '1'.
QIINTBANK2_19	00080000h	Bit 19, UIO5 terminal signal is changed to '1'.
QIINTBANK2_20	00100000h	Bit 20, UIO6 terminal signal is changed to '1'.
QIINTBANK2_21	00200000h	Bit 21, UIO7 terminal signal is changed to '1'.
QIINTBANK2_22	00400000h	Bit 22, UIO8 terminal signal is changed to '1'.
QIINTBANK2_23	00800000h	Bit 23, UIO9 terminal signal is changed to '1'.

QIINTBANK2_24	01000000h	Bit 24, UIO10 terminal signal is changed to '1'.
QIINTBANK2_25	02000000h	Bit 25, UIO11 terminal signal is changed to '1'.
QIINTBANK2_26	04000000h	Bit 26, error stop condition (LMT, ESTOP, STOP, ESTOP, CMD, ALARM) occurrence.
QIINTBANK2_27	08000000h	Bit 27, data setting error is occurred during interpolation.
QIINTBANK2_28	10000000h	Bit 28, Don't Care
QIINTBANK2_29	20000000h	Bit 29, limit signal (PELM, NELM) is inputted.
QIINTBANK2_30	40000000h	Bit 30, additional limit signal (PSLM, NSLM) is inputted.
QIINTBANK2_31	80000000h	Bit 31, emergency stop signal (ESTOP) is input.



C Example

```
// Verify whether interrupt occurred after completion of pulse output on
// axis 0
AxmInterruptSetUserEnable (0, 1 , QIINTBANK2_9);

DWORD dwInterruptNum
AxmInterrupt GetUserEnable (0, 1 , &dwInterruptNum);
```

VB Example

```
' Verify whether interrupt occurred after completion of pulse output on
// axis 0
AxmInterruptSetUserEnable 0, 1 , QIINTBANK2_9

Dim dwInterruptNum As Long
AxmInterrupt GetUserEnable 0, 1 , dwInterruptNum
```

Delphi Example

```
var
dwInterruptNum: DWord;

Begin
{ Verify whether interrupt occurred after completion of pulse output on }
```

```
    axis 0 }  
AxmInterruptSetUserEnable (0, 1, QIINTBANK2_9);  
  
AxmInterrupt GetUserEnable (0, 1 , @dwInterruptNum);  
End;
```

See Also

[AxmInterruptSetAxis](#), [AxmInterruptSetAxisEnable](#), [AxmInterruptGetAxisEnable](#), [AxmInterruptRead](#),
[AxmInterruptReadAxisFlag](#)

AxmInterrupt GetUserEnable

Purpose

Verify whether interrupt user wanted occurred on a specific axis.

Format

C

```
DWORD AxmInterrupt GetUserEnable(long lAxisNo, long lBank, DWORD *uplInterruptNum);
```

Visual Basic

```
Function AxmInterrupt GetUserEnable Lib "Axi.dll" (ByVal lAxisNo As Long, ByVal lBank As Long, ByVal uplInterruptNum As Long) As Long
```

Delphi

```
function AxmInterrupt GetUserEnable (lAxisNo : LongInt; lBank : LongInt; uplInterruptNum : PDWord) : DWord; stdcall;
```

Input / Output

Name	in/out	Format	Init Value	Explanation
AxisNo	in	long		Channel (axis) number(starting from 0)
Bank	In	long		Interrupt bank number user wanted(0,1) -> refer to Description
InterruptNum	Out	DWORD*		Interrupt bank number user wanted (0 – 31) -> refer to Description

Use

#define	Value	Explanation
DISABLE	00h	Interrupt not-used
ENABLE	01h	Interrupt used

Return AXT_RT_SUCCESS(0000) : Successful execution of API.

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

Description

Use when reading 32-bit interrupt value set by '[AxmInterrupt GetUserEnable](#)'.

In case of SMC-2V03

AXT_MOTION_IPINTERRUPT_BANK1_DEF

#define	Bank (0)	Value	Explanation
IPINTBANK1_DONTUSE		00000000h	INTERRUT DISABLED
IPINTBANK1_DRIVE_END		00000001h	Bit 0, Drive end(default value : 1).
IPINTBANK1_ICG		00000002h	Bit 1, INCNT is greater than INCNTCMP.
IPINTBANK1_ICE		00000004h	Bit 2, INCNT is equal with INCNTCMP.
IPINTBANK1_ICL		00000008h	Bit 3, INCNT is less than INCNTCMP.

IPINTBANK1_ECG	00000010h	Bit 4, EXCNT is greater than EXCNTCMP.
IPINTBANK1_ECE	00000020h	Bit 5, EXCNT is equal with EXCNTCMP.
IPINTBANK1_ECL	00000040h	Bit 6, EXCNT is less than EXCNTCMP.
IPINTBANK1_SCRQEMPTY	00000080h	Bit 7, Script control queue is empty.
IPINTBANK1_CAPRQEMPTY	00000100h	Bit 8, Caption result data queue is empty.
IPINTBANK1_SCRREG1EXE	00000200h	Bit 9, Script control register-1 Cmd is executed.
IPINTBANK1_SCRREG2EXE	00000400h	Bit 10, Script control register-2 Cmd is executed.
IPINTBANK1_SCRREG3EXE	00000800h	Bit 11, Script control register-3 Cmd is executed.
IPINTBANK1_CAPREG1EXE	00001000h	Bit 12, Caption control register-1 Cmd is executed.
IPINTBANK1_CAPREG2EXE	00002000h	Bit 13, Caption control register-2 Cmd is executed.
IPINTBANK1_CAPREG3EXE	00004000h	Bit 14, Caption control register-3 Cmd is executed.
IPINTBANK1_INTGGENCMD	00008000h	Bit 15, Interrupt generation Cmd is executed(0xFF)
IPINTBANK1_DOWN	00010000h	Bit 16, At starting point for deceleration drive.
IPINTBANK1_CONT	00020000h	Bit 17, At starting point for constant speed drive.
IPINTBANK1_UP	00040000h	Bit 18, At starting point for acceleration drive.
IPINTBANK1_SIGNALDETECTED	00080000h	Bit 19, Signal assigned in MODE1 is detected.
IPINTBANK1_SP23E	00100000h	Bit 20, Current speed is equal with rate change point RCP23.
IPINTBANK1_SP12E	00200000h	Bit 21, Current speed is equal with rate change point RCP12.
IPINTBANK1_SPE	00400000h	Bit 22, Current speed is equal with speed comparison data(SPDCMP).
IPINTBANK1_INCEICM	00800000h	Bit 23, INTCNT(1'st counter) is equal with ICM(1'st count minus limit data)
IPINTBANK1_SCRQEXE	01000000h	Bit 24, Script queue Cmd is executed

		When SCRCONQ's 30 bit is '1'.
IPINTBANK1_CAPQEXE	02000000h	Bit 25, Caption queue Cmd is executed When CAPCONQ's 30 bit is '1'.
IPINTBANK1_SLM	04000000h	Bit 26, NSLM/PSLM input signal is activated.
IPINTBANK1_ELM	08000000h	Bit 27, NELM/PELM input signal is activated.
IPINTBANK1_USERDEFINE1	10000000h	Bit 28, Selectable interrupt source 0(refer "0xFE" Cmd).
IPINTBANK1_USERDEFINE2	20000000h	Bit 29, Selectable interrupt source 1(refer "0xFE" Cmd).
IPINTBANK1_USERDEFINE3	40000000h	Bit 30, Selectable interrupt source 2(refer "0xFE" Cmd).
IPINTBANK1_USERDEFINE4	80000000h	Bit 31, Selectable interrupt source 3(refer "0xFE" Cmd).

AXT_MOTION_IPINTERRUPT_BANK2_DEF

#define	IBank (1)	Value	Explanation
IPINTBANK2_DONTUSE	00000000h		INTERRUT DISABLED
IPINTBANK2_L_C_INP_Q_EMPTY	00000001h		Bit 0, Linear/Circular interpolation parameter queue is empty.
IPINTBANK2_P_INP_Q_EMPTY	00000002h		Bit 1, Bit pattern interpolation queue is empty.
IPINTBANK2_ALARM_ERROR	00000004h		Bit 2, Alarm input signal is activated.
IPINTBANK2_INPOS	00000008h		Bit 3, InPos input signal is activated.
IPINTBANK2_MARK_SIGNAL_HIGH	00000010h		Bit 4, Mark input signal is activated.
IPINTBANK2_SSTOP_SIGNAL	00000020h		Bit 5, SSTOP input signal is activated.
IPINTBANK2_ESTOP_SIGNAL	00000040h		Bit 6, ESTOP input signal is activated.
IPINTBANK2_SYNC_ACTIVATED	00000080h		Bit 7, SYNC input signal is activated.
IPINTBANK2_TRIGGER_ENABLE	00000100h		Bit 8, Trigger output is activated.
IPINTBANK2_EXCNTCLR	00000200h		Bit 9, External(2'nd) counter is cleared by EXCNTCLR setting.
IPINTBANK2_FSTCOMPARE_RESULTSULT_BIT0	00000400h		Bit 10, ALU1's compare result bit 0 is activated.

IPINTBANK2_FSTCOMPARE_RE SULT BIT1	00000800h	Bit 11, ALU1's compare result bit 1 is activated.
IPINTBANK2_FSTCOMPARE_RE SULT BIT2	00001000h	Bit 12, ALU1's compare result bit 2 is activated.
IPINTBANK2_FSTCOMPARE_RE SULT_BIT3	00002000h	Bit 13, ALU1's compare result bit 3 is activated.
IPINTBANK2_FSTCOMPARE_RE SULT_BIT4	00004000h	Bit 14, ALU1's compare result bit 4 is activated.
IPINTBANK2_SNDCOMPARE_RE SULT_BIT0	00008000h	Bit 15, ALU2's compare result bit 0 is activated.
IPINTBANK2_SNDCOMPARE_RE SULT_BIT1	00010000h	Bit 16, ALU2's compare result bit 1 is activated.
IPINTBANK2_SNDCOMPARE_RE SULT_BIT2	00020000h	Bit 17, ALU2's compare result bit 2 is activated.
IPINTBANK2_SNDCOMPARE_RE SULT_BIT3	00040000h	Bit 18, ALU2's compare result bit 3 is activated.
IPINTBANK2_SNDCOMPARE_RE SULT_BIT4	00080000h	Bit 19, ALU2's compare result bit 4 is activated.
IPINTBANK2_L_C_INP_Q_LESS_4	00100000h	Bit 20, Linear/Circular interpolation parameter queue is less than 4.
IPINTBANK2_P_INP_Q_LESS_4	00200000h	Bit 21, Pattern interpolation parameter queue is less than 4
IPINTBANK2_XSYNC_ACTIVATE_D	00400000h	Bit 22, X axis sync input signal is activated.
IPINTBANK2_YSYNC_ACTIVATE_D	00800000h	Bit 23, Y axis sync input signal is activated.
IPINTBANK2_P_INP_END_BY_EN_D_PATTERN	01000000h	Bit 24, Bit pattern interpolation is terminated by end pattern.

In case of PCI-N404/804

AXT_MOTION_QIINTERRUPT_BANK1_DEF

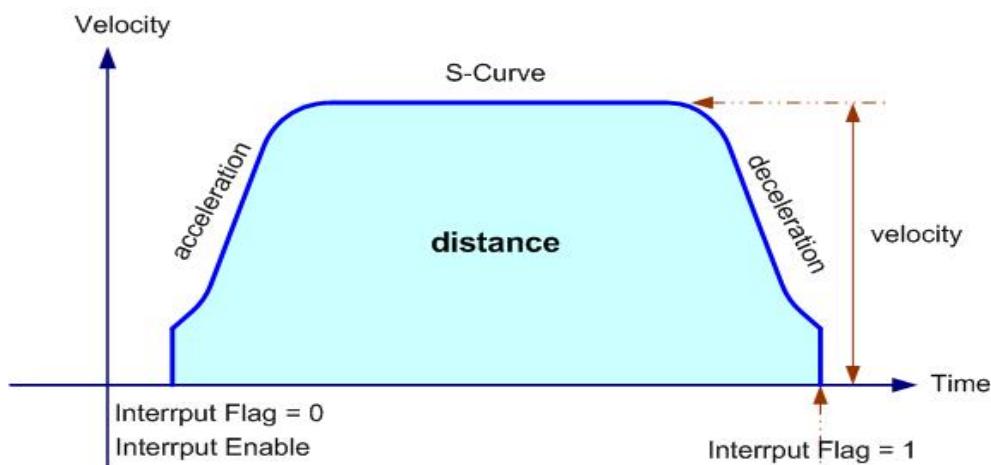
#define	IBank (0)	Value	Explanation
QIINTBANK1_0	00000001h	Bit 0, when drive set interrupt occurrence use exits	
QIINTBANK1_1	00000002h	Bit 1, when drive is exited	
QIINTBANK1_2	00000004h	Bit 2, when drive is started	
QIINTBANK1_3	00000008h	Bit 3, counter #1 < comparator #1 event occurrence	

QIINTBANK1_4	00000010h	Bit 4, counter #1 = comparator #1 event occurrence
QIINTBANK1_5	00000020h	Bit 5, counter #1 > comparator #1 event occurrence
QIINTBANK1_6	00000040h	Bit 6, counter #2 < comparator #2 event occurrence
QIINTBANK1_7	00000080h	Bit 7, counter #2 = comparator #2 event occurrence
QIINTBANK1_8	00000100h	Bit 8, counter #2 > comparator #2 event occurrence
QIINTBANK1_9	00000200h	Bit 9, counter #3 < comparator #3 event occurrence
QIINTBANK1_10	00000400h	Bit 10, counter #3 = comparator #3 event occurrence
QIINTBANK1_11	00000800h	Bit 11, counter #3 > comparator #3 event occurrence
QIINTBANK1_12	00001000h	Bit 12, counter #4 < comparator #4 event occurrence
QIINTBANK1_13	00002000h	Bit 13, counter #4 = comparator #4 event occurrence
QIINTBANK1_14	00004000h	Bit 14, counter #4 < comparator #4 event occurrence
QIINTBANK1_15	00008000h	Bit 15, counter #5 < comparator #5 event occurrence
QIINTBANK1_16	00010000h	Bit 16, counter #5 = comparator #5 event occurrence
QIINTBANK1_17	00020000h	Bit 17, counter #5 > comparator #5 event occurrence
QIINTBANK1_18	00040000h	Bit 18, timer #1 event occurrence
QIINTBANK1_19	00080000h	Bit 19, timer #2 event occurrence
QIINTBANK1_20	00100000h	Bit 20, drive reservation setting Queue is cleared.
QIINTBANK1_21	00200000h	Bit 21, drive reservation setting Queue is full
QIINTBANK1_22	00400000h	Bit 22, trigger occurring distance period/absolute position Queue is cleared.
QIINTBANK1_23	00800000h	Bit 23, trigger occurring distance period/absolute position Queue is full
QIINTBANK1_24	01000000h	Bit 24, trigger signal occurrence event
QIINTBANK1_25	02000000h	Bit 25, script #1 command reservation setting Queue is cleared.
QIINTBANK1_26	04000000h	Bit 26, script #2 command reservation setting Queue is cleared.
QIINTBANK1_27	08000000h	Bit 27, script #3 initialized with execution of command reservation setting register.
QIINTBANK1_28	10000000h	Bit 28, script #4 initialized with execution of command reservation setting register.
QIINTBANK1_29	20000000h	Bit 29, servo alarm signal is permitted
QIINTBANK1_30	40000000h	Bit 30, $ CNT1 - CNT2 \geq CNT4 $ event occurrence.
QIINTBANK1_31	80000000h	Bit 31, interrupt occurrence command INTGEN1 execution.

AXT_MOTION_QIINTERRUPT_BANK2_DEF

#define	IBank (1)	Value	Explanation
QIINTBANK2_0		00000001h	Bit 0, script #1 reading command result Queue is full.
QIINTBANK2_1		00000002h	Bit 1, script #2 reading command result Queue is full.
QIINTBANK2_2		00000004h	Bit 2, script #3 reading command result register is renewed with new data
QIINTBANK2_3		00000008h	Bit 3, script #4 reading command result register is renewed with new data
QIINTBANK2_4		00000010h	Bit 4, when reservation command of script #1 is executed, command set by interrupt occurrence is executed
QIINTBANK2_5		00000020h	Bit 5, when reservation command of script #2 is executed, command set by interrupt occurrence is executed.
QIINTBANK2_6		00000040h	Bit 6, when reservation command of script #3 is executed, command set by interrupt occurrence is executed.
QIINTBANK2_7		00000080h	Bit 7, when reservation command of script #4 is executed, command set by interrupt occurrence is executed.
QIINTBANK2_8		00000100h	Bit 8, start drive
QIINTBANK2_9		00000200h	Bit 9, drive using servo position determination completion(InPos) function, exit condition occurrence.
QIINTBANK2_10		00000400h	Bit 10, event selection #1 condition occurrence to use during event counter operation.
QIINTBANK2_11		00000800h	Bit 11, event selection #2 condition occurrence to use during event counter operation.
QIINTBANK2_12		00001000h	Bit 12, SQSTR1 signal is permitted.
QIINTBANK2_13		00002000h	Bit 13, SQSTR2 signal is permitted.
QIINTBANK2_14		00004000h	Bit 14, UIO0 terminal signal is changed to '1'.
QIINTBANK2_15		00008000h	Bit 15, UIO1 terminal signal is changed to '1'.
QIINTBANK2_16		00010000h	Bit 16, UIO2 terminal signal is changed to '1'.
QIINTBANK2_17		00020000h	Bit 17, UIO3 terminal signal is changed to '1'.

QIINTBANK2_18	00040000h	Bit 18, UIO4 terminal signal is changed to '1'.
QIINTBANK2_19	00080000h	Bit 19, UIO5 terminal signal is changed to '1'.
QIINTBANK2_20	00100000h	Bit 20, UIO6 terminal signal is changed to '1'.
QIINTBANK2_21	00200000h	Bit 21, UIO7 terminal signal is changed to '1'.
QIINTBANK2_22	00400000h	Bit 22, UIO8 terminal signal is changed to '1'.
QIINTBANK2_23	00800000h	Bit 23, UIO9 terminal signal is changed to '1'.
QIINTBANK2_24	01000000h	Bit 24, UIO10 terminal signal is changed to '1'.
QIINTBANK2_25	02000000h	Bit 25, UIO11 terminal signal is changed to '1'.
QIINTBANK2_26	04000000h	Bit 26, error stop condition (LMT, ESTOP, STOP, ESTOP, CMD, ALARM) occurrence.
QIINTBANK2_27	08000000h	Bit 27, data setting error is occurred during interpolation.
QIINTBANK2_28	10000000h	Bit 28, Don't Care
QIINTBANK2_29	20000000h	Bit 29, limit signal (PELM, NELM) is inputted.
QIINTBANK2_30	40000000h	Bit 30, additional limit signal (PSLM, NSLM) is inputted.
QIINTBANK2_31	80000000h	Bit 31, emergency stop signal (ESTOP) is input.



C Example

```
// Verify whether interrupt occurred after completion of pulse output on
// axis 0
AxmInterruptSetUserEnable (0, 1 , QIINTBANK2_9);

DWORD dwInterruptNum
AxmInterrupt GetUserEnable (0, 1 , &dwInterruptNum);
```

VB Example

```
' Verify whether interrupt occurred after completion of pulse output on
axis 0
AxmInterruptSetUserEnable 0, 1 , QIINTBANK2_9

Dim dwInterruptNum As Long
AxmInterrupt GetUserEnable 0, 1 , dwInterruptNum
```

Delphi Example

```
var
dwInterruptNum: DWord;

Begin
{ Verify whether interrupt occurred after completion of pulse output on
axis 0 }
AxmInterruptSetUserEnable (0, 1, QIINTBANK2_9);

AxmInterruptGetUserEnable (0, 1 , @dwInterruptNum);
End;
```

See Also

[AxmInterruptSetAxis](#), [AxmInterruptSetAxisEnable](#), [AxmInterruptGetAxisEnable](#), [AxmInterruptRead](#),
[AxmInterruptReadAxisFlag](#)

Motion Parameter Setting API

APIs that are introduced in this unit describe all parameter setting related to motion. Before drive execution, read motion parameter APIs carefully and execute drive.

Function	Description
AxmMotLoadParaAll	Calls 33 parameter files stored with AxmMotSaveParaAll . Arbitrary change is available by user's calling.
AxmMotSaveParaAll	Stores all 33 parameters by axis for the current all axes. Store as .mot file. Call a file using AxmMotLoadParaAll.
AxmMotSetParaLoad	Use AxmMotSaveParaAll API to store information which is set on register of CAMC-IP, QI as a file. The API stores information of all axes as a file. Set 4 parameters of dInitPos, dInitVel, dInitAccel and dInitDecel only from 28 to 31 among parameters. The rest parameters are stored using AxmMotSaveParaAll API reading current state when library is closed.
AxmMotGetParaLoad	Use AxmMotSaveParaAll API to store information which is set on register of CAMC-IP, QI as a file. The API stores information of all axes as a file. The API is used to call only 4 parameters of dInitPos, dInitVel, dInitAccel and dInitDecel from 28 to 31 among parameters that information of all axes is stored.
AxmMotSetPulseOutMethod	Sets pulse output method on a specific axis.
AxmMotGetPulseOutMethod	Returns pulse output method setting on a specific axis.
AxmMotSetEnclnputMethod	Sets encoder input method on a specific axis including increasing direction setting of external (actual) count on a specific axis.
AxmMotGetEnclnputMethod	Returns encoder input method on a specific axis including increasing direction setting of external (actual) count on a specific axis.
AxmMotSetMoveUnitPerPulse	Sets moving distance per pulse on a specific axis.
AxmMotGetMoveUnitPerPulse	Returns moving distance per pulse on a specific axis.
AxmMotSetDecelMode	Sets deceleration start point search method on a specific axis.
AxmMotGetDecelMode	Returns deceleration start point search method on a specific axis.
AxmMotSetRemainPulse	Sets remaining pulse in manual deceleration mode on a specific axis.
AxmMotGetRemainPulse	Returns remaining pulse in manual deceleration mode on a specific axis.

AxmMotSetMaxVel	Sets maximum velocity in constant speed drive API on a specific axis.
AxmMotGetMaxVel	Returns maximum velocity in constant speed drive API on a specific axis.
AxmMotSetAbsRelMode	<p>Sets moving distance calculation mode on a specific axis. If absolute position mode (0) is set, position value used in moving API is to be absolute position value. For example, if current position value is “10” and moves from absolute position mode to “100” position, it will stop at “100” position moving as much as “+90” from the current position.</p> <p>If relative position mode (1) is set, position value used moving API is to be moving amount to move from the current position. For example, if current position value is “10” and moves from relative position mode to “100” position, it will stop at “110” position moving as much as “+100” from the current position.</p>
AxmMotGetAbsRelMode	Returns moving distance calculation mode which is set on a specific axis.
AxmMotSetProfileMode	<p>Sets drive velocity profile mode on a specific axis.</p> <p>Velocity profile mode has symmetrical or asymmetrical Trapezoide / S-Curve mode and Quasi-S mode.</p>
AxmMotGetProfileMode	Returns drive velocity profile mode which is set on a specific axis.
AxmMotSetAccelUnit	<p>Sets acceleration unit on a specific axis.</p> <p>Acceleration unit can be selected among u/s^2, sec, rev/s^2.</p>
AxmMotGetAccelUnit	Returns acceleration unit which is set on a specific axis.
AxmMotSetMinVel	Sets initial velocity on a specific axis.
AxmMotGetMinVel	Returns initial velocity on a specific axis.
AxmMotSetAccelJerk	Sets acceleration jerk value on a specific axis.
AxmMotGetAccelJerk	Returns acceleration jerk value on a specific axis.
AxmMotSetDecelJerk	Sets deceleration jerk value on a specific axis.
AxmMotGetDecelJerk	Returns deceleration jerk value on a specific axis.

AxmMotLoadParaAll

Purpose

To read the configuration values of all axes from the specified file and set the system with those values. Parameter files are save by [AxmMotSaveParaAll](#) function, and the saved files can be edited by the user without restrictions.

Format

C

```
DWORD AxmMotLoadParaAll (char *spFilePath);
```

Visual Basic

```
Function AxmMotLoadParaAll (ByRef spFilePath As String) As Long
```

Delphi

```
function AxmMotLoadParaAll (spFilePath : PChar) : DWord ; stdcall;
```

Input / Output

Name	in/out	Format	Init Value	Explanation
spFilePath	in	char *		File name to call

Return AXT_RT_SUCCESS(0000) : Successful execution of API.

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

Description

Read and set from *spFilePath file assigned initial value of all axes. See [AxmMotSaveParaAll](#) table for loaded parameter. No .MOT setting file at the beginning. Basic setting value is set internally. To set directly, store .MOT file using [AxmMotSaveParaAll](#) API and call to AxmMotLoadParaAll.

Initial value of internal variable that file is set

No	Signal Related Variable	Parameter Description	Setting Value
0	AXIS_NO	Axis number	Setting axis number
0	PULSE_OUT_METHOD	Setting of the pulse output method which is used as a servo pack order on the motion board	TwoCcwCwHigh
0	Enc[Encoder]	Setting of the input method of encoder signal	Sqr4Mode
0	InP[InPos]	Setting of the use of completion signal for servo pack position decision and Active Level	2
0	Alm[Alarm]	Setting of the use of servo pack alarm signal and Active Level	1
0	NLimit	Setting of the use of machine (-) limit sensor and	1

		Active Level	
0	PLimit	Setting of the use of machine (+) limit sensor and Active Level	1
0	MinSpd	Setting of the initial drive velocity when the motor drives	1.0
0	MaxSpd	Setting of the maximum drive velocity when the motor drives	700000.0
0	HmSig	Setting of a signal to use as home sensor	4
1	HmLev	Setting of Active Level of home sensor	1
1	HmDir	Setting of the direction of initial search process during home sensor search	0
1	Zphas_Lev	Active level on encoder Z phase	1
1	Zphas_Use	Setting of the search of encoder Z phase after home sensor search	0
1	Stop Signal Mode	Mode when used ESTOP, SSTOP	0
1	Stop Signal level	Use level of ESTOP, SSTOP	0
1	VelFirst	Initial high speed search velocity during home search (unit of velocity is PPS[Pulses/sec] if Unit/pulse is 1/1)	10000.0
1	VelSecond	Velocity that comes out from the reverse direction after the 1 st sensor search during home search	10000.0
1	VelThird	Velocity of research process after the 1 st sensor search during home search	2000.0
1	VelLast	Setting of the final search velocity during home search [Accuracy decision of home search]	100.0
2	AccFirst	Initial high speed search acceleration during home search (unit of acceleration is PPS[Pulses/sec ²] if Unit/pulse is 1/1)	40000.0
2	AccSecond	Acceleration that comes out from the reverse direction after the 1 st sensor search during home search (unit of acceleration is PPS[Pulses/sec ²] if Unit/pulse is 1/1)	40000.0

2	HClrTim	Waiting time before clear of Cmd, Act position after home search completion	1000.0
2	HOffset	Moving offset value during reset of machine home after home search completion	0.0
2	NSWLimit	Setting of (-) Software Limit value to apply after home search completion	0.0
2	PSWLimit	Setting of (+) Software Limit value to apply after home search completion	0.0
2	OenPulse	Setting of pulse number that requires for 1 rotation of motor [refer to servo pack electric gear ratio]	1.0
2	OneUnit	Moving amount that is moved for 1 rotation of motor [mm, degree, etc]	1
2	InitPosition	Position for initialization	200
2	InitVel	Velocity for the initialization (unit of velocity is PPS[Pulses/sec] if Unit/pulse is 1/1)	200
3	InitAccel	Acceleration for the initialization (acceleration unit of acceleration is PPS[Pulses/sec^2] if Unit/pulse is 1/1)	400
3	InitDecel	Deceleration for the initialization (deceleration unit of acceleration is PPS[Pulses/sec^2] if Unit/pulse is 1/1)	400
3	AbReMode	Absolute/relative mode for the initialization	0
3	ProfMode	Velocity profile mode for the initialization	4

C Example

```
char szFileName[50];
strcpy(szFileName, "C:\\Data\\CamCIP.mot");

AxmMotLoadParaAll(szFileName);
// Read initial values of all axes on file and set.
```

VB Example

```
Dim szFileName As String

szFileName = "C:\\Data\\CamCIP.mot"
AxmMotLoadParaAll szFileName
' Read initial values of all axes on file and set.
```

Delphi Example

```
Var  
szFileName : String;  
  
begin  
szFileName := 'C:\\Data\\CamCIP.mot';  
AxmMotLoadParaAll(@szFileName);  
{ Read initial values of all axes on file and set. }  
end;
```

See Also

[AxmMotSaveParaAll](#), [AxmMotSetParaLoad](#), [AxmMotGetParaLoad](#) [AxmMotSetPulseOutMethod](#),
[AxmMotSetEnclInputMethod](#), [AxmMotGetEnclInputMethod](#), [AxmMotSetMoveUnitPerPulse](#),
[AxmMotGetMoveUnitPerPulse](#), [AxmMotSetDecelMode](#), [AxmMotGetDecelMode](#), [AxmMotSetSRate](#),
[AxmMotGetSRate](#), [AxmMotSetRemainPulse](#), [AxmMotGetRemainPulse](#)

AxmMotSaveParaAll

Purpose

Store all 33 parameters by axis for the all current axes. Store as .mot file. Call the file using [AxmMotLoadParaAll](#).

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 *		File name to store

Return AXT_RT_SUCCESS(0000) : Successful execution of API.

[* See error code Table for more information on status error codes](#)

Description

Store initial value of all axes in *szFilePath file. See [AxmMotSaveParaAll](#) table for loaded parameter. No .MOT setting file at the beginning. Basic setting value is set internally. To set directly, store .MOT file using [AxmMotSaveParaAll](#) API and call to AxmMotLoadParaAll.

Initial value of internal variable that file is set

No	Signal Related Variable	Parameter Description	Setting Value
0	AXIS_NO	Axis number	Setting axis number
0	PULSE_OUT METH OD	Setting of the pulse output method which is used as a servo pack order on the motion board	TwoCcWcwHigh
0	Enc[Encoder]	Setting of the input method of encoder signal	Sqr4Mode
0	InP[InPos]	Setting of the use of completion signal for servo pack position decision and Active Level	2
0	Alm[Alarm]	Setting of the use of servo pack alarm signal and Active Level	1
0	NLimit	Setting of the use of machine (-) limit sensor and Active	1

		Level	
0	PLimit	Setting of the use of machine (+) limit sensor and Active Level	1
0	MinSpd	Setting of the initial drive velocity when the motor drives	1.0
0	MaxSpd	Setting of the maximum drive velocity when the motor drives	700000.0
0	HmSig	Setting of a signal to use as home sensor	4
1	HmLev	Setting of Active Level of home sensor	1
1	HmDir	Setting of the direction of initial search process during home sensor search	0
1	Zphas_Lev	Active level on encoder Z phase	1
1	Zphas_Use	Setting of the search of encoder Z phase after home sensor search	0
1	Stop Signal Mode	Mode when used ESTOP, SSTOP	0
1	Stop Signal level	Use level of ESTOP, SSTOP	0
1	VelFirst	Initial high speed search velocity during home search (unit of velocity is PPS[Pulses/sec] if Unit/pulse is 1/1)	10000.0
1	VelSecond	Velocity that comes out from the reverse direction after the 1 st sensor search during home search	10000.0
1	VelThird	Velocity of research process after the 1 st sensor search during home search	2000.0
1	VelLast	Setting of the final search velocity during home search [Accuracy decision of home search]	100.0
2	AccFirst	Initial high speed search acceleration during home search (unit of acceleration is PPS[Pulses/sec ²] if Unit/pulse is 1/1)	40000.0
2	AccSecond	Acceleration that comes out from the reverse direction after the 1 st sensor search during home search (unit of acceleration is PPS[Pulses/sec ²] if Unit/pulse is 1/1)	40000.0
2	HClrTim	Waiting time before clear of Cmd, Act position after home search completion	1000.0

2	HOffset	Moving offset value during reset of machine home after home search completion	0.0
2	NSWLimit	Setting of (-) Software Limit value to apply after home search completion	0.0
2	PSWLimit	Setting of (+) Software Limit value to apply after home search completion	0.0
2	OenPulse	Setting of pulse number that requires for 1 rotation of motor [refer to servo pack electric gear ratio]	1.0
2	OneUnit	Moving amount that is moved for 1 rotation of motor [mm, degree, etc]	1
2	InitPosition	Position for initialization	200
2	InitVel	Velocity for the initialization (unit of velocity is PPS[Pulses/sec] if Unit/pulse is 1/1)	200
3	InitAccel	Acceleration for the initialization (acceleration unit of acceleration is PPS[Pulses/sec^2] if Unit/pulse is 1/1)	400
3	InitDecel	Deceleration for the initialization (deceleration unit of acceleration is PPS[Pulses/sec^2] if Unit/pulse is 1/1)	400
3	AbReMode	Absolute/relative mode for the initialization	0
3	ProfMode	Velocity profile mode for the initialization	4

C Example

```
char szFileName[50];
strcpy(szFileName, "C:\\Data\\CamCIP.mot");
AxmMotSaveParaAll(szFileName);
// Store initial values of all axes on file.
```

VB Example

```
Dim szFileName As String
szFileName = "C:\\Data\\CamCIP.mot"
AxmMotSaveParaAll szFileName
' Store initial values of all axes on file.
```

Delphi Example

```
Var  
szFileName : String;  
  
begin  
szFileName := 'C:\\Data\\CamCIP.mot';  
  
AxmMotSaveParaAll(@szFileName);  
{ Store initial values of all axes on file. }  
end;
```

See Also

[AxmMotLoadParaAll](#), [AxmMotSetParaLoad](#), [AxmMotGetParaLoad](#), [AxmMotSetPulseOutMethod](#),
[AxmMotSetEnclInputMethod](#), [AxmMotGetEnclInputMethod](#), [AxmMotSetMoveUnitPerPulse](#),
[AxmMotGetMoveUnitPerPulse](#), [AxmMotSetDecelMode](#), [AxmMotGetDecelMode](#), [AxmMotSetSRate](#),
[AxmMotGetSRate](#), [AxmMotSetRemainPulse](#), [AxmMotGetRemainPulse](#),

AxmMotSetParaLoad

Purpose

To set the parameter no. 28 to 31 – dInitPos, dInitVel, dInitAccel, dInitDecel – among the parameters saved by the AxmMotSaveParaAll function.

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		Channel (axis) number(starting from 0)
InitPos	in	double	1000	Initialization position (user brings this value as soon as library opens.)
InitVel	in	double	200	Initialization velocity (user brings this value as soon as library opens.)
InitAccel	In	double	400	Initialization acceleration (user brings this value as soon as library opens.)
InitDecel	In	double	400	Initialization deceleration (user brings this value as soon as library opens.)

Return AXT_RT_SUCCESS(0000) : Successful execution of API.

[* See error code Table for more information on status error codes](#)

Description

No .MOT setting file at the beginning. Basic setting value is set internally. To set directly, store .MOT file using AxmMotSaveParaAll API and call with AxmMotLoadParaAll. When close library, set 4 parameters with [AxmMotSetParaLoad](#) reading current state and store using AxmMotSaveParaAll API. If no setting, no stored of 4 parameters.

Initial value of internal variable that file is set

No	Signal Related Variable	Parameter Description	Setting Value
0	AXIS_NO	Axis number	Setting axis number

0	PULSE_OUT METH OD	Setting of the pulse output method which is used as a servo pack order on the motion board	TwoCcwCwHigh
0	Enc[Encoder]	Setting of the input method of encoder signal	Sqr4Mode
0	InP[InPos]	Setting of the use of completion signal for servo pack position decision and Active Level	2
0	Alm[Alarm]	Setting of the use of servo pack alarm signal and Active Level	1
0	NLimit	Setting of the use of machine (-) limit sensor and Active Level	1
0	PLimit	Setting of the use of machine (+) limit sensor and Active Level	1
0	MinSpd	Setting of the initial drive velocity when the motor drives	1.0
0	MaxSpd	Setting of the maximum drive velocity when the motor drives	700000.0
0	HmSig	Setting of a signal to use as home sensor	4
1	HmLev	Setting of Active Level of home sensor	1
1	HmDir	Setting of the direction of initial search process during home sensor search	0
1	Zphas_Lev	Active level on encoder Z phase	1
1	Zphas_Use	Setting of the search of encoder Z phase after home sensor search	0
1	Stop Signal Mode	Mode when used ESTOP, SSTOP	0
1	Stop Signal level	Use level of ESTOP, SSTOP	0
1	VelFirst	Initial high speed search velocity during home search (unit of velocity is PPS[Pulses/sec] if Unit/pulse is 1/1)	10000.0
1	VelSecond	Velocity that comes out from the reverse direction after the 1 st sensor search during home search	10000.0
1	VelThird	Velocity of research process after the 1 st sensor search during home search	2000.0
1	VelLast	Setting of the final search velocity during home search	100.0

		[Accuracy decision of home search]	
2	AccFirst	Initial high speed search acceleration during home search (unit of acceleration is PPS[Pulses/sec^2] if Unit/pulse is 1/1)	40000.0
2	AccSecond	Acceleration that comes out from the reverse direction after the 1 st sensor search during home search (unit of acceleration is PPS[Pulses/sec^2] if Unit/pulse is 1/1)	40000.0
2	HClrTim	Waiting time before clear of Cmd, Act position after home search completion	1000.0
2	HOffset	Moving offset value during reset of machine home after home search completion	0.0
2	NSWLimit	Setting of (-) Software Limit value to apply after home search completion	0.0
2	PSWLimit	Setting of (+) Software Limit value to apply after home search completion	0.0
2	OenPulse	Setting of pulse number that requires for 1 rotation of motor [refer to servo pack electric gear ratio]	1.0
2	OneUnit	Moving amount that is moved for 1 rotation of motor [mm, degree, etc]	1
2	InitPosition	Position for initialization	200
2	InitVel	Velocity for the initialization (unit of velocity is PPS[Pulses/sec] if Unit/pulse is 1/1)	200
3	InitAccel	Acceleration for the initialization (acceleration unit of acceleration is PPS[Pulses/sec^2] if Unit/pulse is 1/1)	400
3	InitDecel	Deceleration for the initialization (deceleration unit of acceleration is PPS[Pulses/sec^2] if Unit/pulse is 1/1)	400
3	AbReMode	Absolute/relative mode for the initialization	0
3	ProfMode	Velocity profile mode for the initialization	4

Set parameters in force from 28 to 33.

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](#), [AxmMotSetEnclInputMethod](#),
[AxmMotGetEnclInputMethod](#), [AxmMotSetMoveUnitPerPulse](#), [AxmMotGetMoveUnitPerPulse](#),
[AxmMotSetDecelMode](#), [AxmMotGetDecelMode](#), [AxmMotSetSRate](#),
[AxmMotGetSRate](#), [AxmMotSetRemainPulse](#), [AxmMotGetRemainPulse](#),

AxmMotGetParaLoad

Purpose

To get the parameter no. 28 to 33 – dInitPos, dInitVel, dInitAccel, dInitDecel, uInitAbsRelMode, and uInitProfileMode – among the parameters saved by the AxmMotSaveParaAll function.

Format

C

```
DWORD AxmMotGetParaLoad(long lAxisNo, double *dInitPos, double *dInitVel, double *dInitAccel, double
*dInitDecel);
```

Visual Basic

```
Function AxmMotGetParaLoad (ByVal lAxisNo As Long, ByRef dInitPos As Double, ByRef dInitVel As Double,
ByRef dInitAccel As Double, ByRef dInitDecel As Double) As Long
```

Delphi

```
function AxmMotGetParaLoad (lAxisNo : LongInt; dInitPos : PDouble; dInitVel : PDouble; dInitAccel : PDouble;
dInitDecel : PDouble) : DWord; stdcall;
```

Input / Output

Name	in/out	Format	Init Value	Explanation
AxisNo	in	long*		Channel (axis) number(starting from 0)
InitPos	out	double*	1000	Initialization position (user brings this value as soon as library opens.)
InitVel	out	double*	200	Initialization velocity (user brings this value as soon as library opens.)
InitAccel	out	double*	400	Initialization acceleration (user brings this value as soon as library opens.)
InitDecel	out	double*	400	Initialization deceleration (user brings this value as soon as library opens.)

Return AXT_RT_SUCCESS(0000) : Successful execution of API.

[* See error code Table for more information on status error codes](#)

Description

*Note: When only required data are needed to bring, it is able to use with input NULL in unrequired variable.

No .MOT setting file at the beginning. Basic setting value is set internally. To set directly, store .MOT file using AxmMotSaveParaAll API and call with AxmMotLoadParaAll.

Among parameters, use only 4 parameters of dInitPos, dInitVel, dInitAccel, dInitDecel, uInitAbsRelMode, and uInitProfileMode from 28 to 31 to read set value that is set inside library currently.

Initial value of internal variable that file is set

No	Signal Related Variable	Parameter Description	Setting Value

0	AXIS_NO	Axis number	Setting axis number
0	PULSE_OUT METH OD	Setting of the pulse output method which is used as a servo pack order on the motion board	TwoCcwCwHigh
0	Enc[Encoder]	Setting of the input method of encoder signal	Sqr4Mode
0	InP[InPos]	Setting of the use of completion signal for servo pack position decision and Active Level	2
0	Alm[Alarm]	Setting of the use of servo pack alarm signal and Active Level	1
0	NLimit	Setting of the use of machine (-) limit sensor and Active Level	1
0	PLimit	Setting of the use of machine (+) limit sensor and Active Level	1
0	MinSpd	Setting of the initial drive velocity when the motor drives	1.0
0	MaxSpd	Setting of the maximum drive velocity when the motor drives	700000.0
0	HmSig	Setting of a signal to use as home sensor	4
1	HmLev	Setting of Active Level of home sensor	1
1	HmDir	Setting of the direction of initial search process during home sensor search	0
1	Zphas_Lev	Active level on encoder Z phase	1
1	Zphas_Use	Setting of the search of encoder Z phase after home sensor search	0
1	Stop Signal Mode	Mode when used ESTOP, SSTOP	0
1	Stop Signal level	Use level of ESTOP, SSTOP	0
1	VelFirst	Initial high speed search velocity during home search (unit of velocity is PPS[Pulses/sec] if Unit/pulse is 1/1)	10000.0
1	VelSecond	Velocity that comes out from the reverse direction after the 1 st sensor search during home search	10000.0
1	VelThird	Velocity of research process after the 1 st sensor search	2000.0

		during home search	
1	VelLast	Setting of the final search velocity during home search [Accuracy decision of home search]	100.0
2	AccFirst	Initial high speed search acceleration during home search (unit of acceleration is PPS[Pulses/sec^2] if Unit/pulse is 1/1)	40000.0
2	AccSecond	Acceleration that comes out from the reverse direction after the 1 st sensor search during home search (unit of acceleration is PPS[Pulses/sec^2] if Unit/pulse is 1/1)	40000.0
2	HClrTim	Waiting time before clear of Cmd, Act position after home search completion	1000.0
2	HOffset	Moving offset value during reset of machine home after home search completion	0.0
2	NSWLimit	Setting of (-) Software Limit value to apply after home search completion	0.0
2	PSWLimit	Setting of (+) Software Limit value to apply after home search completion	0.0
2	OenPulse	Setting of pulse number that requires for 1 rotation of motor [refer to servo pack electric gear ratio]	1.0
2	OneUnit	Moving amount that is moved for 1 rotation of motor [mm, degree, etc]	1
2	InitPosition	Position for initialization	200
2	InitVel	Velocity for the initialization (unit of velocity is PPS[Pulses/sec] if Unit/pulse is 1/1)	200
3	InitAccel	Acceleration for the initialization (acceleration unit of acceleration is PPS[Pulses/sec^2] if Unit/pulse is 1/1)	400
3	InitDecel	Deceleration for the initialization (deceleration unit of acceleration is PPS[Pulses/sec^2] if Unit/pulse is 1/1)	400
3	AbReMode	Absolute/relative mode for the initialization	0
3	ProfMode	Velocity profile mode for the initialization	4

Return parameters from 28 to 31.

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
AxmMotGetParaLoad 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;
AxmMotGetParaLoad (0, dInitpos, dInitvel, dInitaccel, dInitdecel);

AxmMotGetParaLoad(0, @dReadInitpos,@dReadInitvel, @dReadInitaccel,
@dReadInitdecel);

End;
```

See Also

[AxmMotLoadParaAll](#), [AxmMotSetParaLoad](#), [AxmMotSetParaLoad](#) [AxmMotSetPulseOutMethod](#),
[AxmMotSetEnclInputMethod](#), [AxmMotGetEnclInputMethod](#), [AxmMotSetMoveUnitPerPulse](#),
[AxmMotGetMoveUnitPerPulse](#), [AxmMotSetDecelMode](#), [AxmMotGetDecelMode](#), [AxmMotSetSRate](#),
[AxmMotGetSRate](#), [AxmMotSetRemainPulse](#), [AxmMotGetRemainPulse](#),

AxmMotSetPulseOutMethod

Purpose

Set pulse output method on a specific axis.

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		Channel (axis) number(starting from 0)
Method	In	DWORD	4	Pulse output method

Method

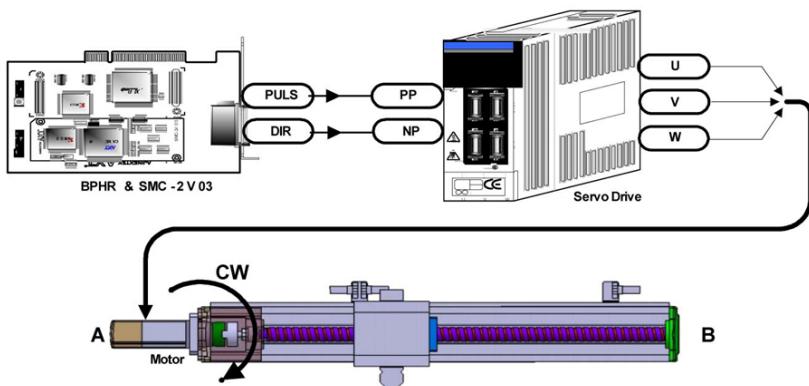
#define	Value	Explanation
OneHighLowHigh	00h	1 pulse method, PULSE(Active High), forward direction(DIR=Low) / reverse direction(DIR=High)
OneHighHighLow	01h	1 pulse method, PULSE(Active High), forward direction (DIR=High) / reverse direction (DIR=Low)
OneLowLowHigh	02h	1 pulse method, PULSE(Active Low), forward direction (DIR=Low) / reverse direction (DIR=High)
OneLowHighLow	03h	1 pulse method, PULSE(Active Low), forward direction (DIR=High) / reverse direction (DIR=Low)
TwoCcwCwHigh	04h	2 pulse method, PULSE(CCW: reverse direction), DIR(CW: forward direction), Active High
TwoCcwCwLow	05h	2 pulse method, PULSE(CCW: reverse direction), DIR(CW: forward direction), Active Low
TwoCwCcwHigh	06h	2 pulse method, PULSE(CW: forward direction), DIR(CCW: reverse direction), Active High
TwoCwCcwLow	07h	2 pulse method, PULSE(CW: forward direction), DIR(CCW: reverse direction), Active Low
TwoPhase	08h	2 phase (90° phase difference), PULSE lead DIR(CW: forward direction), PULSE lag DIR(CCW: reverse

		direction) – Non-used of IP
TwoPhaseReverse	09h	2 phase (90° phase difference), PULSE lead DIR(CCW: forward direction), PULSE lag DIR(CW: reverse direction) – Non-used of IP

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

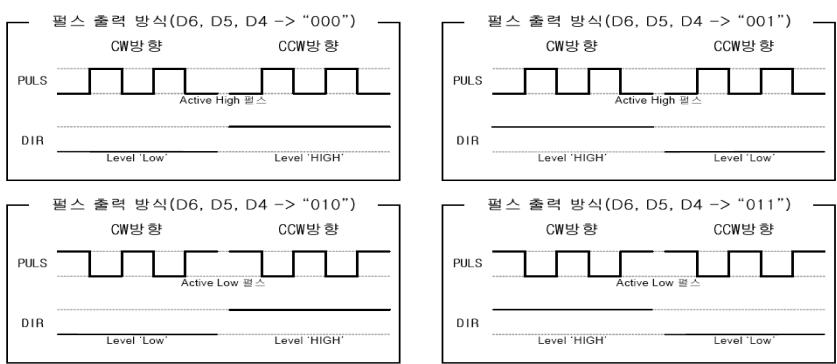
Description

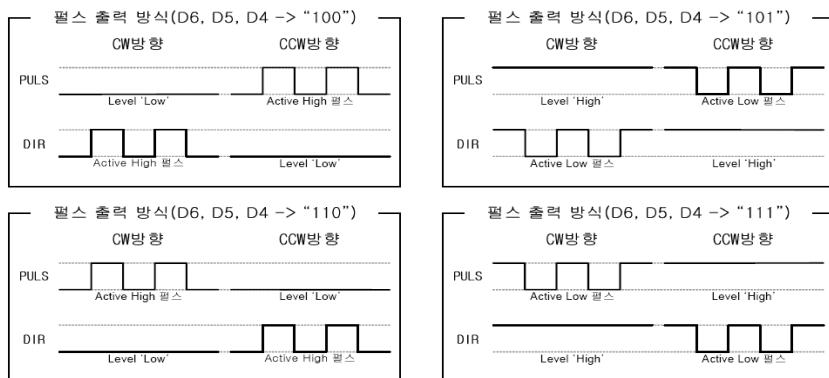
There are 2 pulse output methods generally; one pulse and two pulse output method, and each is divided into 4 methods. Pulse output method must be set same with servo driver in use. Pulse output method is generally divided into 1Pulse and 2Pulse method, and each method shows below. However, IP does not provide Method 8 to 9.



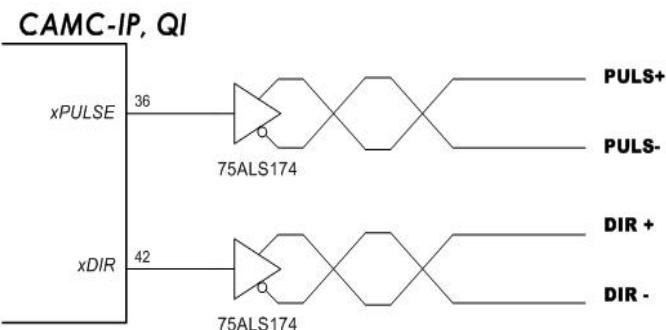
PULS, DIR pulse output method of motion controller outputted to servo drive is divided into One Pulse and Two Pulse method.

One Pulse Method





Value	Direction Signal (DIR)		Pulse Signal	Motiond	
	CW	CCW			
0	L	H	Active H Pulse	1 Pulse Method	
1	H	L			
2	L	H			
3	H	L			
4	CW Pulse	Active H	CCW Pulse	2 Pulse Method	
5		Active L			
6	CCW Pulse	Active H	CW Pulse		
7		Active L			

**C Example**

```
// Set to Two pulse method on axis 0
AxmMotSetPulseOutMethod(0, TwoCwCcwhigh);

// Verify pulse output method on axis 0
DWORD Method;
AxmMotGetPulseOutMethod (0, &Method);
```

VB Example

```
'Set to Two pulse method on axis 0
AxmMotSetPulseOutMethod 0, TwoCwCcwhigh

'Verify pulse output method on axis 0
```

```
Dim Method As Long  
AxmMotGetPulseOutMethod 0, Method
```

Delphi Example

```
{ Verify pulse output method on axis 0 }  
var  
Method: DWord;  
  
Begin  
{ Set to Two pulse method on axis 0 }  
AxmMotSetPulseOutMethod(0, TwoCwCcwHigh);  
  
AxmMotGetPulseOutMethod (0, @Method);  
end;
```

See Also

[AxmMotLoadParaAll](#), [AxmMotSaveParaAll](#), [AxmMotGetParaLoad](#), [AxmMotSetParaLoad](#),
[AxmMotGetPulseOutMethod](#), [AxmMotSetEnclInputMethod](#), [AxmMotGetEnclInputMethod](#),
[AxmMotSetMoveUnitPerPulse](#), [AxmMotGetMoveUnitPerPulse](#), [AxmMotSetDecelMode](#),
[AxmMotGetDecelMode](#), [AxmMotSetSRate](#), [AxmMotGetSRate](#), [AxmMotSetRemainPulse](#),
[AxmMotGetRemainPulse](#)

AxmMotGetPulseOutMethod

Purpose

Return pulse output method setting on a specific axis.

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		Channel (axis) number(starting from 0)
Method	out	DWORD*	4	Pulse output method

Method

#define	Value	Explanation
OneHighLowHigh	00h	1 pulse method, PULSE(Active High), forward direction(DIR=Low) / reverse direction(DIR=High)
OneHighHighLow	01h	1 pulse method, PULSE(Active High), forward direction (DIR=High) / reverse direction (DIR=Low)
OneLowLowHigh	02h	1 pulse method, PULSE(Active Low), forward direction (DIR=Low) / reverse direction (DIR=High)
OneLowHighLow	03h	1 pulse method, PULSE(Active Low), forward direction (DIR=High) / reverse direction (DIR=Low)
TwoCcwCwHigh	04h	2 pulse method, PULSE(CCW: reverse direction), DIR(CW: forward direction), Active High
TwoCcwCwLow	05h	2 pulse method, PULSE(CCW: reverse direction), DIR(CW: forward direction), Active Low
TwoCwCcwHigh	06h	2 pulse method, PULSE(CW: forward direction), DIR(CCW: reverse direction), Active High
TwoCwCcwLow	07h	2 pulse method, PULSE(CW: forward direction), DIR(CCW: reverse direction), Active Low
TwoPhase	08h	2 phase (90° phase difference), PULSE lead DIR(CW: forward direction), PULSE lag DIR(CCW: reverse direction)
TwoPhaseReverse	09h	2 phase (90° phase difference), PULSE lead DIR(CCW:

		forward direction), PULSE lag DIR(CW: reverse direction)
--	--	--

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

Description

Return pulse output method setting on a specific axis set with [AxmMotSetPulseOutMethod](#)

C Example

```
/ Set to Two pulse method on axis 0
AxmMotSetPulseOutMethod(0, TwoCwCcwHigh);

// Verify pulse output method on axis 0
DWORD Method;
AxmMotGetPulseOutMethod (0, &Method);
```

VB Example

```
'Set to Two pulse method on axis 0
AxmMotSetPulseOutMethod 0, TwoCwCcwHigh

'Verify pulse output method on axis 0
Dim Method As Long
AxmMotGetPulseOutMethod 0, Method
```

Delphi Example

```
{ Verify pulse output method on axis 0 }
var
Method: DWord;

Begin
{ Set to Two pulse method on axis 0 }
AxmMotSetPulseOutMethod(0, TwoCwCcwHigh);

AxmMotGetPulseOutMethod (0, @Method);
end;
```

See Also

[AxmMotSetPulseOutMethod](#), [AxmMotSetEnclInputMethod](#), [AxmMotGetEnclInputMethod](#),
[AxmMotSetMoveUnitPerPulse](#), [AxmMotGetMoveUnitPerPulse](#), [AxmMotSetDecelMode](#),
[AxmMotGetDecelMode](#), [AxmMotSetSRate](#), [AxmMotGetSRate](#), [AxmMotSetRemainPulse](#),
[AxmMotGetRemainPulse](#),

AxmMotSetEncInputMethod

Purpose

Set Encoder input method on a specific axis.

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		Channel (axis) number(starting from 0)
Method	In	DWORD	3	Encoder input method

Method

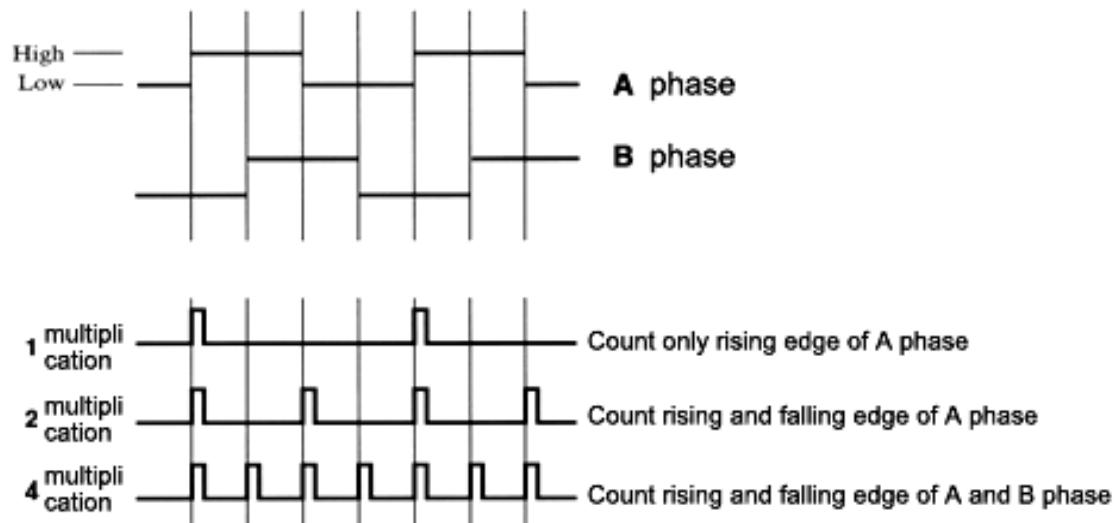
#define	Value	Explanation
ObverseUpDownMode	00h	Forward direction Up / Down
ObverseSqr1Mode	01h	Forward direction 1 multiplication
ObverseSqr2Mode	02h	Forward direction 2 multiplication
ObverseSqr4Mode	03h	Forward direction 4 multiplication
ReverseUpDownMode	04h	Reverse direction Up / Down
ReverseSqr1Mode	05h	Reverse direction 1 multiplication
ReverseSqr2Mode	06h	Reverse direction 2 multiplication
ReverseSqr4Mode	07h	Reverse direction 4 multiplication

Return AXT_RT_SUCCESS(0000) : Successful execution of API.

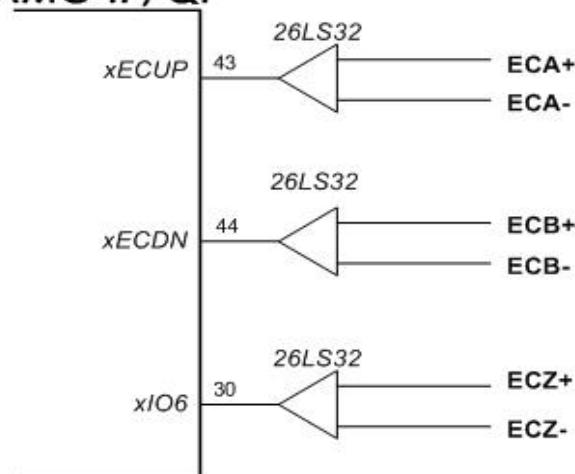
* See error code Table for more information on status error codes

Description

Encoder input method has 8 methods, and most of driver uses 4 multiplication.



CAMC-IP, QI



C Example

```
// Set Encoder input method to 4 multiplication method on axis 0
AxmMotSetEncInputMethod(0, ObverseSqr4Mode);

// Verify Encoder input method on axis 0
DWORD Method;
AxmMotGetEncInputMethod (0, &Method);
```

VB Example

```
'Set Encoder input method to 4 multiplication method on axis 0
AxmMotSetEncInputMethod 0, ObverseSqr4Mode

'Verify Encoder input method on axis 0
Dim Method As Long
AxmMotGetEncInputMethod 0, Method
```

Delphi Example

```
{ Verify Encoder input method on axis 0 }
var
```

```
Method: DWord;  
  
Begin  
{ Set Encoder input method to 4 multiplication method on axis 0 }  
AxmMotSetEncInputMethod(0, ObverseSqr4Mode);  
  
AxmMotGetEncInputMethod (0, @Method);  
end;
```

See Also

[AxmMotLoadParaAll](#), [AxmMotSaveParaAll](#), [AxmMotGetParaLoad](#), [AxmMotSetParaLoad](#),
[AxmMotSetPulseOutMethod](#), [AxmMotGetPulseOutMethod](#), [AxmMotGetEncInputMethod](#),
[AxmMotSetMoveUnitPerPulse](#), [AxmMotGetMoveUnitPerPulse](#), [AxmMotSetDecelMode](#),
[AxmMotGetDecelMode](#), [AxmMotSetSRate](#), [AxmMotGetSRate](#), [AxmMotSetRemainPulse](#),
[AxmMotGetRemainPulse](#),

AxmMotGetEncInputMethod

Purpose

Return Encoder input method setting on a specific axis

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		Channel (axis) number(starting from 0)
Method	out	DWORD*	3	Encoder input method

Method

#define	Value	Explanation
ObverseUpDownMode	00h	Forward direction Up / Down
ObverseSqr1Mode	01h	Forward direction 1 multiplication
ObverseSqr2Mode	02h	Forward direction 2 multiplication
ObverseSqr4Mode	03h	Forward direction 4 multiplication
ReverseUpDownMode	04h	Reverse direction Up / Down
ReverseSqr1Mode	05h	Reverse direction 1 multiplication
ReverseSqr2Mode	06h	Reverse direction 2 multiplication
ReverseSqr4Mode	07h	Reverse direction 4 multiplication

Return AXT_RT_SUCCESS(0000) : Successful execution of API.

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

Description

Return encoder input method setting on a specific axis set with [AxmMotSetEncInputMethod](#).

C Example

```
/// Set Encoder input method to 4 multiplication method on axis 0
AxmMotSetEncInputMethod (0, ObverseSqr4Mode);

// Verify Encoder input method on axis 0
DWORD Method;
AxmMotGetEncInputMethod (0, &Method);
```

VB Example

```
'Set Encoder input method to 4 multiplication method on axis 0
AxmMotSetEncInputMethod0, ObverseSqr4Mode

'Verify Encoder input method on axis 0
Dim Method As Long
AxmMotGetEncInputMethod 0, Method
```

Delphi Example

```
{ Verify Encoder input method on axis 0 }
var
Method: DWord;

Begin
{ Set Encoder input method to 4 multiplication method on axis 0 }
AxmMotSetEncInputMethod(0, ObverseSqr4Mode);

AxmMotGetEncInputMethod (0, @Method);
end;
```

See Also

[AxmMotLoadParaAll](#), [AxmMotSaveParaAll](#), [AxmMotGetParaLoad](#), [AxmMotSetParaLoad](#),
[AxmMotSetPulseOutMethod](#), [AxmMotGetPulseOutMethod](#), [AxmMotSetEncInputMethod](#),
[AxmMotGetEncInputMethod](#), [AxmMotSetMoveUnitPerPulse](#), [AxmMotGetMoveUnitPerPulse](#)
[AxmMotSetDecelMode](#), [AxmMotGetDecelMode](#), [AxmMotSetSRate](#), [AxmMotGetSRate](#),
[AxmMotSetRemainPulse](#), [AxmMotGetRemainPulse](#)

AxmMotSetMoveUnitPerPulse

Purpose

Set output pulse by moving distance on a specific axis.

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

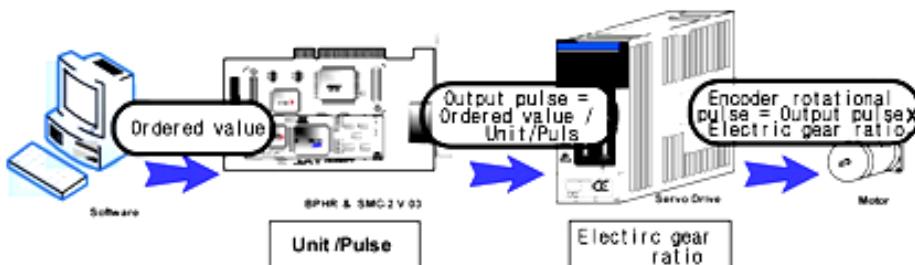
Name	in/out	Format	Init Value	Explanation
AxisNo	in	long		Channel (axis) number(starting from 0)
dUnit	In	double	1	Unit moving distance
lPulse	In	long	1	Unit pulse value

Return AXT_RT_SUCCESS(0000) : Successful execution of API.

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

Description

Basically, ordered value of moving distance that user assigns on the commands of library takes two steps of unit conversion as the picture below.



If moving amount is inputted on the software, number of output pulse of motion controller by parameter which is unit/pulse and already set is determined. And this output pulse is converted again by parameter which is electric gear ration set on the servo drive, and the drive pulse that drives actual motor is determined.

Unit/pulse is a parameter that is to assign unit of values given by running the motion drive API, and the API which sets this value is [AxmMotSetMoveUnitPerPulse](#). In other words, this means if it is set 10 on unit and 100 on pulse, when 10 is commanded on the motion drive API, 100 pulses are out to the servo drive.

*** description of Electronic gear Ratio of servo drive**

Unit/pulse is a parameter that is set on the motion controller, but electric gear ratio is a parameter that is set on the servo drive..

$$[\text{Electric gear ratio}] = \frac{[\text{Number of encoder pulse that is generated during 1 rotation of servo motor}]}{[\text{Number of output ordered pulse of motion controller during 1 rotation of servo motor}]}$$

If 17-bit encoder is used, 131,072 of encoder pulse number can be reduced to 10,000 on the motion controller based on above. That is, what a rotational accuracy of servo motor encoder lowers by an accuracy of mechanical part is an electric gear ratio.

For example, in case of 20mm ball screw, 1/2 of electric gear ratio, and 10000 of pulse number is required for 1 rotation of motor, if it is set that IPulse equals 10000 and dUnit equals 10, pulse number that outputs per moving distance of 1mm is set by 1000.



Unit / Pulse = 10 / 10000 Set

1 mm Control Use

Actual pulse output velocity for logical unit velocity sets Pulses/sec. Logical unit velocity means unit amount for velocity and acceleration used in a velocity specific API. But, unit of velocity may be different by user characteristic, i.e. it may be easy to indicate velocity unit with RPM for one user, and is easy to indicate with m/sec for another user. If 3600-pulse is required pulse number for 1 rotation, input 3600 into argument IPulse. 3600/60 which is 60PPS is set internally because RPM is rotation number per minute, therefore, divided by 60. Pulse to output will be 3600-pulse per minute.

If need to set setting velocity unit to RPM(Revolution Per Minute).

ex> rpm calculation:

4500 rpm ?

If unit/ pulse = 1 : 1, it will be pulse/ sec

If need to set to 4500 rpm, $4500 / 60 \text{ sec} : 75 \text{ rpm/ 1 sec}$

Need to know motor has which pulse per rotation. This will be known search Z phase in Encoder.

If 1 rotation:1800 pulse, it needs $75 \times 1800 = 135000$ pulse.

Make to move inputting Unit = 1, Pulse = 1800 in [AxmMotSetMoveUnitPerPulse](#).

C Example

```
// Set pulse number outputted by moving distance on axis 0 to 100
AxmMotSetMoveUnitPerPulse(0, 1, 100);

// Return pulse number outputted by moving distance on axis 0
long lpPulse;
double dpUnit;
AxmMotGetMoveUnitPerPulse (0, &dpUnit, &lpPulse);
```

VB Example

```
'Set pulse number outputted by moving distance on axis 0 to 100
AxmMotSetMoveUnitPerPulse 0, 1, 100

'Return pulse number outputted by moving distance on axis 0
Dim lpPulse As Long
Dim dpUnit As Double
AxmMotGetMoveUnitPerPulse 0, dpUnit, lpPulse
```

Delphi Example

```
{ Return pulse number outputted by moving distance on axis 0 }
var
lpPulse: LongInt;
dpUnit: Double;

begin
{ Set pulse number outputted by moving distance on axis 0 to 100 }
AxmMotSetMoveUnitPerPulse(0, 1, 100);
AxmMotGetMoveUnitPerPulse (0, @dpUnit, @lpPulse);
end;
```

See Also

[AxmMotLoadParaAll](#), [AxmMotSaveParaAll](#), [AxmMotGetParaLoad](#), [AxmMotSetParaLoad](#),
[AxmMotSetPulseOutMethod](#), [AxmMotGetPulseOutMethod](#), [AxmMotSetEnclInputMethod](#),
[AxmMotGetEnclInputMethod](#), [AxmMotGetMoveUnitPerPulse](#), [AxmMotSetDecelMode](#),
[AxmMotGetDecelMode](#), [AxmMotSetSRate](#), [AxmMotGetSRate](#), [AxmMotSetRemainPulse](#),
[AxmMotGetRemainPulse](#),

AxmMotGetMoveUnitPerPulse

Purpose

Return pulse outputted by moving distance on a specific axis

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		Channel (axis) number(starting from 0)
dUnit	out	double*	1	Unit moving distance
lpPulse	out	long*	1	Unit pulse value

Return AXT_RT_SUCCESS(0000) : Successful execution of API.

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

Description

Return pulse outputted by moving distance on a specific axis set with [AxmMotSetMoveUnitPerPulse](#)

*Note: When only required data are needed to bring, it is able to use with input NULL in no required variable.

C Example

```
// Set pulse number outputted by moving distance on axis 0 to 100
AxmMotSetMoveUnitPerPulse(0, 1, 100);

// Return pulse number outputted by moving distance on axis 0
long lpPulse;
double dpUnit;
AxmMotGetMoveUnitPerPulse (0, &dpUnit, &lpPulse);
```

VB Example

```
'Set pulse number outputted by moving distance on axis 0 to 100
AxmMotSetMoveUnitPerPulse 0, 1, 100

'Return pulse number outputted by moving distance on axis 0
Dim lpPulse As Long
Dim dpUnit As Double
AxmMotGetMoveUnitPerPulse 0, dpUnit, lpPulse
```

Delphi Example

```
{ Return pulse number outputted by moving distance on axis 0 }
var
lpPulse: LongInt;
dpUnit: Double;

begin
{ Set pulse number outputted by moving distance on axis 0 to 100 }
AxmMotSetMoveUnitPerPulse (0, 1, 100);
AxmMotGetMoveUnitPerPulse (0, @dpUnit, @lpPulse);
end;
```

See Also

[AxmMotLoadParaAll](#), [AxmMotSaveParaAll](#), [AxmMotGetParaLoad](#), [AxmMotSetParaLoad](#),
[AxmMotSetPulseOutMethod](#), [AxmMotGetPulseOutMethod](#), [AxmMotSetEnclInputMethod](#),
[AxmMotGetEnclInputMethod](#), [AxmMotSetMoveUnitPerPulse](#), [AxmMotSetDecelMode](#),
[AxmMotGetDecelMode](#), [AxmMotSetSRate](#), [AxmMotGetSRate](#), [AxmMotSetRemainPulse](#),
[AxmMotGetRemainPulse](#),

AxmMotSetDecelMode

Purpose

Set deceleration start point search method on a specific axis.

Format

C

```
DWORD AxmMotSetDecelMode (long lAxisNo, DWORD uMethod);
```

Visual Basic

```
Function AxmMotSetDecelMode (ByVal lAxisNo As Long, ByVal uMethod As Long) As Long
```

Delphi

```
function AxmMotSetDecelMode (lAxisNo : LongInt; uMethod : DWord) : DWord; stdcall;
```

Input / Output

Name	in/out	Format	Init Value	Explanation
AxisNo	in	long		Channel (axis) number(starting from 0)
Method	In	DWORD		Deceleration start point search method

Method

#define	Value	Explanation
AutoDetect	00h	Automatic search method
RestPulse	01h	Rest search method

Return AXT_RT_SUCCESS(0000) : Successful execution of API.

[* See error code Table for more information on status error codes](#)

Description

There are 2 search methods; automatic search (Autodetect) and rest search method (RestPulse). Initial value is set to automatic one.

C Example

```
// Set deceleration start point search method on axis 0
AxmMotSetDecelMode(0, AutoDetect);

// Verify deceleration start point search method on axis 0
DWORD Method;
AxmMotGetDecelMode(0, &Method);
```

VB Example

```
'Set deceleration start point search method on axis 0
AxmMotSetDecelMode 0, AutoDetect

'Verify deceleration start point search method on axis 0
Dim Method As Long
AxmMotGetDecelMode0, Method
```

Delphi Example

```
{ Verify deceleration start point search method on axis 0 }
var
Method: DWord;

Begin
{ Set deceleration start point search method on axis 0 }
AxmMotSetDecelMode(0, AutoDetect);

AxmMotGetDecelMode (0, @Method);
end;
```

See Also

[AxmMotLoadParaAll](#), [AxmMotSaveParaAll](#), [AxmMotGetParaLoad](#), [AxmMotSetParaLoad](#),
[AxmMotSetPulseOutMethod](#), [AxmMotGetPulseOutMethod](#), [AxmMotSetEnclInputMethod](#),
[AxmMotGetEnclInputMethod](#), [AxmMotSetMoveUnitPerPulse](#), [AxmMotGetMoveUnitPerPulse](#),
[AxmMotGetDecelMode](#), [AxmMotSetSRate](#), [AxmMotGetSRate](#), [AxmMotSetRemainPulse](#),
[AxmMotGetRemainPulse](#),

AxmMotGetDecelMode

Purpose

Return deceleration start point search method on a specific axis

Format

C

```
DWORD AxmMotGetDecelMode (long lAxisNo, DWORD *upMethod);
```

Visual Basic

```
Function AxmMotGetDecelMode (ByVal lAxisNo As Long, ByRef upMethod As Long) As Long
```

Delphi

```
function AxmMotGetDecelMode (lAxisNo : LongInt; upMethod : PDWord) : DWord; stdcall;
```

Input / Output

Name	in/out	Format	Init Value	Explanation
AxisNo	in	long		Channel (axis) number(starting from 0)
Method	out	long*		Deceleration start point search method

Method

#define	Value	Explanation
AutoDetect	00h	Automatic search method
RestPulse	01h	Rest search method

Return AXT_RT_SUCCESS(0000) : Successful execution of API.

[* See error code Table for more information on status error codes](#)

Description

Return deceleration start point search method on a specific axis set with [AxmMotSetDecelMode](#)

C Example

```
// Set search start point deceleration method
AxmMotSetDecelMode(0, AutoDetect);

// Verify deceleration start point search method on axis 0
DWORD Method;
AxmMotGetDecelMode(0, &Method);
```

VB Example

```
'Set search start point deceleration method
AxmMotSetDecelMode 0, AutoDetect

'Verify deceleration start point search method on axis 0
Dim Method As Long
AxmMotGetDecelMode 0, Method
```

Delphi Example

```
{ Verify deceleration start point search method on axis 0 }
var
Method: DWord;

Begin
{ Set search start point deceleration method }
AxmMotSetDecelMode(0, AutoDetect);

AxmMotGetDecelMode(0, @Method);
end;;
```

See Also

[AxmMotLoadParaAll](#), [AxmMotSaveParaAll](#), [AxmMotGetParaLoad](#), [AxmMotSetParaLoad](#),
[AxmMotSetPulseOutMethod](#), [AxmMotGetPulseOutMethod](#), [AxmMotSetEnclInputMethod](#),
[AxmMotGetEnclInputMethod](#), [AxmMotSetMoveUnitPerPulse](#), [AxmMotGetMoveUnitPerPulse](#)
[AxmMotSetDecelMode](#), [AxmMotSetSRate](#), [AxmMotGetSRate](#), [AxmMotSetRemainPulse](#),
[AxmMotGetRemainPulse](#),

AxmMotSetRemainPulse

Purpose

Set remaining pulse on manual deceleration mode on a specific axis.

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		Channel (axis) number(starting from 0)
Data	In	DWORD		Set remaining pulse

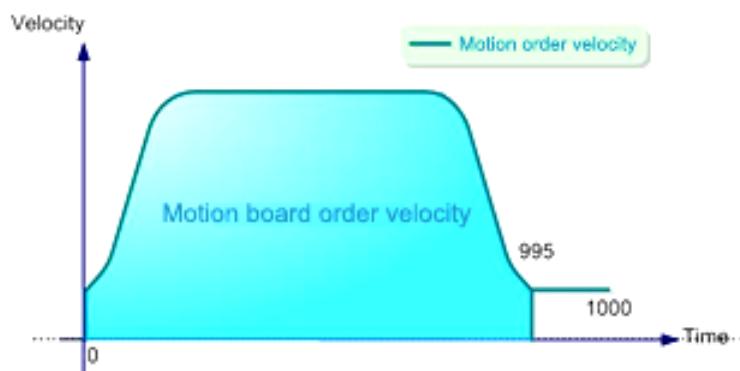
Return AXT_RT_SUCCESS(0000) : Successful execution of API.

[* See error code Table for more information on status error codes](#)

Description

When the deceleration mode is set to AutoDetect (0) by the [AxmMotSetDecelMode](#) function, this function is used to set the drive pulses after reducing the velocity of the motion.

For example, when the system is configured to move by 1000 pulses with 5 of remaining pulses using the AxmMoveStartPos function, the system will decelerate to the velocity configured by AxmMotSetMinVel function from the 995th pulse, and decelerate for the period of 5 pulses by keeping the specified velocity. This function is used to maintain deceleration at the end..



When the deceleration mode is set to RestPulse (1) by the [AxmMotSetDecelMode](#) function, this function is to set the number of remaining pulses for the deceleration segment.

For example, when the system is configured to move by 1000 pulses with 100 pulses required during acceleration, the operation is as followings depending on the configuration by the AxmMotSetRemainPulse function.

When the remaining pulse setting value is smaller than 100 : The drive will be completed before decelerating to the start stop velocity.

When the remaining pulse setting value is bigger than 100 : The drive will be completed after decelerating to start stop velocity and moving by the value bigger than 100.

Note) Since input value is pulse unit, position value (unit) should be converted to pulse unit by using the value set by [AxmMotSetMoveUnitPerPulse](#).

C Example

```
// Set remaining pulse on manual deceleration mode on axis 0
AxmMotSetRemainPulse(0, 5);

AxmMoveStartPos (0, 1000, 100, 200, 200);

// Verify remaining pulse on manual deceleration mode on axis 0
DWORD Data;
AxmMotGetRemainPulse (0, &Data);
```

VB Example

```
Set remaining pulse on manual deceleration mode on axis 0
AxmMotSetRemainPulse 0, 5

AxmMoveStartPos 0, 1000, 100, 200, 200

'Verify remaining pulse on manual deceleration mode on axis 0
Dim Data As Long
AxmMotGetRemainPulse 0, Data
```

Delphi Example

```
Verify remaining pulse on manual deceleration mode on axis 0 }
var
Data: DWord;

Begin
{ Set remaining pulse on manual deceleration mode on axis 0 }
AxmMotSetRemainPulse(0, 5);

AxmMoveStartPos (0, 1000, 100, 200, 200);

AxmMotGetRemainPulse(0, @Data);
end;
```

See Also

[AxmMotLoadParaAll](#), [AxmMotSaveParaAll](#), [AxmMotGetParaLoad](#), [AxmMotSetParaLoad](#),
[AxmMotSetPulseOutMethod](#), [AxmMotGetPulseOutMethod](#), [AxmMotSetEnclInputMethod](#),
[AxmMotGetEnclInputMethod](#), [AxmMotSetMoveUnitPerPulse](#), [AxmMotGetMoveUnitPerPulse](#)
[AxmMotSetDecelMode](#), [AxmMotGetDecelMode](#), [AxmMotSetSRate](#), [AxmMotGetSRate](#),
[AxmMotGetRemainPulse](#),

AxmMotGetRemainPulse

Purpose

Return remaining pulse on manual deceleration mode on a specific axis.

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		Channel (axis) number(starting from 0)
Data	out	DWORD*		Set remaining pulse

Return AXT_RT_SUCCESS(0000) : Successful execution of API.

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

Description

Return remaining pulse on manual deceleration mode on a specific axis using [AxmMotSetRemainPulse](#) API.

C Example

```
// Set remaining pulse on manual deceleration mode on axis 0
AxmMotSetRemainPulse (0, 1000);

// Verify remaining pulse on manual deceleration mode on axis 0
DWORD Data;
AxmMotGetRemainPulse (0, &Data);
```

VB Example

```
'Set remaining pulse on manual deceleration mode on axis 0
AxmMotSetRemainPulse 0, 1000

'Verify remaining pulse on manual deceleration mode on axis 0
Dim Data As Long
AxmMotGetRemainPulse 0, Data
```

Delphi Example

```
{ Verify remaining pulse on manual deceleration mode on axis 0 }
var
Data: DWord;
```

```
Begin
{ Set remaining pulse on manual deceleration mode on axis 0 }
AxmMotSetRemainPulse (0, 1000);
AxmMotGetRemainPulse (0, @Data);
end;
```

See Also

[AxmMotLoadParaAll](#), [AxmMotSaveParaAll](#), [AxmMotGetParaLoad](#), [AxmMotSetParaLoad](#),
[AxmMotSetPulseOutMethod](#), [AxmMotGetPulseOutMethod](#), [AxmMotSetEnclInputMethod](#),
[AxmMotGetEnclInputMethod](#), [AxmMotSetMoveUnitPerPulse](#), [AxmMotGetMoveUnitPerPulse](#)
[AxmMotSetDecelMode](#), [AxmMotGetDecelMode](#), [AxmMotSetSRate](#), [AxmMotGetSRate](#),
[AxmMotSetRemainPulse](#)

AxmMotSetMaxVel

Purpose

Limit maximum drive velocity of all drive API to assigned velocity.

Format

C

```
DWORD AxmMotSetMaxVel (long lAxisNo, double dVel);
```

Visual Basic

```
Function AxmMotSetMaxVel (ByVal lAxisNo As Long, ByVal dVel As Double) As Long
```

Delphi

```
function AxmMotSetMaxVel (lAxisNo : LongInt; dVel : Double): DWord; stdcall;
```

Input / Output

Name	in/out	Format	Init Value	Explanation
AxisNo	in	long		Channel (axis) number(starting from 0)
Vel	In	double	700000	Maximum drive velocity (unit of drive velocity is PPS[Pulses/sec] if Unit/pulse is 1/1)

Return AXT_RT_SUCCESS(0000) : Successful execution of API.

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

Description

This function limits all input velocity values of drive functions for the specified axis to the velocity set by the [AxmMotSetMaxVel](#) function.

Note: The velocity can be set to up to 4MPPS for SMV-V203 product, and up to 10 MPPS for PCI-Nx04 products. The input velocity to the AxmMotSetMaxVel function is applied differently depending on the Unit/Pulse value set by [AxmMotSetMoveUnitPerPulse](#) API. For example, the maximum value that can be set is (4M/100) for SMC-2V03 and (10M/100) for PCI-Nx04 if the Unit/Pulse value is 0.01.

C Example

```
// Set maximum velocity of 1000 for constant speed drive on axis 0
AxmMotSetMaxVel (0, 1000);

// Verify constant speed drive maximum velocity on axis 0
double Vel;
AxmMotGetMaxVel(0, &Vel);
```

VB Example

```
'Set maximum velocity of 1000 for constant speed drive on axis 0
AxmMotSetMaxVel 0, 1000

'Verify constant speed drive maximum velocity on axis 0
Dim Vel As Double
```

```
AxmMotGetMaxVel 0, Vel
```

Delphi Example

```
{ Verify constant speed drive maximum velocity on axis 0 }
var
Vel: Double;

Begin
{ Set maximum velocity of 1000 for constant speed drive on axis 0 }
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

Return maximum velocity in constant speed drive API on a specific axis

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		Channel (axis) number(starting from 0)
Vel	out	double*	700000	Maximum drive velocity (unit of drive velocity is PPS[Pulses/sec] if Unit/pulse is 1/1)

Return AXT_RT_SUCCESS(0000) : Successful execution of API.

[* See error code Table for more information on status error codes](#)

Description

Return maximum velocity in constant speed drive API on a specific axis set in [AxmMotSetMaxVel](#)

C Example

```
// Set maximum velocity of 1000 for constant speed drive on axis 0
AxmMotSetMaxVel (0, 1000);

// Verify constant speed drive maximum velocity on axis 0
double Vel;
AxmMotGetMaxVel (0, &Vel);
```

VB Example

```
'Set maximum velocity of 1000 for constant speed drive on axis 0
AxmMotSetMaxVel 0, 1000

'Verify constant speed drive maximum velocity on axis 0
Dim Vel As Double
AxmMotGetMaxVel 0, Vel
```

Delphi Example

```
{ Verify constant speed drive maximum velocity on axis 0 }
var
Vel: Double;
```

```
Begin
{ Set maximum velocity of 1000 for constant speed drive on axis 0 }
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

Set moving distance calculation mode on a specific axis.

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		Channel (axis) number(starting from 0)
AbsRelMode	In	DWORD	0	Pulse output method

AbsRelMode

#define	Value	Explanation
POS_ABS_MODE	00h	Position drive absolute mode
POS_REL_MODE	01h	Position drive relative mode

Return AXT_RT_SUCCESS(0000) : Successful execution of API.

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

Description

If absolute position mode (0) is set, position value used in moving API is to be absolute position value.

For example, if current position value is “10” and moves from absolute position mode to “100” position, it will stop at “100” position moving as much as “+90” from the current position.

If relative position mode (1) is set, position value used moving API is to be moving amount to move from the current position. For example, if current position value is “10” and moves from relative position mode to “100” position, it will stop at “110” position moving as much as “+100” from the current position.

C Example

```
/ Set axis 0 to absolute mode.
AxmMotSetAbsRelMode(0, POS_ABS_MODE);

// Verify moving mode of axis 0.
DWORD Mode;
AxmMotGetAbsRelMode(0, &Mode);
```

VB Example

```
'Set axis 0 to absolute mode.  
AxmMotSetAbsRelMode 0, POS_ABS_MODE  
  
'Verify moving mode of axis 0  
Dim Mode As Long  
AxmMotGetAbsRelMode0, Mode
```

Delphi Example

```
{ Verify moving mode of axis 0 }  
var  
Mode : DWord;  
  
Begin  
{ Set axis 0 to absolute mode. }  
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

Return moving distance calculation mode which is set on a specific axis.

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		Channel (axis) number(starting from 0)
AbsRelMode	out	DWORD*	0	Pulse output method

AbsRelMode

#define	Value	Explanation
POS_ABS_MODE	00h	Position drive absolute mode
POS_REL_MODE	01h	Position drive relative mode

Return AXT_RT_SUCCESS(0000) : Successful execution of API.

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

Description

Return moving distance calculation mode which is set with [AxmMotSetAbsRelMode](#) on a specific axis.

C Example

```
// Set axis 0 to absolute mode.
AxmMotSetAbsRelMode (0, POS_ABS_MODE);

// Verify moving mode of axis 0.
DWORD Mode;
AxmMotGetAbsRelMode (0, &Mode);
```

VB Example

```
'Set axis 0 to absolute mode.
AxmMotSetAbsRelMode 0, POS_ABS_MODE

'Verify moving mode of axis 0
Dim Mode As Long
AxmMotGetAbsRelMode 0, Mode
```

Delphi Example

```
{ Verify moving mode of axis 0 }
var
Mode : DWord;

Begin
{ Set axis 0 to absolute mode. }
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

Set drive velocity profile mode on a specific axis.

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		Channel (axis) number(starting from 0)
ProfileMode	In	DWORD	4	Velocity profile mode

ProfileMode

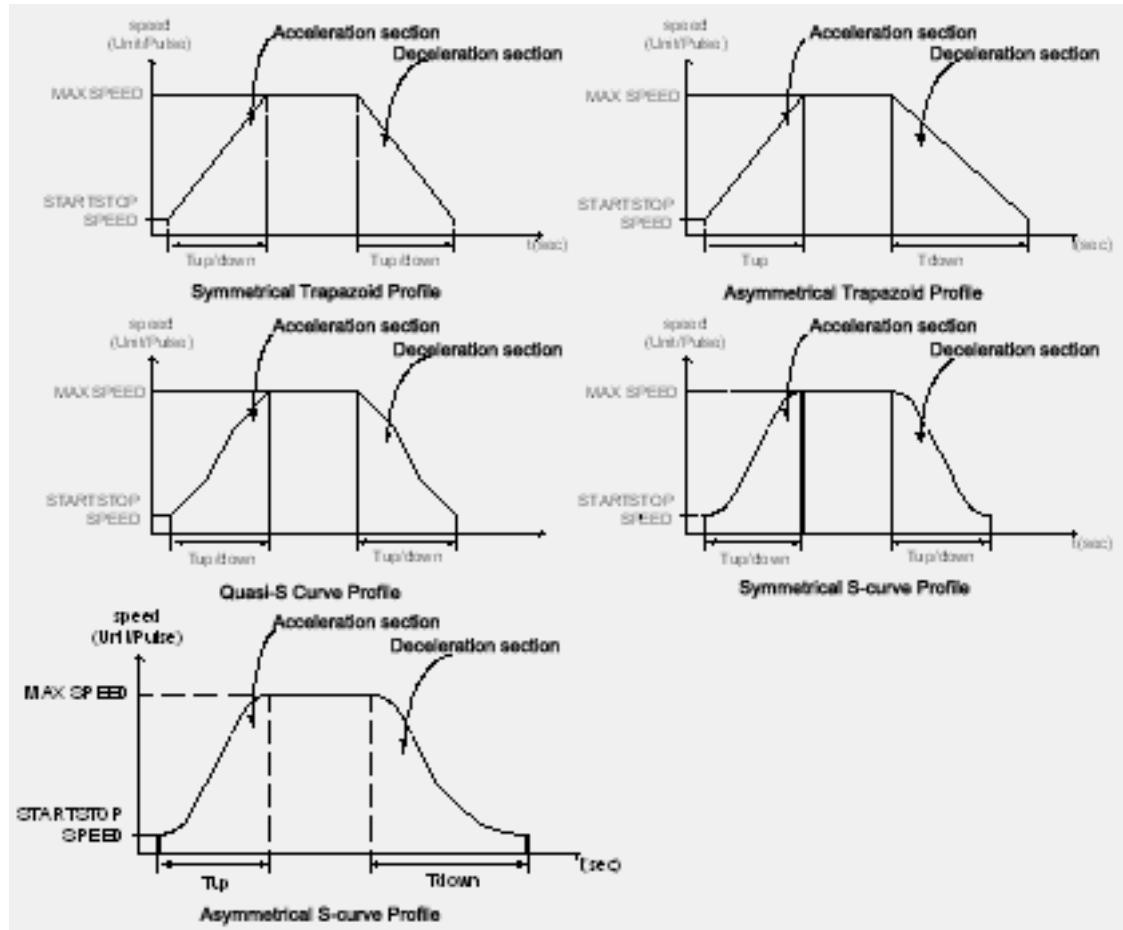
#define	Value	Explanation
SYM_TRAPEZOIDE_MODE	00h	Symmetrical Trapezode
ASYM_TRAPEZOIDE_MODE	01h	Asymmetrical Trapezode
QUASI_S_CURVE_MODE	02h	Symmetrical Quasi-S Curve
SYM_S_CURVE_MODE	03h	Symmetrical S Curve
ASYM_S_CURVE_MODE	04h	Asymmetrical S Curve

Return AXT_RT_SUCCESS(0000) : Successful execution of API.

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

Description

Velocity profile mode has 5 modes; symmetrical Trapezoid , asymmetrical Trapezoid, Quasi-S Curve mode, symmetrical S-curve and asymmetrical S-curve mode. Graph of each velocity profile mode shows below.

**C Example**

```
// Set velocity profile on axis 0 to symmetrical S-Curve.
AxmMotSetProfileMode (0, 3);

// Verify velocity profile on axis 0.
DWORD uProfile;
AxmMotGetProfileMode(0, &uProfile);
```

VB Example

```
'Set velocity profile on axis 0 to symmetrical S-Curve.
AxmMotSetProfileMode 0, 3

'Verify velocity profile on axis 0.
Dim uProfile As Long
AxmMotGetProfileMode 0, uProfile
```

Delphi Example

```
{ Verify velocity profile on axis 0. }
var
uProfile : DWord;

begin
{ Set velocity profile on axis 0 to symmetrical S-Curve. }
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

Return drive velocity profile which is set on a specific axis.

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		Channel (axis) number(starting from 0)
ProfileMode	out	DWORD*	4	Velocity profile mode

ProfileMode

#define	Value	Explanation
SYM_TRAPEZOIDE_MODE	00h	Symmetrical Trapezode
ASYM_TRAPEZOIDE_MODE	01h	Asymmetrical Trapezode
QUASI_S_CURVE_MODE	02h	Symmetrical Quasi-S Curve
SYM_S_CURVE_MODE	03h	Symmetrical S Curve
ASYM_S_CURVE_MODE	04h	Asymmetrical S Curve

Return AXT_RT_SUCCESS(0000) : Successful execution of API.

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

Description

Return drive velocity profile which is set with [AxmMotSetProfileMode](#) on a specific axis.

C Example

```
// Set velocity profile on axis 0 to symmetrical S-Curve.
AxmMotSetProfileMode (0, 3);

// Verify velocity profile on axis 0.
DWORD uProfile;
AxmMotGetProfileMode (0, &uProfile);
```

VB Example

```
'Set velocity profile on axis 0 to symmetrical S-Curve.
AxmMotSetProfileMode0, 3
```

```
'Verify velocity profile on axis 0.  
Dim uProfile As Long  
AxmMotGetProfileMode 0, uProfile
```

Delphi Example

```
{ Verify velocity profile on axis 0. }  
var  
uProfile : DWord;  
  
begin  
{ Set velocity profile on axis 0 to symmetrical S-Curve. }  
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

Set acceleration unit on a specific axis.

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		Channel (axis) number(starting from 0)
AccelUnit	in	DWORD		Acceleration (unit of acceleration is PPS[Pulses/sec^2] if Unit/pulse is 1/1)

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 for more information on status error codes

Description

Acceleration on a specific axis can be set to unit/sec² calculated based on acceleration rate of moving velocity, and be set acceleration time sec (1).

C Example

```
// Set acceleration unit of unit/sec2on axis 0.
AxmMotSetAccelUnit (0, UNIT_SEC2);

// Verify acceleration unit on axis 0.
DWORD uAccelUnit;
AxmMotGetAccelUnit 0, &uAccelUnit);
```

VB Example

```
'Set acceleration unit of unit/sec2on axis 0.
AxmMotSetAccelUnit 0, UNIT_SEC2

'Verify acceleration unit on axis 0.
```

```
Dim uAccelUnit As Long  
AxmMotGetAccelUnit 0, uAccelUnit
```

Delphi Example

```
{ Verify acceleration unit on axis 0. }  
var  
uAccelUnit: DWord;  
  
begin  
{ Set acceleration unit of unit/sec2on axis 0. }  
AxmMotSetAccelUnit (0, UNIT_SEC2);  
  
AxmMotGetAccelUnit (0, @uAccelUnit);  
end;
```

See Also

[AxmMotLoadParaAll](#), [AxmMotSaveParaAll](#), [AxmMotGetParaLoad](#), [AxmMotSetParaLoad](#),
[AxmMotSetMaxVel](#), [AxmMotGetMaxVel](#), [AxmMotSetAbsRelMode](#), [AxmMotGetAbsRelMode](#),
[AxmMotSetProfileMode](#), [AxmMotGetProfileMode](#), [AxmMotGetAccelUnit](#), [AxmMotSetMinVel](#),
[AxmMotGetMinVel](#), [AxmMotSetAccelJerk](#), [AxmMotGetAccelJerk](#), [AxmMotSetDecelJerk](#),
[AxmMotGetDecelJerk](#)

AxmMotGetAccelUnit

Purpose

Return acceleration unit which is set on a specific axis.

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		Channel (axis) number(starting from 0)
AccelUnit	out	DWORD*		Acceleration (unit of acceleration is PPS[Pulses/sec^2] if Unit/pulse is 1/1)

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 for more information on status error codes](#)

Description

Return acceleration unit which is set with [AxmMotSetAccelUnit](#) on a specific axis.

C Example

```
// Set acceleration unit of unit/sec on axis 0.  
AxmMotSetAccelUnit 0, UNIT_SEC2;  
  
// Verify acceleration unit on axis 0.  
DWORD uAccelUnit;  
AxmMotGetAccelUnit (0, &uAccelUnit);
```

VB Example

```
'Set acceleration unit of unit/sec2on axis 0.  
AxmMotSetAccelUnit 0, UNIT_SEC2  
  
'Verify acceleration unit on axis 0.  
Dim uAccelUnit As Long  
AxmMotGetAccelUnit 0, uAccelUnit
```

Delphi Example

```
{ Verify acceleration unit on axis 0. }  
var  
uAccelUnit: DWord;  
  
begin  
{ Set acceleration unit of unit/sec2on axis 0. }  
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

Set initial velocity on a specific axis

Format

C

```
DWORD AxmMotSetMinVel (long lAxisNo, double dMinVelocity);
```

Visual Basic

```
Function AxmMotSetMinVel (ByVal lAxisNo As Long, ByVal dMinVelocity As Double) As Long
```

Delphi

```
function AxmMotSetMinVel (lAxisNo : LongInt; dMinVelocity : Double): DWord; stdcall;
```

Input/Output

Name	in/out	Format	Init Value	Explanation
AxisNo	in	long		Channel (axis) number(starting from 0)
MinVelocity	In	double	1	Initial velocity (unit of initial velocity is PPS[Pulses/sec] if Unit/pulse is 1/1)

Return AXT_RT_SUCCESS(0000) : Successful execution of API.

[* See error code Table for more information on status error codes](#)

Description

Minimum velocity means start velocity of acceleration and end velocity of deceleration. If the minimum velocity is set to higher than working velocity, the system starts with the existing minimum velocity and reaches to the working velocity, and then moves with the configured minimum velocity without acceleration. The function is not needed except special cases, and the default value of minimum velocity is 1. Deviation phenomenon could be avoided by setting the minimum velocity in case of using stepping motors.

Note: If the minimum velocity is set to lower than the value of Unit/Pulse value, minimum velocity will be set to the value of Unit/Pulse since minimum unit is set to the value of Unit/Pulse.

C Example

```
// Set minimum velocity to 1 on axis 0
AxmMotSetMinVel (0, 1);

// Verify minimum velocity on axis 0
double MinVelocity;
AxmMotGetMinVel (0, &MinVelocity);
```

VB Example

```
'Set minimum velocity to 1 on axis 0
AxmMotSetMinVel 0, 1

'Verify minimum velocity on axis 0
```

```
Dim MinVelocity As Double  
AxmMotGetMinVel 0, MinVelocity
```

Delphi Example

```
{ Verify minimum velocity on axis 0 }  
var  
  MinVelocity: Double;  
  
begin  
{ Set minimum velocity to 1 on axis 0 }  
AxmMotSetMinVel (0, 1);  
  
AxmMotGetMinVel (0, @MinVelocity)
```

See Also

[AxmMotLoadParaAll](#), [AxmMotSaveParaAll](#), [AxmMotGetParaLoad](#), [AxmMotSetParaLoad](#),
[AxmMotSetMaxVel](#), [AxmMotGetMaxVel](#), [AxmMotSetAbsRelMode](#), [AxmMotGetAbsRelMode](#), [AxmMotSetProfileMode](#),
[AxmMotGetProfileMode](#), [AxmMotSetAccelUnit](#), [AxmMotGetAccelUnit](#), [AxmMotGetMinVel](#), [AxmMotSetAccelJerk](#),
[AxmMotGetAccelJerk](#), [AxmMotSetDecelJerk](#), [AxmMotGetDecelJerk](#)

AxmMotGetMinVel

Purpose

Return initial velocity on a specific axis.

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		Channel (axis) number(starting from 0)
MinVelocity	out	double*	1	Initial velocity (unit of initial velocity is PPS[Pulses/sec] if Unit/pulse is 1/1)

Return AXT_RT_SUCCESS(0000) : Successful execution of API.

[* See error code Table for more information on status error codes](#)

Description

Return initial velocity set with [AxmMotSetMinVel](#) on a specific axis.

C Example

```
// Set initial velocity to 1 on axis 0
AxmMotSetMinVel (0, 1);

// Verify initial velocity on axis 0
double MinVelocity;
AxmMotGetMinVel (0, &MinVelocity);
```

VB Example

```
'Set initial velocity to 1 on axis 0
AxmMotSetMinVel0, 1

'Verify initial velocity on axis 0
Dim MinVelocity As Double
AxmMotGetMinVel 0, MinVelocity
```

Delphi Example

```
{ Verify initial velocity on axis 0 }
var
  MinVelocity: Double;
```

```
begin
{ Set initial velocity to 1 on axis 0 }
AxmMotSetMinVel (0, 1);

AxmMotGetMinVel (0, @MinVelocity)
end;
```

See Also

[AxmMotLoadParaAll](#), [AxmMotSaveParaAll](#), [AxmMotGetParaLoad](#), [AxmMotSetParaLoad](#),
[AxmMotSetMaxVel](#), [AxmMotGetMaxVel](#), [AxmMotSetAbsRelMode](#), [AxmMotGetAbsRelMode](#),
[AxmMotSetProfileMode](#), [AxmMotGetProfileMode](#), [AxmMotSetAccelUnit](#), [AxmMotGetAccelUnit](#),
[AxmMotSetMinVel](#), [AxmMotSetAccelJerk](#), [AxmMotGetAccelJerk](#), [AxmMotSetDecelJerk](#),
[AxmMotGetDecelJerk](#)

AxmMotSetAccelJerk

Purpose

Set acceleration jerk value on a specific axis.

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		Channel (axis) number(starting from 0)
AccelJerk	In	double		Accelerating jerk value, unit is %

Return AXT_RT_SUCCESS(0000) : Successful execution of API.

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

Description

Unit is %. If it drives with S-Curve velocity profile mode, it sets section to drive with S-Curve to % between maximum velocity and start velocity during acceleration.

C Example

```
/ Set jerk value on axis 0
AxmMotSetAccelJerk (0, 2000);
AxmMotSetDecelJerk (0, 2000);

// Verify jerk value on axis 0
double dAccelJerk, dDecelJerk;
AxmMotGetAccelJerk (0, &dAccelJerk);
AxmMotGetDecelJerk (0, &dDecelJerk);
```

VB Example

```
'Set jerk value on axis 0
AxmMotSetAccelJerk 0, 2000
AxmMotSetDecelJerk 0, 2000

'Verify jerk value on axis 0
Dim dAccelJerk , dDecelJerk As Double
AxmMotGetAccelJerk 0, dAccelJerk
AxmMotGetDecelJerk 0, dDecelJerk
```

Delphi Example

```
{ Verify jerk value on axis 0 }
var
dAccelJerk, dDecelJerk: Double;

Begin
{ Set jerk value on axis 0}
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

Return acceleration jerk value which is set on a specific axis.

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		Channel (axis) number(starting from 0)
AccelJerk	out	double*		Accelerating jerk value, unit is %

Return AXT_RT_SUCCESS(0000) : Successful execution of API.

[* See error code Table for more information on status error codes](#)

Description

Unit is %. If it drives with S-Curve velocity profile mode, it sets section to drive with S-Curve to % between maximum velocity and start velocity during acceleration.

C Example

```
// Set jerk value on axis 0
AxmMotSetAccelJerk (0, 2000);
AxmMotSetDecelJerk (0, 2000);

// Verify jerk value on axis 0
double dAccelJerk, dDecelJerk;
AxmMotGetAccelJerk (0, &dAccelJerk);
AxmMotGetDecelJerk (0, &dDecelJerk);
```

VB Example

```
'Set jerk value on axis 0
AxmMotSetAccelJerk 0, 2000
AxmMotSetDecelJerk 0, 2000

'Verify jerk value on axis 0
Dim dAccelJerk , dDecelJerk As Double
AxmMotGetAccelJerk 0, dAccelJerk
AxmMotGetDecelJerk 0, dDecelJerk
```

Delphi Example

```
{ Verify jerk value on axis 0 }
var
dAccelJerk, dDecelJerk: Double;

Begin
{ Set jerk value on axis 0 }
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

Set deceleration jerk value on a specific axis.

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		Channel (axis) number(starting from 0)
DecelJerk	In	double		Decelerating jerk value, unit is %

Return AXT_RT_SUCCESS(0000) : Successful execution of API.

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

Description

Unit is %. If it drives with S-Curve velocity profile mode, it sets section to drive with S-Curve to % between maximum velocity and start velocity during acceleration.

C Example

```
// Set jerk value on axis 0
AxmMotSetAccelJerk(0, 2000);
AxmMotSetDecelJerk(0, 2000);

// Verify jerk value on axis 0
double dAccelJerk, dDecelJerk;
AxmMotGetAccelJerk (0, &dAccelJerk);
AxmMotGetDecelJerk (0, &dDecelJerk);
```

VB Example

```
'Set jerk value on axis 0
AxmMotSetAccelJerk 0, 2000
AxmMotSetDecelJerk 0, 2000

'Verify jerk value on axis 0
Dim dAccelJerk , dDecelJerk As Double
AxmMotGetAccelJerk 0, dAccelJerk
AxmMotGetDecelJerk 0, dDecelJerk
```

Delphi Example

```
{ Verify jerk value on axis 0 }
var
dAccelJerk, dDecelJerk: Double;

begin
{ Set jerk value on axis 0 }
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

Return deceleration jerk value which is set on a specific axis.

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		Channel (axis) number(starting from 0)
DecelJerk	out	double*		Decelerating jerk value, unit is %

Return AXT_RT_SUCCESS(0000) : Successful execution of API.

[* See error code Table for more information on status error codes](#)

Description

Unit is %. If it drives with S-Curve velocity profile mode, it sets section to drive with S-Curve to % between maximum velocity and start velocity during acceleration.

C Example

```
// Set jerk value on axis 0
AxmMotSetAccelJerk(0, 2000);
AxmMotSetDecelJerk(0, 2000);

// Verify jerk value on axis 0
double dAccelJerk, dDecelJerk;
AxmMotGetAccelJerk (0, &dAccelJerk);
AxmMotSetDecelJerk (0, &dDecelJerk);
```

VB Example

```
'Set jerk value on axis 0
AxmMotSetAccelJerk 0, 2000
AxmMotSetDecelJerk 0, 2000

'Verify jerk value on axis 0
Dim dAccelJerk , dDecelJerk As Double
AxmMotGetAccelJerk 0, dAccelJerk
AxmMotSetDecelJerk 0, dDecelJerk
```

Delphi Example

```
{ Verify jerk value on axis 0 }
var
dAccelJerk, dDecelJerk: Double;
```

```
begin
{ Set jerk value on axis 0 }
AxmMotSetAccelJerk 0, 2000;
AxmMotSetDecelJerk(0, 2000);

AxmMotGetAccelJerk (0, @dAccelJerk);
AxmMotSetDecelJerk (0, @dDecelJerk);
end;
```

See Also

[AxmMotLoadParaAll](#), [AxmMotSaveParaAll](#), [AxmMotGetParaLoad](#), [AxmMotSetParaLoad](#),
[AxmMotSetMaxVel](#), [AxmMotGetMaxVel](#), [AxmMotSetAbsRelMode](#), [AxmMotGetAbsRelMode](#),
[AxmMotSetProfileMode](#), [AxmMotGetProfileMode](#), [AxmMotSetAccelUnit](#), [AxmMotGetAccelUnit](#),
[AxmMotSetMinVel](#), [AxmMotGetMinVel](#), [AxmMotSetAccelJerk](#), [AxmMotGetAccelJerk](#),
[AxmMotSetDecelJerk](#),

Input and Output Level Setting API

In this unit, level by external sensors, universal input and output, and level API related to servo drive will be introduced. If level is set incorrectly, motor may not operate. Therefore, set level correctly and start drive.

Function	Description
AxmSignalSetZphaseLevel	Sets Z phase level on a specific axis.
AxmSignalGetZphaseLevel	Returns Z phase level on a specific axis.
AxmSignalSetServoOnLevel	Sets output level of Amp-On signal on a specific axis.
AxmSignalGetServoOnLevel	Returns output level of Amp-On signal on a specific axis.
AxmSignalSetServoAlarmResetLevel	Sets output level of Servo-Alarm Reset signal on a specific axis.
AxmSignalGetServoAlarmResetLevel	Returns output level of Servo-Alarm Reset signal on a specific axis.
AxmSignalSetInpos	Sets whether Inpositon signal is used and signal input level on a specific axis.
AxmSignalGetInpos	Returns whether Inpositon signal is used and signal input level on a specific axis.
AxmSignalReadInpos	Returns Inpositon signal input state on a specific axis.
AxmSignalSetServoAlarm	Sets whether emergency stop is used and signal input level during alarm signal input on a specific axis.
AxmSignalGetServoAlarm	Returns whether emergency stop is used and signal input level during alarm signal input on a specific axis.
AxmSignalReadServoAlarm	Returns input level of alarm signal on a specific axis.
AxmSignalSetLimit	Sets whether end limit sensor is used and input level of signal on a specific axis. Available to set for decelerate stop or emergency stop during end limit sensor signal input.
AxmSignalGetLimit	Returns use of end limit sensor and signal input level and stop mode during signal input on a specific axis.
AxmSignalReadLimit	Returns input state of end limit sensor on a specific axis.
AxmSignalSetSoftLimit	Sets use of Software limit, count to use, and stop method on a specific axis.
AxmSignalGetSoftLimit	Returns use of Software limit, count to use, and stop method on a specific axis.
AxmSignalSetStop	Sets stop method (emergency stop/decelerate stop) or use

	of emergency stop signal.
<u>AxmSignalGetStop</u>	Returns stop method (emergency stop/decelerate stop) or use of emergency stop signal.
<u>AxmSignalReadStop</u>	Returns input state of emergency stop signal.

AxmSignalSetZphaseLevel

Purpose

Set output level of Z phase input signal on a specific axis

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		Channel (axis) number(starting from 0)
Level	In	DWORD	1	Active Level of Z phase signal input

Level

#define	Value	Explanation
LOW	00h	B contact (Normal Close)
HIGH	01h	A contact (Normal Open)

Return AXT_RT_SUCCESS(0000) : Successful execution of API.

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

Description

Set Z phase input level on an axis assigned by user.

Z phase signal means zero pulse signal of encoder. CAMC-IP, QI chip uses universal input 1 as Z phase of Encoder. If this signal is used, more accurate home all work can be executed when ORG signal is used.

If read Bit 8 using [AxmStatusReadMechanical](#) API to read Home Level, it can be verified if level is changed.

C Example

```
// Set Z phase level on axis 0.
AxmSignalSetZphaseLevel(0, HIGH);

// Verify Z phase level value on axis 0
DWORD state;

AxmSignalGetZphaseLevel 0, &state);
```

VB Example

```
'Set Z phase level on axis 0.  
AxmSignalSetZphaseLevel 0, HIGH  
  
'Verify Z phase level value on axis 0  
Dim state As Long  
  
AxmSignalGetZphaseLevel 0, state
```

Delphi Example

```
{ Verify z phase level value on axis 0 }  
var  
state: DWord;  
  
Begin  
{ Set z phase level on axis 0. }  
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

Return input level of Z phase signal on a specific axis.

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		Channel (axis) number(starting from 0)
Level	out	DWORD*	1	Active Level of Z phase signal input

Level

#define	Value	Explanation
LOW	00h	B contact (Normal Close)
HIGH	01h	A contact (Normal Open)

Return AXT_RT_SUCCESS(0000) : Successful execution of API.

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

Description

Return Z phase input level set with '[AxmSignalSetZphaseLevel](#)'.

C Example

```
// Set Z phase level on axis 0.
AxmSignalSetZphaseLevel(0, HIGH);

// Verify Z phase level value on axis 0
DWORD upLevel;

AxmSignalGetZphaseLevel (0, &upLevel);
```

VB Example

```
'Set Z phase level on axis 0.
AxmSignalSetZphaseLevel 0, HIGH

'Verify Z phase level value on axis 0
Dim state As Long
```

```
AxmSignalGetZphaseLevel 0, upLevel
```

Delphi Example

```
{ Verify Z phase level value on axis 0 }
var
state: DWord;

Begin
{ Set Z phase level on axis 0. }
AxmSignalSetZphaseLevel(0, HIGH);
AxmSignalGetZphaseLevel(0, @upLevel);
end;
```

See Also

[AxmSignalSetZphaseLevel](#), [AxmSignalSetServoOnLevel](#), [AxmSignalGetServoOnLevel](#),
[AxmSignalSetServoAlarmResetLevel](#), [AxmSignalGetServoAlarmResetLevel](#), [AxmSignalSetInpos](#),
[AxmSignalGetInpos](#), [AxmSignalReadInpos](#), [AxmSignalSetServoAlarm](#), [AxmSignalGetServoAlarm](#),
[AxmSignalReadServoAlarm](#), [AxmSignalSetLimit](#), [AxmSignalGetLimit](#), [AxmSignalReadLimit](#),
[AxmSignalSetSoftLimit](#), [AxmSignalGetSoftLimit](#), [AxmSignalSetStop](#), [AxmSignalGetStop](#),
[AxmSignalReadStop](#)

AxmSignalSetServoOnLevel

Purpose

Set output level of Servo-On signal on a specific axis.

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		Channel (axis) number(starting from 0)
Level	In	DWORD		Active Level of Servo-On signal

Level

#define	Value	Explanation
LOW	00h	B contact (Normal Close)
HIGH	01h	A contact (Normal Open)

Return AXT_RT_SUCCESS(0000) : Successful execution of API.

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

Description

Set output level of Servo-On signal on an axis assigned by user. Use to set level because every company may have different input level of Servo-Drive.

C Example

```
// Set Servo-On level on axis 0.
AxmSignalSetServoOnLevel(0, HIGH);

// Verify Servo-On level value on axis 0
DWORD uLevel;

AxmSignalGetServoOnLevel (0, & uLevel);
```

VB Example

```
Set Servo-On level on axis 0.
AxmSignalSetServoOnLevel 0, HIGH

'Verify Servo-On level value on axis 0
Dim uLevel As Long
```

```
AxmSignalGetServoOnLevel 0, state
```

Delphi Example

```
{ Verify Servo-On level value on axis 0 }
var
uLevel: DWord;

Begin
{ Set Servo-On level on axis 0. }
AxmSignalSetServoOnLevel (0, HIGH);
AxmSignalGetServoOnLevel (0, @uLevel);
end;
```

See Also

[AxmSignalSetZphaseLevel](#), [AxmSignalGetZphaseLevel](#), [AxmSignalGetServoOnLevel](#),
[AxmSignalGetServoAlarmResetLevel](#), [AxmSignalSetInpos](#), [AxmSignalGetInpos](#),
[AxmSignalReadInpos](#), [AxmSignalSetServoAlarm](#), [AxmSignalGetServoAlarm](#), [AxmSignalReadServoAlarm](#),
[AxmSignalSetLimit](#), [AxmSignalGetLimit](#), [AxmSignalReadLimit](#), [AxmSignalSetSoftLimit](#),
[AxmSignalGetSoftLimit](#), [AxmSignalSetStop](#), [AxmSignalGetStop](#), [AxmSignalReadStop](#)

AxmSignalGetServoOnLevel

Purpose

Return output level setting of Servo-On signal on a specific axis.

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		Channel (axis) number(starting from 0)
Level	out	DWORD*		Active Level of Servo-On signal

Level

#define	Value	Explanation
LOW	00h	B contact (Normal Close)
HIGH	01h	A contact (Normal Open)

Return AXT_RT_SUCCESS(0000) : Successful execution of API.

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

Description

Return output level setting of Servo-On signal set with '[AxmSignalSetServoOnLevel](#)'.

C Example

```
// Set Servo-On level on axis 0.
AxmSignalSetServoOnLevel 0, HIGH;

// Verify Servo-On level value on axis 0
DWORD state;

AxmSignalGetServoOnLevel (0, &state);
```

VB Example

```
'Set Servo-On level on axis 0.
AxmSignalSetServoOnLevel 0, HIGH

'Verify Servo-On level value on axis 0
Dim state As Long
```

```
AxmSignalGetServoOnLevel 0, state
```

Delphi Example

```
{ Verify Servo-On level value on axis 0 }
var
state: DWord;

Begin
{ Set Servo-On level on axis 0. }
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

Return output level setting of Servo-On signal set with '[AxmSignalSetServoOnLevel](#)'.

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		Channel (axis) number(starting from 0)
Level	In	DWORD		Active Level Servo-Alarm Reset signal

Level

#define	Value	Explanation
LOW	00h	B contact (Normal Close)
HIGH	01h	A contact (Normal Open)

Return AXT_RT_SUCCESS(0000) : Successful execution of API.

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

Description

When alarm occurred due to abnormal operation or other reason during motion drive, clear alarm to output ServoAlarm Reset. Use to set level because every company may have different input level of Servo-Drive.

C Example

```
// Set Servo-Alarm Reset level on axis 0.
AxmSignalSetServoAlarmResetLevel(0, HIGH);

// Verify Servo-Alarm Reset level value on axis 0
DWORD upLevel;

AxmSignalGetServoAlarmResetLevel (0, & upLevel);
```

VB Example

```
'Set Servo-Alarm Reset level on axis 0.
AxmSignalSetServoAlarmResetLevel 0, HIGH
```

```
'Verify Servo-Alarm Reset level value on axis 0
Dim upLevel As Long

AxmSignalGetServoAlarmResetLevel 0, upLevel
```

Delphi Example

```
{ Verify Servo-Alarm Reset level value on axis 0 }
var
upLevel: DWord;

Begin
{ Set Servo-Alarm Reset level on axis 0. }
AxmSignalSetServoAlarmResetLevel(0, HIGH);
AxmSignalGetServoAlarmResetLevel 0, @ upLevel);
end;
```

See Also

[AxmSignalSetZphaseLevel](#), [AxmSignalGetZphaseLevel](#), [AxmSignalSetServoOnLevel](#),
[AxmSignalGetServoOnLevel](#), [AxmSignalGetServoAlarmResetLevel](#), [AxmSignalSetInpos](#),
[AxmSignalGetInpos](#), [AxmSignalReadInpos](#), [AxmSignalSetServoAlarm](#), [AxmSignalGetServoAlarm](#),
[AxmSignalReadServoAlarm](#), [AxmSignalSetLimit](#), [AxmSignalGetLimit](#), [AxmSignalReadLimit](#),
[AxmSignalSetSoftLimit](#), [AxmSignalGetSoftLimit](#), [AxmSignalSetStop](#), [AxmSignalGetStop](#),
[AxmSignalReadStop](#)

AxmSignalGetServoAlarmResetLevel

Purpose

Return output level setting of Servo–Alarm Reset signal on a specific axis.

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		Channel (axis) number(starting from 0)
Level	out	DWORD*		Active Level Servo–Alarm Reset signal

Level

#define	Value	Explanation
LOW	00h	B contact (Normal Close)
HIGH	01h	A contact (Normal Open)

Return AXT_RT_SUCCESS(0000) : Successful execution of API.

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

Description

Return output level setting of Servo–Alarm Reset signal set with '[AxmSignalSetServoOnLevel](#)'.

C Example

```
// Set Servo–Alarm Reset level on axis 0.
AxmSignalSetServoAlarmResetLevel(0, HIGH);

// Verify Servo–Alarm Reset level value on axis 0
DWORD state;

AxmSignalGetServoAlarmResetLevel(0, &state);
```

VB Example

```
'Set Servo–Alarm Reset level on axis 0.
AxmSignalSetServoAlarmResetLevel 0, HIGH

'Verify Servo–Alarm Reset level value on axis 0
Dim state As Long
```

```
AxmSignalGetServoAlarmResetLevel 0, state
```

Delphi Example

```
{ Verify Servo-Alarm Reset level value on axis 0 }
var
state: DWord;

Begin
{ Set Servo-Alarm Reset level on axis 0. }
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

Set whether InPos signal is used on a specific axis. Inposition signal is only effective when servo drive is used, and it is a signal that servo drive completed position decision movement. To use InPos function, servo drive sets InPos range with proper value because InPos range can be set on servo driver.

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		Channel (axis) number(starting from 0)
Use	In	DWORD	2	Whether InPos signal is used

Use

#define	Value	Explanation
LOW	00h	B CONTACT (NORMAL CLOSE)
HIGH	01h	A CONTACT (NORMAL OPEN)
UNUSED	02h	Not-used
USED	03h	Maintain current state

Return AXT_RT_SUCCESS(0000) : Successful execution of API.

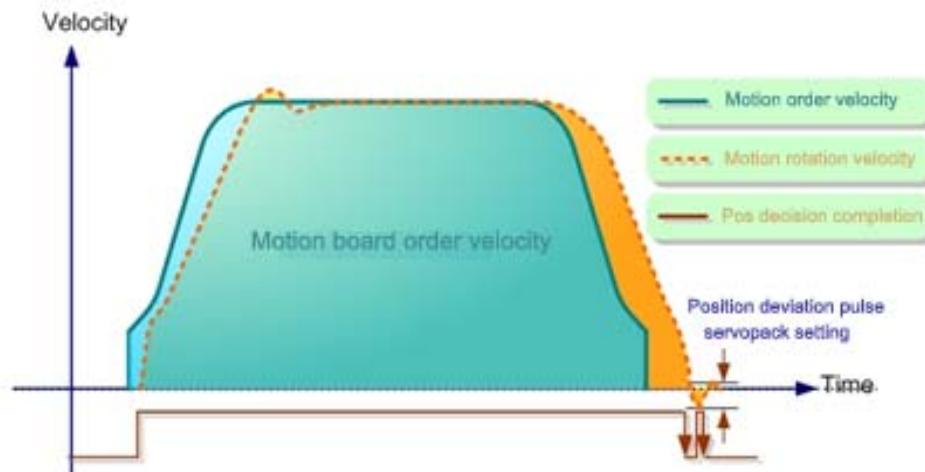
* See error code Table for more information on status error codes

Description

When InPos signal is used, set to ‘ENABLE,’ otherwise, set to ‘DISABLE.’ If sets to ‘ENABLE’ Active Level of InPos signal is applied previously set level. If set Active Level of InPos signal, set to ‘HIGH’ or ‘LOW’.

If calls this API, it recognizes that drive is ended when inPos signal (output of servo pack) comes in during motion exit. That is, move APIs operating within a loop in an API will stay in the API until inPos is to be active. InPos signal is generated when difference between command pulse input inside servo motor driver and feedback pulse input from motor is in the error range set on servo driver. Hence, this signal can be used for position decision completion signal.

Verification of InPos signal input returns [AxmSignalReadInpos](#) API.



For the reference, if step motor is used or InPos signal is not connected, it is recommended not to use by no means.

C Example

```
// Set whether InPos signal is used on axis 0
AxmSignalSetInpos (0, HIGH);

// Verify whether InPos signal is used on axis 0
DWORD      use;

AxmSignalGetInpos (0, &use);
```

VB Example

```
'Set whether InPos signal is used on axis 0
AxmSignalSetInpos 0, HIGH

'Verify whether InPos signal is used on axis 0
Dim use As Long

AxmSignalGetInpos 0, use
```

Delphi Example

```
{ Verify whether InPos signal is used on axis 0 }
var
  use: DWord;

begin
  { Set whether InPos signal is used on axis 0 }
  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

Return whether InPos signal is used on a specific axis.

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		Channel (axis) number(starting from 0)
Use	out	DWORD*	2	Whether InPos signal is used

Use

#define	Value	Explanation
LOW	00h	B CONTACT (NORMAL CLOSE)
HIGH	01h	A CONTACT (NORMAL OPEN)
UNUSED	02h	Not-used
USED	03h	Maintain current state

Return AXT_RT_SUCCESS(0000) : Successful execution of API.

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

Description

Return InPos signal state set with [AxmSignalSetInpos](#).

Return whether it is used if InPos signal assigned by user is 'ENABLE' or 'DISABLE.' If it uses InPos signal, Active Level of InPos signal can be verified

C Example

```
// Set whether InPos signal is used on axis 0
AxmSignalSetInpos (0, HIGH);

// Verify whether InPos signal is used on axis 0
DWORD      use;

AxmSignalGetInpos (0, &use);
```

VB Example

```
'Set whether InPos signal is used on axis 0
AxmSignalSetInpos 0, HIGH

'Verify whether InPos signal is used on axis 0
Dim use As Long

AxmSignalGetInpos 0, use
```

Delphi Example

```
{ Verify whether InPos signal is used on axis 0 }
var
use: DWord;

Begin
{ Set whether InPos signal is used on axis 0 }
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

Return inposition signal input state on a specific axis.

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		Channel (axis) number(starting from 0)
Status	out	DWORD*		Input state of InPos signal

Status

#define	Value	Explanation
INACTIVE	00h	No-activation
ACTIVE	01h	Activation

Return AXT_RT_SUCCESS(0000) : Successful execution of API.

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

Description

Return current InPos signal input state set with '[AxmSignalSetInpos](#)'.

C Example

```
// Verify Imposition signal input on axis 0
DWORD upStatus;
AxmSignalReadInpos (0, &upStatus);
```

VB Example

```
' Verify Imposition signal input on axis 0
Dim upStatus As Long
AxmSignalReadInpos 0, upStatus
```

Delphi Example

```
{ Verify Imposition signal input on axis 0 }
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

Set whether to use emergency when an alarm signal on a specific axis is inputted and the active level of the alarm signal.

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		Channel (axis) number(starting from 0)
Use	In	DWORD	0	Whether to use alarm signal and the active level

Use

#define	Value	Explanation
LOW	00h	B CONTACT (NORMAL CLOSE)
HIGH	01h	A CONTACT (NORMAL OPEN)
UNUSED	02h	Not-used
USED	03h	Maintain current state

Return AXT_RT_SUCCESS(0000) : Successful execution of API.

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

Description

When alarm input is on, the motion task on the corresponding axis performs emergency stop.

Low(B contact) : Close normally → Open if detected

High(A contact): Open normally → Close if detected

C Example

```
// Set whether to use emergency stop when an alarm signal on axis 0 is
// inputted
AxmSignalSetServoAlarm (0, HIGH);

// Verify settings for whether to use emergency stop when an alarm signal
// on axis 0 is inputted
DWORD      use;
```

```
AxmSignalGetServoAlarm (0, &use);
```

VB Example

```
' Set whether to use emergency stop when an alarm signal on axis 0 is
    inputted
AxmSignalSetServoAlarm 0, HIGH

' Verify settings for whether emergency stop is used when an alarm signal
    on axis 0 is inputted
Dim use As Long

AxmSignalGetServoAlarm 0, use
```

Delphi Example

```
{ Verify settings for whether to use emergency stop when an alarm signal
    on axis 0 is inputted }
var
use : DWord;

Begin
{ Set whether to use emergency stop when an alarm signal on axis 0 is
    inputted }
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

Return whether to use emergency stop and the settings for the active level of the alarm signal when an alarm signal on a specific axis is inputted.

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		Channel (axis) number(starting from 0)
Use	out	DWORD*	0	Whether to use alarm signal and the active level

Use

#define	Value	Explanation
LOW	00h	B CONTACT (NORMAL CLOSE)
HIGH	01h	A CONTACT (NORMAL OPEN)
UNUSED	02h	Not-used
USED	03h	Maintain current state

Return AXT_RT_SUCCESS(0000) : Successful execution of API.

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

Description

It returns the output level settings for the Servo-Alarm signal set by '[AxmSignalSetServoAlarm](#)'.

When alarm input is on, the motion task on the corresponding axis performs emergency stop.

Low(B contact) : Close normally → Open if detected

High(A contact): Open normally → Close if detected

C Example

```
// Set whether to use emergency stop when an alarm signal on axis 0 is
// inputted
AxmSignalSetServoAlarm 0, HIGH);

// Verify settings for whether to use emergency stop when an alarm signal
// on axis 0 is inputted
```

```
DWORD      use;  
  
AxmSignalGetServoAlarm(0, &use);
```

VB Example

```
' Set whether to use emergency stop when an alarm signal on axis 0 is  
    inputted  
AxmSignalSetServoAlarm 0, HIGH  
  
' Verify settings for whether to use emergency stop when an alarm signal  
    on axis 0 is inputted Dim use As Long  
  
AxmSignalGetServoAlarm 0, use
```

Delphi Example

```
{ Verify settings for whether to use emergency stop when an alarm signal  
    on axis 0 is inputted }  
var  
use : DWord;  
  
Begin  
{ Set whether to use emergency stop when an alarm signal on axis 0 is  
    inputted }  
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

Return input state of alarm signal on a specific axis.

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		Channel (axis) number(starting from 0)
Status	out	DWORD*		Input state of alarm signal

Status

#define	Value	Explanation
INACTIVE	00h	Non-activation
ACTIVE	01h	Activation

Return AXT_RT_SUCCESS(0000) : Successful execution of API.

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

Description

It returns the current input state of the Servo-Alarm signal set by '[AxmSignalSetServoAlarm](#)'.

When alarm input is on, the motion task on the corresponding axis performs emergency stop.
upStatus returns 1 when the input signal is on, or returns 0 when it is off.

Low(B contact) : Close normally → Open if detected

High(A contact): Open normally → Close if detected

C Example

```
// Verify alarm signal on axis 0
DWORD upStatus;
AxmSignalReadServoAlarm (0, &upStatus);
```

VB Example

```
' Verify alarm signal on axis 0
Dim upStatus As Long
AxmSignalReadServoAlarm 0, upStatus
```

Delphi Example

```
{ Verify alarm signal on axis 0 }
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

Set whether to use limit sensor on a specific axis and the signal Active Level.

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		Channel (axis) number(starting from 0)
StopMode	In	DWORD		Stop mode value after detecting limit sensor
PositiveLevel	In	DWORD	0	+ Limit Sensor Active Level
NegativeLevel	In	DWORD	0	- Limit Sensor Active Level

StopMode

#define	Value	Explanation
EMERGENCY_STOP	00h	Emergency stop
SLOWDOWN_STOP	01h	Deceleration stop

PositiveLevel

#define	Value	Explanation
LOW	00h	B CONTACT (NORMAL CLOSE)
HIGH	01h	A CONTACT (NORMAL OPEN)
UNUSED	02h	Not-used
USED	03h	Maintain current state

NegativeLevel

#define	Value	Explanation
LOW	00h	B CONTACT (NORMAL CLOSE)
HIGH	01h	A CONTACT (NORMAL OPEN)
UNUSED	02h	Not-used

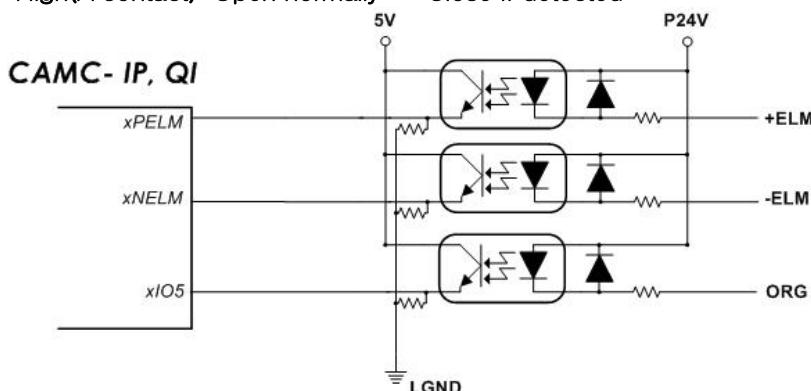
USED	03h	Maintain current state
------	-----	------------------------

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

Description

If -End Limit input is on, the (-) direction motion task on the corresponding axis stops, and if +End Limit input is on, the (+) direction motion task stops.

Low(B contact) : Close normally → Open if detected
High(A contact): Open normally → Close if detected



C Example

```
// Set so that end limit sensor on axis 0 can be used
AxmSignalSetLimit (0, EMERGENCY_STOP, HIGH, HIGH);

// Verify whether to use end limit sensor on axis 0
DWORD StopMode, PositiveLevel, NegativeLevel;
AxmSignalGetLimit (0, &StopMode, &PositiveLevel, &NegativeLevel);
```

VB Example

```
' Set so that end limit sensor on axis 0 can be used
AxmSignalSetLimit 0, EMERGENCY_STOP, HIGH, HIGH

' Verify whether to use end limit sensor on axis 0
Dim StopMode, PositiveLevel, NegativeLevel As Long
AxmSignalGetLimit 0, StopMode, PositiveLevel, NegativeLevel
```

Delphi Example

```
{ Verify whether to use end limit sensor on axis 0 }
var
  StopMode, PositiveLevel, NegativeLevel: DWord;

begin
  { Set so that end limit sensor on axis 0 can be used }
  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

Return whether to use limit sensor on a specific axis and the setting value for the signal Active Level.

Format

C

```
DWORD AxmSignalGetLimit (long IAxisNo, DWORD *upStopMode, DWORD *upPositiveLevel, DWORD
*upNegativeLevel);
```

Visual Basic

```
Function AxmSignalGetLimit (ByVal IAxisNo As Long, ByRef upStopMode As Long, ByRef upPositiveLevel As Long,
ByRef upNegativeLevel As Long) As Long
```

Delphi

```
function AxmSignalGetLimit (IAxisNo : LongInt; upStopMode : PDword; upPositiveLevel : PDword; upNegativeLevel :
PDword) : Dword; stdcall;
```

Input / Output

Name	in/out	Format	Init Value	Explanation
AxisNo	in	long		Channel (axis) number(starting from 0)
StopMode	out	DWORD*		Stop mode value after detecting limit sensor
PositiveLevel	out	DWORD*	0	+ Limit Sensor Active Level
NegativeLevel	out	DWORD*	0	- Limit Sensor Active Level

StopMode

#define	Value	Explanation
EMERGENCY_STOP	00h	Emergency stop
SLOWDOWN_STOP	01h	Deceleration stop

PositiveLevel

#define	Value	Explanation
LOW	00h	B CONTACT (NORMAL CLOSE)
HIGH	01h	A CONTACT (NORMAL OPEN)
UNUSED	02h	Not-used
USED	03h	Maintain current state

NegativeLevel

#define	Value	Explanation
LOW	00h	B CONTACT (NORMAL CLOSE)
HIGH	01h	A CONTACT (NORMAL OPEN)

UNUSED	02h	Not-used
USED	03h	Maintain current state

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

Description

Tip : It is possible to get only necessary data by inserting NULL into unnecessary variables.
 It returns the output level settings for Pos/Neg end limit signal set by '[AxmSignalSetLimit](#)' and returns StopMode. If -End Limit input is on, the (-) direction motion task on the corresponding axis will stop, and if +End Limit input is on, (+) direction motion task will stop.
 Low(B contact) : Close normally → Open if detected
 High(A contact): Open normally → Close if detected

C Example

```
// Set so that end limit sensor on axis 0 can be used
AxmSignalSetLimit (0, EMERGENCY_STOP, HIGH, HIGH);

// Verify whether to use end limit sensor on axis 0
DWORD StopMode, PositiveLevel, NegativeLevel;
AxmSignalGetLimit (0, &StopMode, &PositiveLevel, &NegativeLevel);
```

VB Example

```
' Set so that end limit sensor on axis 0 can be used
AxmSignalSetLimit 0, EMERGENCY_STOP, HIGH, HIGH

' Verify whether to use end limit sensor on axis 0
Dim StopMode, PositiveLevel, NegativeLevel As Long
AxmSignalGetLimit 0, StopMode, PositiveLevel, NegativeLevel
```

Delphi Example

```
{ Verify whether to use end limit sensor on axis 0 }
var
  StopMode, PositiveLevel, NegativeLevel: DWord;

Begin
  { Set so that end limit sensor on axis 0 can be used }
  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

Return input state of the limit sensor on a specific axis.

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		Channel (axis) number(starting from 0)
PositiveStatus	out	DWORD*		+ Limit Sensor's signal input state
NegativeStatus	out	DWORD*		- Limit Sensor's signal input state

PositiveStatus

#define	Value	Explanation
INACTIVE	00h	No-activation + Limit sensor's signal is Off
ACTIVE	01h	Activation + Limit sensor's signal is On

NegativeStatus

#define	Value	Explanation
INACTIVE	00h	No-activation - Limit sensor's signal is Off
ACTIVE	01h	Activation - Limit sensor's signal is On

Return AXT_RT_SUCCESS(0000) : Successful execution of API.

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

Description

Tip : It is possible to get only necessary data by inserting NULL into unnecessary variables.

It returns the current input state of Pos/Neg end limit signal set by '[AxmSignalSetLimit](#)'.

If -End Limit input is on, the (-) direction motion task on the corresponding axis will stop, and if +End Limit input is on, (+) direction motion task will stop.

Low(B contact) : Close normally → Open if detected

High(A contact): Open normally → Close if detected

C Example

```
// Verify the input state of limit sensor on axis 0
DWORD upPositiveLevel, upNegativeLevel;
AxmSignalReadLimit (0, &upPositiveLevel, &upNegativeLevel);
```

VB Example

```
' Verify the input state of limit sensor on axis 0
Dim upPositiveLevel , upNegativeLevel As Long
AxmSignalReadLimit 0, upPositiveLevel, upNegativeLevel
```

Delphi Example

```
{ Verify the input state of limit sensor on axis 0 }
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

Set soft limit function on a specific axis.

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		Channel (axis) number(starting from 0)
Use	In	DWORD		Whether to use soft limit function
StopMode	In	DWORD		Motion stop mode when soft limit is On
Selection	In	DWORD		Choose current position input source
PositivePos	In	double	0	Position value of + direction soft limit
NegativePos	In	double	0	Position value of – direction soft limit

Use

#define	Value	Explanation
DISABLE	00h	Soft limit not-used
ENABLE	01h	Soft limit used

StopMode

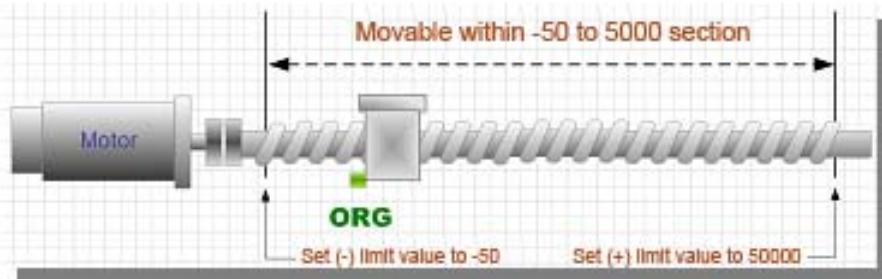
#define	Value	Explanation
EMERGENCY_STOP	00h	Emergency stop
SLOWDOWN_STOP	01h	Deceleration stop

Selection

#define	Value	Explanation
COMMAND	00h	Command position Use Cmd Pos as a current position's comparator value
ACTUAL	01h	Actual position

		Use Act Pos as a current position's comparator value
--	--	--

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



Description

Using soft limit function allows the motion movement within the range between dPositivePositiin and dNegativePos. If it gets out of range, the motion stops with the soft limit On.

Cautiion: During home search, in case it is operated by preset software limit using this API, it becomes disabled if home search stops.

N Limit : Specify stroke limit value in (-) direction.

P Limit : Specify stroke limit value in (+) direction.

C Example

```
// Set Soft Limit on axis 0
AxmSignalSetSoftLimit(0, ENABLE, EMERGENCY_STOP, COMMAND, 50000, -50);

// Verify Soft Limit settings on axis 0
DWORD upUse, upStopMode, upSelection;
double dpPositivePos, dpNegativePos;

AxmSignalGetSoftLimit (0, &upUse, &upStopMode, &upSelection,
    &dpPositivePos, &dpNegativePos);
```

VB Example

```
' Set Soft Limit on axis 0
AxmSignalSetSoftLimit 0, ENABLE, EMERGENCY_STOP, COMMAND, 50000, -50

' Verify Soft Limit settings on axis 0
Dim upUse , upStopMode , upSelection As Long
Dim dpPositivePos , dpNegativePos As Double

AxmSignalGetSoftLimit 0, upUse, upStopMode, upSelection, dpPositivePos,
    dpNegativePos
```

Delphi Example

```
{ Verify Soft Limit settings on axis 0 }
var
  upUse, upStopMode, upSelection: DWord;
  dpPositivePos, dpNegativePos: Double;
```

```
Begin
{ Set Soft Limit on axis 0 }
AxmSignalSetSoftLimit(0, ENABLE, EMERGENCY_STOP, COMMAND, 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

Return whether to use soft limit function on a specific axis, position, and stop mode, etc.

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	–	Channel(axis) number (start from 0)
Use	out	DWORD		Whether to use soft limit function
StopMode	out	DWORD		Motion stop mode when soft limit is On
Selection	out	DWORD		Choose current position input source
PositivePos	out	double	0	Position value of + direction soft limit
NegativePos	out	double	0	Position value of – direction soft limit

Use

#define	Value	Explanation
DISABLE	00h	Soft limit not-used
ENABLE	01h	Soft limit used

StopMode

#define	Value	Explanation
EMERGENCY_STOP	00h	Emergency stop
SLOWDOWN_STOP	01h	Deceleration stop

Selection

#define	Value	Explanation
COMMAND	00h	Command position Use Cmd Pos as a current position's comparator value
ACTUAL	01h	Actual position

		Use Act Pos as a current position's comparator value
--	--	--

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

Description

Tip : It is possible to get only necessary data by inserting NULL into unnecessary variables.

It returns the Pos/Neg limit value set by '[AxmSignalSetSoftLimit](#)'.

N Limit : Specify stroke limit value in (-) direction.

P Limit : Specify stroke limit value in (+) direction

C Example

```
// Set Soft Limit on axis 0
AxmSignalSetSoftLimit(0, ENABLE, EMERGENCY_STOP, CMD, 50000, -50);

// Verify Soft Limit settings on axis 0
DWORD upUse, upStopMode, upSelection;
double dpPositivePos, dpNegativePos;

AxmSignalGetSoftLimit (0, &upUse, &upStopMode, &upSelection,
                      &dpPositivePos, &dpNegativePos);
```

VB Example

```
' Set Soft Limit on axis 0
AxmSignalSetSoftLimit0, ENABLE, EMERGENCY_STOP, CMD, 50000, -50

' Verify Soft Limit settings on axis 0
Dim upUse , upStopMode , upSelection As Long
Dim dpPositivePos , dpNegativePos As Double

AxmSignalGetSoftLimit 0, upUse, upStopMode, upSelection, dpPositivePos,
                      dpNegativePos
```

Delphi Example

```
{ Verify Soft Limit settings on axis 0 }
var
  upUse, upStopMode, upSelection: DWord;
  dpPositivePos, dpNegativePos: Double;

Begin
{ Set Soft Limit on axis 0 }
  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

Set stop mode and the level of emergency stop signal (SSTOP, ESTOP) on a specific axis.

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		Channel (axis) number(starting from 0)
StopMode	In	DWORD	0	Stop mode(QL supports emergency stop only)
Level	In	DWORD	1	Whether to use stop signal and the active level

StopMode

#define	Value	Explanation
EMERGENCY_STOP	00h	Emergency stop
SLOWDOWN_STOP	01h	Deceleration stop

Level

#define	Value	Explanation
LOW	00h	B CONTACT (NORMAL CLOSE)
HIGH	01h	A CONTACT (NORMAL OPEN)
UNUSED	02h	Not-used
USED	03h	Maintain current state

Return AXT_RT_SUCCESS(0000) : Successful execution of API.

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

Description

User sets the level and the stop mode of emergency stop signal on the axis specified by a user. It is possible that pulse output becomes unavailable if settings are incorrect. This signal level can be verified by [AxmStatusReadMechanical](#).

Low(B contact) : Close normally → Open if detected
 High(A contact): Open normally → Close if detected

C Example

```
// Set whether to use stop signal on axis 0
AxmSignalSetStop (0, EMERGENCY_STTOP, ENABLE);

// Verify settings of whether to use emergency stop signal on axis 0
DWORD upStopMode, upLevel;
AxmSignalGetStop (0, &upStopMode, &upLevel);
```

VB Example

```
' Set whether to use stop signal on axis 0
AxmSignalSetStop 0, EMERGENCY_STTOP, ENABLE

' Verify settings of whether to use emergency stop signal on axis 0
Dim upStopMode , upLevel As Long
AxmSignalGetStop 0, upStopMode, upLevel
```

Delphi Example

```
{ Verify settings of whether to use emergency stop signal on axis 0 }
var
  upStopMode, upLevel: DWord;

Begin
  { Set whether to use stop signal on axis 0 }
  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

Return stop mode and the level of emergency stop signal (SSTOP, ESTOP) on a specific axis.

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	-	Channel (axis) number(starting from 0)
StopMode	out	DWORD*	0	Stop mode(QL supports emergency stop only)
Level	out	DWORD*	0	Whether to use stop signal and the active level

StopMode

#define	Value	Explanation
EMERGENCY_STOP	00h	Emergency stop
SLOWDOWN_STOP	01h	Deceleration stop

Level

#define	Value	Explanation
LOW	00h	B CONTACT (NORMAL CLOSE)
HIGH	01h	A CONTACT (NORMAL OPEN)
UNUSED	02h	Not-used
USED	03h	Maintain current state

Return AXT_RT_SUCCESS(0000) : Successful execution of API.

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

Description

Tip : It is possible to get only necessary data by inserting NULL into unnecessary variables. It returns stop mode and level of emergency stop signal(SSTOP, ESTOP) set by '[AxmSignalSetStop](#)'.

User sets the level and the stop mode of emergency stop signal on the axis specified by a user. It is possible that pulse output becomes unavailable if settings are incorrect. This signal level can be verified

by [AxmStatusReadMechanical](#).

Low(B contact) : Close normally → Open if detected

High(A contact): Open normally → Close if detected

C Example

```
// Set whether to use stop signal on axis 0
AxmSignalSetStop (0, EMERGENCY_STTOP, ENABLE);

// Verify settings of whether to use emergency stop signal on axis 0
DWORD upStopMode, upLevel;
AxmSignalGetStop (0, &upStopMode, &upLevel);
```

VB Example

```
Set whether to use stop signal on axis 0
AxmSignalSetStop 0, EMERGENCY_STTOP, ENABLE

' Verify settings of whether to use emergency stop signal on axis 0
Dim upStopMode , upLevel As Long
AxmSignalGetStop 0, upStopMode, upLevel
```

Delphi Example

```
{ Verify settings of whether to use emergency stop signal on axis 0 }
var
  upStopMode, upLevel: DWord;

Begin
  { Set whether to use stop signal on axis 0 }
  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

Return input state of emergency stop signal on a specific axis.

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		Channel (axis) number(starting from 0)
Status	out	DWORD*		Pulse output method

Status

#define	Value	Explanation
INACTIVE	00h	No-activation (Stop signal Off)
ACTIVE	01h	Activation (Stop signal On)

Return AXT_RT_SUCCESS(0000) : Successful execution of API.

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

Description

It returns the current input state of emergency stop signal (SSTOP, ESTOP) set by '[AxmSignalSetStop](#)'.

C Example

```
// Verify the state of STOP signal on axis 0
DWORD uStatus;
AxmSignalReadStop (0, &uStatus);
```

VB Example

```
' Verify the state of STOP signal on axis 0
Dim uStatus As Long
AxmSignalReadStop 0, uStatus
```

Delphi Example

```
{ Verify the state of STOP signal on axis 0 }
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

Set output of Servo-On signal on a specific axis.

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		Channel (axis) number(starting from 0)
OnOff	in	DWORD		Set Serve-On signal output

OnOff

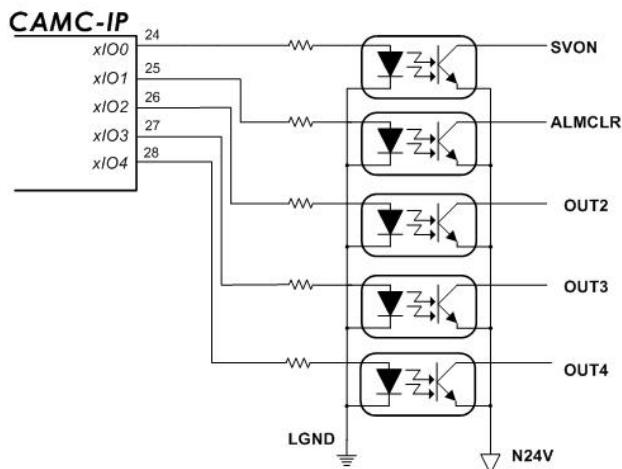
#define	Value	Explanation
FALSE	00h	Servo-On output is Off
TRUE	01h	Servo-On output is On

Return AXT_RT_SUCCESS(0000) : Successful execution of API.

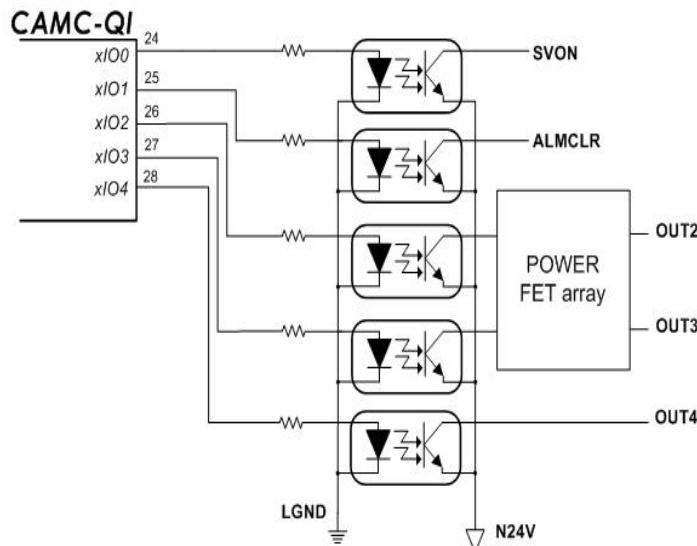
* See error code Table for more information on status error codes

Description

Servo-On signal is an external switch that is used to control ON/OFF of the servo driver when a servo driver is used. This API controls ON/OFF of the Servo-On signal. [AxmSignalServoOn](#) API returns the output state of the current Servo-On signal, and the active level can be set using [AxmSignalSetServoOnLevel](#) API.



The difference between IP and PCI-N804/404 is that PCI-N804/404 in the universal 2,3 is outputted enough for current to relay move because its output is from the photo coupler through the power FET.



C Example

```
// Servo On on axis 0
AxmSignalServoOn (0, ENABLE);

// Verify Servo On on axis 0
DWORD upOnOff;

AxmSignalIsServoOn (0, & upOnOff);
```

VB Example

```
Servo On on axis 0
AxmSignalServoOn 0, ENABLE

' Verify Servo On on axis 0
Dim upOnOff As Long

AxmSignalIsServoOn 0, upOnOff
```

Delphi Example

```
{ Verify Servo On on axis 0 }
var
upOnOff: DWord;

Begin

{ Servo On on axis 0 }
AxmSignalServoOn (0, ENABLE);
AxmSignalIsServoOn (0, @ upOnOff);
end;
```

See Also

[AxmSignalIsServoOn](#), [AxmSignalServoAlarmReset](#), [AxmSignalWriteOutput](#), [AxmSignalReadOutput](#),
[AxmSignalWriteOutputBit](#), [AxmSignalReadOutputBit](#), [AxmSignalReadInput](#), [AxmSignalReadInputBit](#)

AxmSignalIsServoOn

Purpose

Return state of Servo-On output signal on a specific axis.

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		Channel (axis) number(starting from 0)
OnOff	in	DWORD*		Set Serve-On signal output

OnOff

#define	Value	Explanation
FALSE	00h	Servo-On output is Off
TRUE	01h	Servo-On output is On

Return AXT_RT_SUCCESS(0000) : Successful execution of API.

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

Description

It returns the state of Servo-On output signal set by '[AxmSignalServoOn](#)'. Servo-On signal is an external switch that is used to control ON/OFF of the servo driver when a servo driver is used. This API controls ON/OFF of the Servo-On signal. AxmSignalIsServoOn API returns the output state of the current Servo-On signal, and the active level can be set using [AxmSignalSetServoOnLevel](#) API. The universal output 0 on the board is used as servo-on and the universal output 1 is used as servo-alarm reset.

C Example

```
// Servo On on axis 0
AxmSignalServoOn (0, ENABLE);

// Verify Servo On on axis 0
DWORD level;

AxmSignalIsServoOn (0, &level);
```

VB Example

```
Servo On on axis 0
AxmSignalServoOn 0, ENABLE

' Verify Servo On on axis 0
Dim level As Long

AxmSignalIsServoOn 0, level
```

Delphi Example

```
{ Verify Servo On on axis 0 }
var
level: DWord;

Begin
{ Servo On on axis 0 }
AxmSignalServoOn (0, ENABLE);
AxmSignalIsServoOn (0, @level);
end;
```

See Also

[AxmSignalServoOn](#), [AxmSignalServoAlarmReset](#), [AxmSignalWriteOutput](#), [AxmSignalReadOutput](#),
[AxmSignalWriteOutputBit](#), [AxmSignalReadOutputBit](#), [AxmSignalReadInput](#), [AxmSignalReadInputBit](#)

AxmSignalServoAlarmReset

Purpose

Set output of Servo–Alarm Reset signal on a specific axis.

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		Channel (axis) number(starting from 0)
OnOff	in	DWORD		Set output of Serve–Alarm Reset signal

OnOff

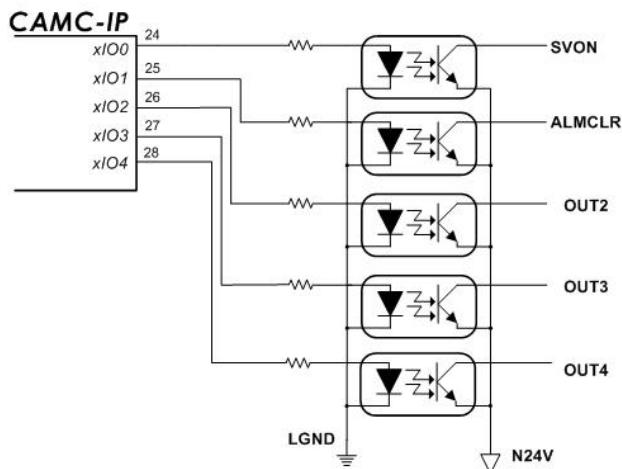
#define	Value	Explanation
FALSE	00h	Servo–Alarm Reset Output is Off
TRUE	01h	Servo–Alarm Reset Output is On

Return AXT_RT_SUCCESS(0000) : Successful execution of API.

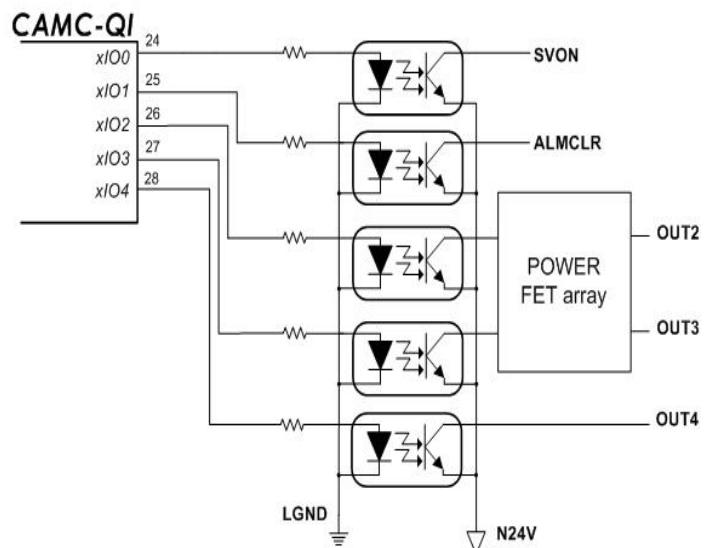
* See error code Table for more information on status error codes

Description

It is used to clear alarm in case alarm was generated in Servo Drive due to overload on a specific axis or any other reasons. The universal output 0 on the board is used as servo-on and the universal output 1 is used as servo-alarm reset.



The difference between IP and PCI-N804/404 is that PCI-n804/404 in the universal 2, 3 is outputted enough for current to relay move because its output is from the photo coupler through the power FET.



C Example

```
// Servo Alarm Reset on axis 0
AxmSignalServoAlarmReset (0, ENABLE);
```

VB Example

```
' Servo Alarm Reset on axis 0
AxmSignalServoAlarmReset 0, ENABLE
```

Delphi Example

```
begin
{ Servo Alarm Reset on axis 0 }
AxmSignalServoAlarmReset (0, ENABLE);
End;
```

See Also

[AxmSignalServoOn](#), [AxmSignallsServoOn](#), [AxmSignalWriteOutput](#), [AxmSignalReadOutput](#),

[AxmSignalWriteOutputBit](#), [AxmSignalReadOutputBit](#), [AxmSignalReadInput](#), [AxmSignalReadInputBit](#)

AxmSignalWriteOutput

Purpose

Set the value of universal output on a specific axis. This value is inputted in a hexadecimal value.

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

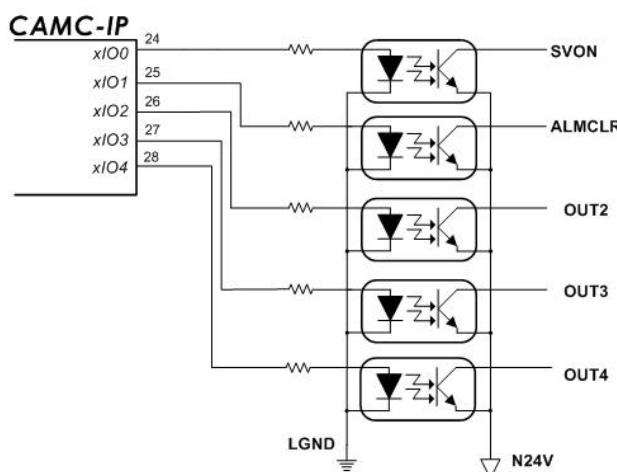
Name	in/out	Format	Init Value	Explanation
AxisNo	in	long		Channel (axis) number(starting from 0)
Value	In	DWORD		Universal output value (Hex Value)

Return AXT_RT_SUCCESS(0000) : Successful execution of API.

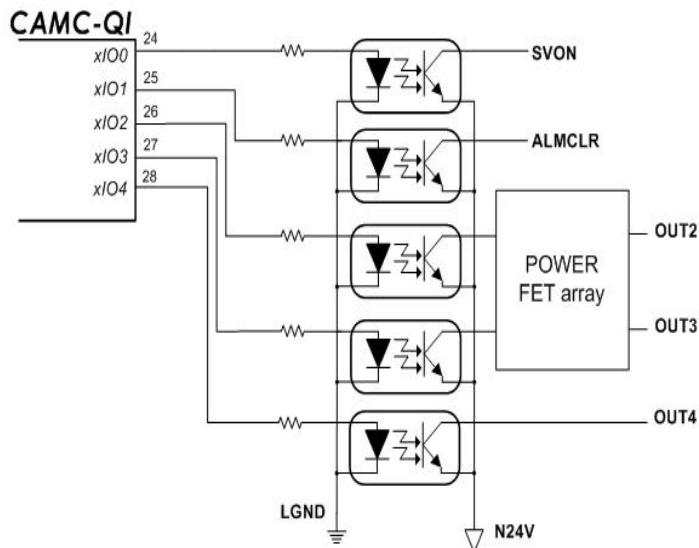
[* See error code Table for more information on status error codes](#)

Description

Universal output of the board ranges from 0 to 4. By default, universal(0) is used for Servo On, and universal(1) is used for alarm clear output signal. AxmSignalWriteOutput API is used when multiple outputs are used simultaneously, while [AxmSignalWriteOutputBit](#) API is used when only one output is necessary.



The difference between IP and PCI-N804/404 is that PCI-N804/404 in the universal 2, 3 is outputted enough for current to relay move because its output is from the photo coupler through the power FET.



C Example

```
// Set the universal output signal on axis 0
AxmSignalWriteOutput(0, 1);

// Verify the universal output signal on axis 0
DWORD uValue;
AxmSignalReadOutput (0, &uValue);
```

VB Example

```
'Set the universal output signal on axis 0
AxmSignalWriteOutput 0, 1

'Verify the universal output signal on axis 0
Dim uValue As Long
AxmSignalReadOutput 0, uValue
```

Delphi Example

```
{ Verify the universal output signal on axis 0 }
var
uValue: DWord;

Begin
{ Set the universal output signal on axis 0 }
AxmSignalWriteOutput(0, 1);

AxmSignalReadOutput (0, @uValue);
end;
```

See Also

[AxmSignalServoOn](#), [AxmSignalIsServoOn](#), [AxmSignalServoAlarmReset](#), [AxmSignalReadOutput](#),
[AxmSignalWriteOutputBit](#), [AxmSignalReadOutputBit](#), [AxmSignalReadInput](#), [AxmSignalReadInputBit](#)

AxmSignalReadOutput

Purpose

Return the value of universal output in a hexadecimal value.

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		Channel (axis) number(starting from 0)
Value	out	DWORD*		Universal output value

Return AXT_RT_SUCCESS(0000) : Successful execution of API.

[* See error code Table for more information on status error codes](#)

Description

It returns the universal output value set by '[AxmSignalWriteOutput](#)'

C Example

```
// Set the universal output signal on axis 0
AxmSignalWriteOutput(0, 1);

// Verify the universal output signal on axis 0
DWORD uValue;
AxmSignalReadOutput (0, &uValue);
```

VB Example

```
'Set the universal output signal on axis 0
AxmSignalWriteOutput 0, 1

'Verify the universal output signal on axis 0
Dim uValue As Long
AxmSignalReadOutput 0, uValue
```

Delphi Example

```
{ Verify the universal output signal on axis 0 }
var
uValue: DWord;

Begin
```

```
{ Set the universal output signal on axis 0 }
AxmSignalWriteOutput (0, 1);

AxmSignalReadOutput (0, @uValue);
end;
```

See Also

[AxmSignalServoOn](#), [AxmSignalIsServoOn](#), [AxmSignalServoAlarmReset](#), [AxmSignalWriteOutput](#),
[AxmSignalWriteOutputBit](#), [AxmSignalReadOutputBit](#), [AxmSignalReadInput](#), [AxmSignalReadInputBit](#)

AxmSignalWriteOutputBit

Purpose

Set On/Off state of a specific bit in universal output on a specific axis.

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		Channel (axis) number(starting from 0)
BitNo	in	long		Universal output bit number
OnOff	In	DWORD		Output On/Off

BitNo

#define	Value	Explanation
UIO_OUT0	00h	Servo On output (SVON)
UIO_OUT1	01h	Alarm clear output (ALARMCLEAR)
UIO_OUT2	02h	Universal output 2
UIO_OUT3	03h	Universal output 3
UIO_OUT4	04h	Universal output 4

OnOff

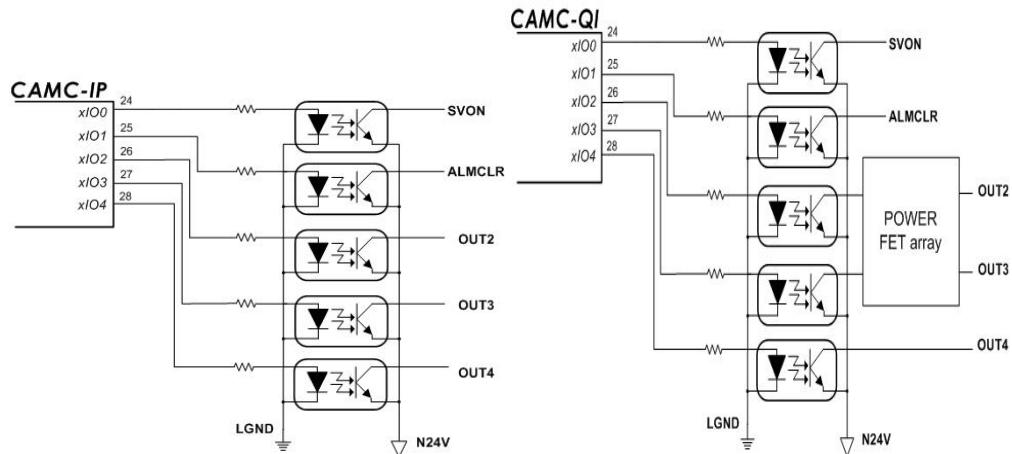
#define	Value	Explanation
FALSE	00h	Output is Off (0)
TRUE	01h	Output is On (1)

Return AXT_RT_SUCCESS(0000) : Successful execution of API.

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

Description

Universal output of the board ranges from 0 to 4. By default, universal(0) is used for Servo On, and universal(1) is used for alarm clear output signal. [AxmSignalWriteOutput](#) API is used when multiple outputs are used simultaneously, while [AxmSignalWriteOutputBit](#) API is used when only one output is necessary.

**C Example**

```
// Universal output signal 4 on axis 0 is ON
AxmSignalWriteOutputBit(0, 4, 1);

// Verify universal output signal 4 on axis 0
DWORD uOn;
AxmSignalReadOutputBit (0, 4, &uOn);
```

VB Example

```
' Universal output signal 4 on axis 0 is ON
AxmSignalWriteOutputBit 0, 4, 1

' Verify universal output signal 4 on axis 0
Dim uOn As Long
AxmSignalReadOutputBit 0, 4, uOn
```

Delphi Example

```
{ Verify universal output signal 4 on axis 0 }
var
uOn: DWord;

Begin
{ Universal output signal 4 on axis 0 is ON }
AxmSignalWriteOutputBit(0, 4, 1);

AxmSignalReadOutputBit (0, 4, @uOn);
end;
```

See Also

[AxmSignalServoOn](#), [AxmSignalIsServoOn](#), [AxmSignalServoAlarmReset](#), [AxmSignalWriteOutput](#),
[AxmSignalReadOutput](#), [AxmSignalReadOutputBit](#), [AxmSignalReadInput](#), [AxmSignalReadInputBit](#)

AxmSignalReadOutputBit

Purpose

Return On/Off state of a specific bit in universal output on a specific axis.

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	-	Channel (axis) number(starting from 0)
BitNo	in	long		Universal output bit number
OnOff	out	DWORD*		Output On/Off

BitNo

#define	Value	Explanation
UIO_OUT0	00h	Servo On output (SVON)
UIO_OUT1	01h	Alarm clear output (ALARMCLEAR)
UIO_OUT2	02h	Universal output 2
UIO_OUT3	03h	Universal output 3
UIO_OUT4	04h	Universal output 4

OnOff

#define	Value	Explanation
FALSE	00h	Output is Off (0)
TRUE	01h	Output is On (1)

Return AXT_RT_SUCCESS(0000) : Successful execution of API.

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

Description

It returns the universal output value set by '[AxmSignalWriteOutputBit](#)'.

C Example

```
// Universal output signal 4 on axis 0 is ON
AxmSignalWriteOutputBit (0, 4, 1);

// Verify universal output signal 4 on axis 0
```

```
DWORD uOn;
AxmSignalReadOutputBit(0, 4, &uOn);
```

VB Example

```
' Universal output signal 4 on axis 0 is ON
AxmSignalWriteOutputBit 0, 4, 1

' Verify universal output signal 4 on axis 0
Dim uOn As Long
AxmSignalReadOutputBit 0, 4, uOn
```

Delphi Example

```
{ Verify universal output signal 4 on axis 0 }
var
uOn: DWord;

Begin
{ Universal output signal 4 on axis 0 is ON }
AxmSignalWriteOutputBit (0, 4, 1);

AxmSignalReadOutputBit (0, 4, @uOn);
end;
```

See Also

[AxmSignalServoOn](#), [AxmSignalIsServoOn](#), [AxmSignalServoAlarmReset](#), [AxmSignalWriteOutput](#),
[AxmSignalReadOutput](#), [AxmSignalWriteOutputBit](#), [AxmSignalReadInput](#), [AxmSignalReadInputBit](#)

AxmSignalReadInput

Purpose

Return the value of universal input in a hexadecimal value.

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

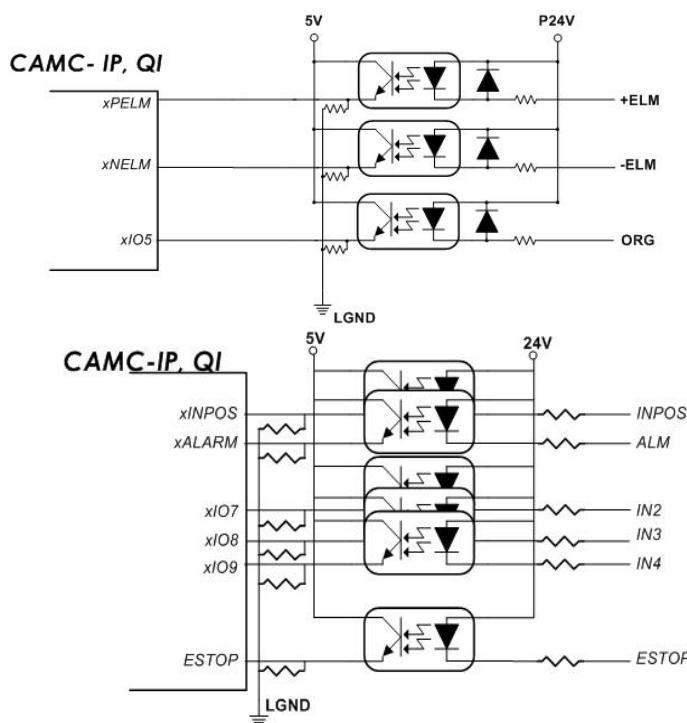
Name	in/out	Format	Init Value	Explanation
AxisNo	in	long		Channel (axis) number(starting from 0)
Value	Out	DWORD		Universal input value

Return AXT_RT_SUCCESS(0000) : Successful execution of API.

[* See error code Table for more information on status error codes](#)

Description

It returns universal input 0,1,2,3 in a hexadecimal value.



C Example

```
// Verify universal input signal on axis 0
DWORD uValue;
AxmSignalReadInput (0, &uValue);
```

VB Example

```
' Verify universal input signal on axis 0
Dim uValue As Long
AxmSignalReadInput 0, uValue
```

Delphi Example

```
{ Verify universal input signal on axis 0 }
var
  uValue: DWord;

begin
  AxmSignalReadInput (0, @uValue);
end;
```

See Also

[AxmSignalServoOn](#), [AxmSignalIsServoOn](#), [AxmSignalServoAlarmReset](#), [AxmSignalWriteOutput](#),
[AxmSignalReadOutput](#), [AxmSignalWriteOutputBit](#), [AxmSignalReadOutputBit](#), [AxmSignalReadInputBit](#)

AxmSignalReadInputBit

Purpose

Return On/Off state of a corresponding bit in universal input on a specific axis.

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		Channel (axis) number(starting from 0)
BitNo	in	long		Universal output bit number
On	in	DWORD*		On/Off state of the input bit

BitNo

#define	Value	Explanation
UIO_INP0	00h	Home sensor input
UIO_INP1	01h	Z phase input
UIO_INP2	02h	Universal input 2
UIO_INP3	03h	Universal input 3
UIO_INP4	04h	Universal input 4

On

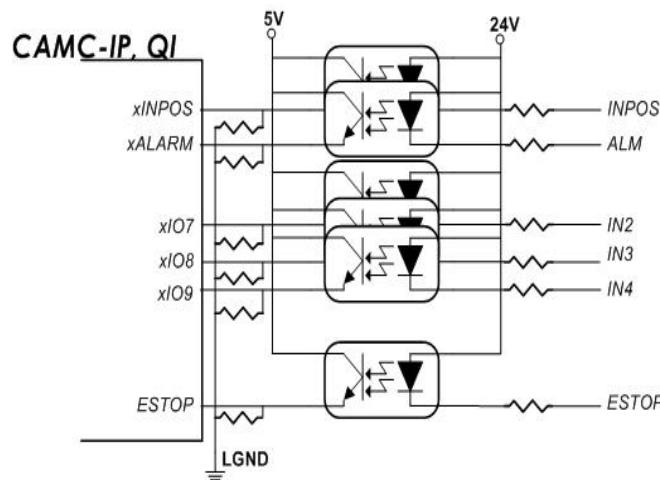
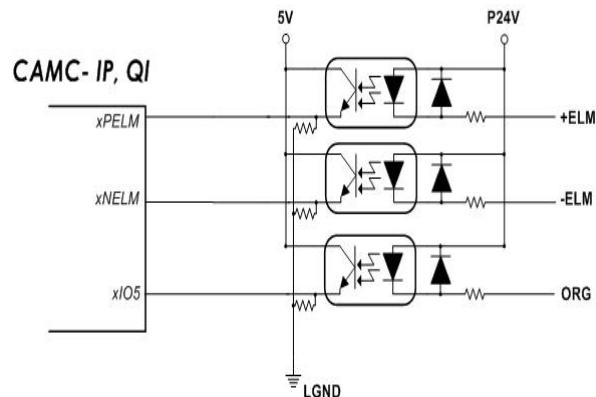
#define	Value	Explanation
FALSE	00h	Output is Off (0)
TRUE	01h	Output is On (0)

Return AXT_RT_SUCCESS(0000) : Successful execution of API.

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

Description

It returns the corresponding bit in universal input on a specific axis in a bit value.



C Example

```
// Verify the value of universal input 4 on axis 0
DWORD uOn;
AxmSignalReadInputBit(0, 4, &uOn);
```

VB Example

```
' Verify the value of universal input 4 on axis 0
Dim uOn As Long
AxmSignalReadInputBit 0, 4, uOn
```

Delphi Example

```
{ Verify the value of universal input 4 on axis 0 }
var
uOn: DWord;

begin
AxmSignalReadInputBit (0, 4, @uOn);
end;
```

See Also

[AxmSignalServoOn](#), [AxmSignalIsServoOn](#), [AxmSignalServoAlarmReset](#), [AxmSignalWriteOutput](#),
[AxmSignalReadOutput](#), [AxmSignalWriteOutputBit](#), [AxmSignalReadOutputBit](#)

Status Checking APIs After Motion Move

This chapter explains the APIs that are related to the status of motion control. Status APIs are groups of APIs required to watch motion status. Motion statuses contain checking the motion's velocity, acceleration, deceleration, position, and whether it is in-motion. Using the status APIs also allows checking the reason of stop when it is stopped, and the difference between Command values and Actual values.

Function	Description
AxmStatusReadInMotion	Return pulse out state on a specific axis.
AxmStatusReadDrivePulseCount	Return moving pulse counter value on a specific axis.
AxmStatusReadMotion	Return DriveStatus register on a specific axis.
AxmStatusReadStop	Return EndStatus register on a specific axis.
AxmStatusReadMechanical	Return Mechanical Signal Data on a specific axis.
AxmStatusReadVel	Read current moving velocity on a specific axis.
AxmStatusReadPosError	Return the difference between Command Position and Actual Position.
AxmStatusReadDriveDistance	Check the moving distance to the final drive.
AxmStatusSetActPos	Set Act position on a specific axis.
AxmStatusGetActPos	Return Act position on a specific axis.
AxmStatusSetCmdPos	Set Cmd position on a specific axis.
AxmStatusGetCmdPos	Return Cmd position on a specific axis.

AxmStatusReadInMotion

Purpose

Return whether the motion is completed on a specific axis.

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		Channel (axis) number(starting from 0)
Status	Out	DWORD*		Motion move stat

Status

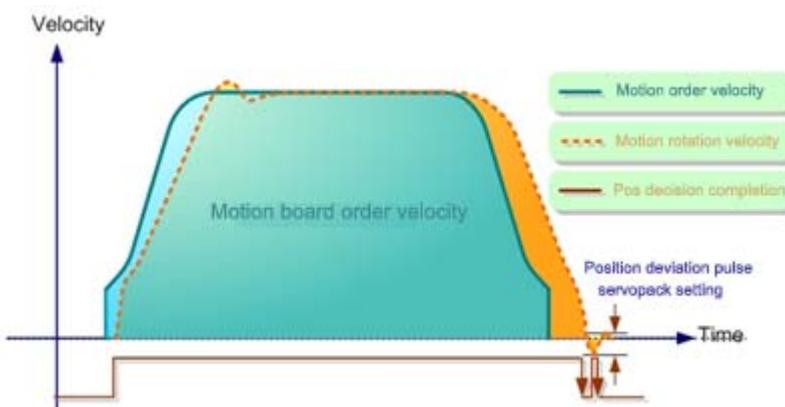
#define	Value	Explanation
FALSE	00h	Not in-motion
TRUE	01h	In-motion

Return AXT_RT_SUCCESS(0000) : Successful execution of API.

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

Description

*upStatus = 1 is outputted if the corresponding axis is in motion. Otherwise, *upStatus = 0 is outputted.



- If Inpos input signal is set to Enable when [AxmSignalSetInpos](#) API is being used, command pulse out is not returned because it is considered that motion is not completed until INP input becomes ON although Command pulse out is completed. Please refer to [AxmSignalSetInpos](#) API regarding this matter. **

C Example

```
/// Verify the moving state on axis 0
DWORD Status;
AxmStatusReadInMotion(0, &Status);
if (Status == FALSE)
    AfxMessageBox("Motor has stopped.");
```

VB Example

```
Verify the moving state on axis 0
Dim Status As Long
AxmStatusReadInMotion 0, Status
If Status = FALSE Then
    MsgBox "Motor has stopped.", vbOKCancel
End If
```

Delphi Example

```
{ Verify the moving state on axis 0 }
var
Status: DWord;

begin
AxmStatusReadInMotion(0, @Status);
if (Status = 0) then
Application.MessageBox('Motor has stopped.', 'AJINEXTEK', MB_OK);
end;
```

See Also

[AxmStatusReadDrivePulseCount](#), [AxmStatusReadMotion](#), [AxmStatusReadStop](#),
[AxmStatusReadMechanical](#), [AxmStatusReadVel](#), [AxmStatusReadPosError](#),
[AxmStatusReadDriveDistance](#), [AxmStatusSetActPos](#), [AxmStatusGetActPos](#),
[AxmStatusSetCmdPos](#), [AxmStatusGetCmdPos](#).

AxmStatusReadDrivePulseCount

Purpose

Return the value of the pulse counter from the start point to the end point for the current motion on a specific axis.

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		Channel (axis) number(starting from 0)
Pulse	Out	long*		Moving pulse counter value

Return AXT_RT_SUCCESS(0000) : Successful execution of API.

[* See error code Table for more information on status error codes](#)

Description

It returns the pulse counter value from the start point to the end point for the current motion on a specific axis. After motion move is completed, it maintains the value of previous out pulse counter first. Then it updates the value of pulse counter when a new motion move starts.

Note: In case of SMC-2V03 module, it has counter value after move exit; in case of N404 and N804, counter value is only shown during move and it is cleared after move exit.

C Example

```
// Get the pulse counter on axis 0
long      Pulse;
AxmStatusReadDrivePulseCount (0, &Pulse);
```

VB Example

```
' Get the pulse counter on axis 0
Dim Pulse As Long
AxmStatusReadDrivePulseCount 0, Pulse
```

Delphi Example

```
{ Get the pulse counter on axis 0 }
var
Pulse: LongInt;

begin
AxmStatusReadDrivePulseCount (0, @Pulse);
end;
```

See Also

[AxmStatusReadInMotion](#), [AxmStatusReadMotion](#), [AxmStatusReadStop](#), [AxmStatusReadMechanical](#),
[AxmStatusReadVel](#), [AxmStatusReadPosError](#), [AxmStatusReadDriveDistance](#), [AxmStatusSetActPos](#),
[AxmStatusGetActPos](#), [AxmStatusSetCmdPos](#), [AxmStatusGetCmdPos](#)

AxmStatusReadMotion

Purpose

Return the value of motion moving state on a specific axis.

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		Channel (axis) number(starting from 0)
Status	out	DWORD*		Motion move state on a specific axis

Return AXT_RT_SUCCESS(0000) : Successful execution of API.

[* See error code Table for more information on status error codes](#)

Description.

Return motion move status on specific axis. The details of move status value are shown in the table below.

Status

Moving state of axis (CAMC-IP)

#define	Value	Explanation
IPDRIVE_STATUS_BUSY	00000001h	Bit 0, BUSY (In DRIVE)
IPDRIVE_STATUS_DOWN	00000002h	Bit 1, DOWN (In Decelerating DRIVE)
IPDRIVE_STATUS_CONST	00000004h	Bit 2, CONST (In Constant DRIVE)
IPDRIVE_STATUS_UP	00000008h	Bit 3, Up (in Accelerating DRIVE)
IPDRIVE_STATUS_ICL	00000010h	Bit 4, ICL (ICM < INCNT)
IPDRIVE_STATUS_ICG	00000020h	Bit 5, ICG (ICM < INCNT)
IPDRIVE_STATUS_ECL	00000040h	Bit 6, ECL (ECM > EXCNT)
IPDRIVE_STATUS_ECG	00000080h	Bit 7, ECG (ECM < EXCNT)
IPDRIVE_STATUS_DRIVE_DIRECTION	00000100h	Bit 8, Drive Direction Signal (0=CW/1=CCW)
IPDRIVE_STATUS_CMD_BUSY	00000200h	In Command Execution

IPDRIVE_STATUS_PRESET_DRIVING	00000400h	In Preset Pulse Drive
IPDRIVE_STATUS_CONTINUOUS_DRIVING	00000800h	In Continuous Drive
IPDRIVE_STATUS_SIGNAL_SEARCH_DRIVING	00001000h	In Signal Search Drive
IPDRIVE_STATUS_ORG_SEARCH_DRIVING	00002000h	In Home Search Drive
IPDRIVE_STATUS MPG_DRIVING	00004000h	In MPG Drive
IPDRIVE_STATUS_SENSOR_DRIVING	00008000h	In Sensor Position Drive
IPDRIVE_STATUS_L_C_INTERPOLATION	00010000h	In Linear/Circular Interpolation Drive
IPDRIVE_STATUS_PATTERN_INTERPOLATION	00020000h	In Pattern Interpolation Drive
IPDRIVE_STATUS_INTERRUPT_BANK1	00040000h	Interrupt Generated on BANK1
IPDRIVE_STATUS_INTERRUPT_BANK2	00080000h	Interrupt Generated on BANK2

축의 구동 상태(PCI-N804/404)

#define	Return Value	Explanation
QIDRIVE_STATUS_0	00000001h	Bit 0, BUSY (In DRIVE)
QIDRIVE_STATUS_1	00000002h	Bit 1, DOWN (In Deceleration)
QIDRIVE_STATUS_2	00000004h	Bit 2, CONST (In Constant Velocity)
QIDRIVE_STATUS_3	00000008h	Bit 3, UP (In Acceleration)
QIDRIVE_STATUS_4	00000010h	Bit 4, In Continuous Drive is in move
QIDRIVE_STATUS_5	00000020h	Bit 5, In Preset Distance Drive is in move
QIDRIVE_STATUS_6	00000040h	Bit 6, In MPG Drive is in move
QIDRIVE_STATUS_7	00000080h	Bit 7, In Home Search Drive is in move
QIDRIVE_STATUS_8	00000100h	Bit 8, In Signal Search Drive is in move
QIDRIVE_STATUS_9	00000200h	Bit 9, In Interpolation Drive is in move
QIDRIVE_STATUS_10	00000400h	Bit 10, In Slave Drive is in move
QIDRIVE_STATUS_11	00000800h	Bit 11, Currently Moving Drive Direction (Display information is different on interpolation drive)

QIDRIVE_STATUS_12	00001000h	Bit 12, Waiting for Servo Position Exit Signal after Pulse Out
QIDRIVE_STATUS_13	00002000h	Bit 13, In Linear Interpolation Drive is in move
QIDRIVE_STATUS_14	00004000h	Bit 14, In Circular Interpolation Drive is in move
QIDRIVE_STATUS_15	00008000h	Bit 15, In Pulse Out
QIDRIVE_STATUS_16	00010000h	Bit 16, Number of Moving-Reserved Data(Start)(0-7)
QIDRIVE_STATUS_17	00020000h	Bit 17, Number of Moving-Reserved Data(Middle)(0-7)
QIDRIVE_STATUS_18	00040000h	Bit 18, Number of Moving-Reserved Data(End)(0-7)
QIDRIVE_STATUS_19	00080000h	Bit 19, Moving-Reserved Queue is empty.
QIDRIVE_STATUS_20	00100000h	Bit 20, Moving-Reserved Queue is full.
QIDRIVE_STATUS_21	00200000h	Bit 21, Velocity mode of current moving drive(Start)
QIDRIVE_STATUS_22	00400000h	Bit 22, Velocity mode of current moving drive (End)
QIDRIVE_STATUS_23	00800000h	Bit 23, MPG Buffer #1 Full
QIDRIVE_STATUS_24	01000000h	Bit 24, MPG Buffer #2 Full
QIDRIVE_STATUS_25	02000000h	Bit 25, MPG Buffer #3 Full
QIDRIVE_STATUS_26	04000000h	Bit 26, MPG Buffer Data OverFlow

C Example

```

DWORD drivestatus;

// Read the moving state of axis 0.
AxmStatusReadMotion (0, &drivestatus);

switch(drivestatus)
{
case IPDRIVE_STATUS_BUSY:
AfxMessageBox("Axis 0 is in move.");
break;

case IPDRIVE_STATUS_DOWN:
AfxMessageBox("Axis 0 is in deceleration.");
break;

case IPDRIVE_STATUS_CONST:
AfxMessageBox("Axis 0 is in constant velocity.");
}

```

```
break;

case IPDRIVE_STATUS_UP:
AfxMessageBox("Axis 0 is in acceleration.");
break;

case IPDRIVE_STATUS_ICL:
AfxMessageBox("Axis 0 is in ICL state.");
break;

case IPDRIVE_STATUS_ICG:
AfxMessageBox("Axis 0 is in ICG state.");
break;

case IPDRIVE_STATUS_ECL:
AfxMessageBox("Axis 0 is in ECL state.");
break;

case IPDRIVE_STATUS_ECG:
AfxMessageBox("Axis 0 is in ECC state.");
break;

case IPDRIVE_STATUS_DRIVE_DIRECTION:
AfxMessageBox("Drive direction signal (0=CW/1=CCW)");
break;

case IPDRIVE_STATUS_COMMAND_BUSY:
AfxMessageBox("Decoding command");
break;

case IPDRIVE_STATUS_PRESET_DRIVING:
AfxMessageBox("Preset pulse drive is in move");
break;

case IPDRIVE_STATUS_CONTINUOUS_DRIVING:
AfxMessageBox("Continuous drive is in move");
break;

case IPDRIVE_STATUS_SIGNAL_SEARCH_DRIVING:
AfxMessageBox("Signal search drive is in move");
break;

case IPDRIVE_STATUS_ORG_SEARCH_DRIVING:
AfxMessageBox("Home search drive is in move");
break;

case IPDRIVE_STATUS MPG_DRIVING:
AfxMessageBox("MPG drive is in move");
break;

case IPDRIVE_STATUS_SENSOR_DRIVING:
AfxMessageBox("Sensor position drive is in move");
break;

case IPDRIVE_STATUS_L_C_INTERPOLATION:
AfxMessageBox("Linear/Circular interpolation drive is in move");
break;

case IPDRIVE_STATUS_INTERRUPT_BANK1:
AfxMessageBox("Interrupt generated on BANK1");
break;

case IPDRIVE_STATUS_INTERRUPT_BANK2:
AfxMessageBox("Interrupt generated on BANK2");
break;

default:
```

```
AfxMessageBox( "Unknown Data." );
break;
}
```

VB Example

```
Dim drivestatus As Long

' Read the moving state of axis 0.
AxmStatusReadMotion 0, drivestatus

Select Case drivestatus
Case IPDRIVE_STATUS_BUSY
MsgBox "Axis 0 is in move."

Case IPDRIVE_STATUS_DOWN
MsgBox "Axis 0 is in deceleration."

Case IPDRIVE_STATUS_CONST
MsgBox "Axis 0 is in constant velocity."

Case IPDRIVE_STATUS_UP
MsgBox "Axis 0 is in acceleration."

Case IPDRIVE_STATUS_ICL
MsgBox "Axis 0 is in ICL state."

Case IPDRIVE_STATUS_ICG
MsgBox "Axis 0 is in ICG state."

Case IPDRIVE_STATUS_ECL
MsgBox "Axis 0 is in ECL state."

Case IPDRIVE_STATUS_ECG
MsgBox "Axis 0 is in ECC state."

Case IPDRIVE_STATUS_DRIVE_DIRECTION
MsgBox "Drive direction signal (0=CW/1=CCW)"

Case IPDRIVE_STATUS_COMMAND_BUSY
MsgBox "Decoding command"

Case IPDRIVE_STATUS_PRESET_DRIVING
MsgBox "Preset pulse drive is in move"

Case IPDRIVE_STATUS_CONTINUOUS_DRIVING
MsgBox "Continuous drive is in move"

Case IPDRIVE_STATUS_SIGNAL_SEARCH_DRIVING
MsgBox "Signal search drive is in move"

Case IPDRIVE_STATUS_ORG_SEARCH_DRIVING
MsgBox "Home search drive is in move"

Case IPDRIVE_STATUS MPG_DRIVING
MsgBox "MPG drive is in move"

Case IPDRIVE_STATUS_SENSOR_DRIVING
MsgBox "Sensor position drive is in move"

Case IPDRIVE_STATUS_L_C_INTERPOLATION
MsgBox "Linear/Circular interpolation drive is in move"

Case IPDRIVE_STATUS_INTERRUPT_BANK1
MsgBox "Interrupt generated on BANK1"

Case IPDRIVE_STATUS_INTERRUPT_BANK2
```

```

MsgBox "Interrupt generated on BANK2"

Case Else
  MsgBox "Unknown Data."
End Select

```

Delphi Example

```

var
drivestatus: DWord;

begin
{ Read the moving state of axis 0. }
AxmStatusReadMotion (0, @drivestatus);

begin
case drivestatus of
IPDRIVE_STATUS_BUSY:
Application.MessageBox ('Axis 0 is in move.', 'Ajinextek', MB_OK);

IPDRIVE_STATUS_DOWN:
Application.MessageBox('Axis 0 is in deceleration.', 'Ajinextek', MB_OK);

IPDRIVE_STATUS_CONST:
Application.MessageBox('Axis 0 is in constant velocity.', 'Ajinextek',
MB_OK);

IPDRIVE_STATUS_UP:
Application.MessageBox ('Axis 0 is in acceleration.', 'Ajinextek', MB_OK);

IPDRIVE_STATUS_ICL:
Application.MessageBox ('Axis 0 is in ICL state.', 'Ajinextek', MB_OK);

IPDRIVE_STATUS_ICG:
Application.MessageBox ('Axis 0 is in ICG state.', 'Ajinextek', MB_OK);

IPDRIVE_STATUS_ECL:
Application.MessageBox ('Axis 0 is in ECL state.', 'Ajinextek', MB_OK);

IPDRIVE_STATUS_ECG:
Application.MessageBox ('Axis 0 is in ECC state.', 'Ajinextek', MB_OK);

IPDRIVE_STATUS_DRIVE_DIRECTION:
Application.MessageBox ('Drive direction signal (0=CW/1=CCW)',
'Ajinextek', MB_OK);

IPDRIVE_STATUS_COMMAND_BUSY:
Application.MessageBox ('Decoding command', 'Ajinextek', MB_OK);

IPDRIVE_STATUS_PRESET_DRIVING:
Application.MessageBox ('Preset pulse drive is in move', 'Ajinextek',
MB_OK);

IPDRIVE_STATUS_CONTINUOUS_DRIVING:
Application.MessageBox ('Continuous drive is in move', 'Ajinextek',
MB_OK);

IPDRIVE_STATUS_SIGNAL_SEARCH_DRIVING:
Application.MessageBox ('Signal search drive is in move', 'Ajinextek',
MB_OK);

IPDRIVE_STATUS_ORG_SEARCH_DRIVING:
Application.MessageBox ('Home search drive is in move', 'Ajinextek',
MB_OK);

IPDRIVE_STATUS MPG_DRIVING:

```

```
Application.MessageBox ('MPG drive is in move', 'Ajinextek', MB_OK);

IPDRIVE_STATUS_SENSOR_DRIVING:
Application.MessageBox ('Sensor position drive is in move', 'Ajinextek',
    MB_OK);

IPDRIVE_STATUS_L_C_INTERPOLATION:
Application.MessageBox ('Linear/Circular interpolation drive is in move',
    'Ajinextek', MB_OK);

IPDRIVE_STATUS_INTERRUPT_BANK1:
Application.MessageBox ('Interrupt generated on BANK1', 'Ajinextek',
    MB_OK);

IPDRIVE_STATUS_INTERRUPT_BANK2:
Application.MessageBox ('Interrupt generated on BANK2', 'Ajinextek',
    MB_OK);

else Application.MessageBox('Unknown Data.', 'Ajinextek', MB_OK);
end;
end;
end;
```

See Also

[AxmStatusReadInMotion](#), [AxmStatusReadDrivePulseCount](#), [AxmStatusReadStop](#),
[AxmStatusReadMechanical](#), [AxmStatusReadVel](#), [AxmStatusReadPosError](#), [AxmStatusReadDriveDistance](#),
[AxmStatusSetActPos](#), [AxmStatusGetActPos](#), [AxmStatusSetCmdPos](#), [AxmStatusGetCmdPos](#), ,

AxmStatusReadStop

Purpose

Return the state about motion exit on a specific axis.

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		Channel (axis) number(starting from 0)
Status	out	DWORD*		Motion exit state of an axis

Return AXT_RT_SUCCESS(0000) : Successful execution of API.

[* See error code Table for more information on status error codes](#)

Description

It returns the value of motion exit state on a specific axis. Using this API, the information about whether motion has normally exited or stopped due to the reason such as external sensor, stop command, or motion error, can be verified. The values of exit state are shown on the table below.

Status

Exit state of axes (SMC-2V03)

#define	Value	Explanation
	0000h	Not in move or initial state
IPEND_STATUS_SLM	0001h	Bit 0, Exit by limit deceleration stop signal input
IPEND_STATUS_ELM	0002h	Bit 1, Exit by limit emergency stop signal input
IPEND_STATUS_SSTOP_SIGNAL	0004h	Bit 2, Exit by deceleration stop signal input
IPEND_STATUS_ESTOP_SIGNAL	0008h	Bit 3, Exit by emergency stop signal input
IPEND_STATUS_SSTOP_CMD	0010h	Bit 4, Exit by deceleration stop command
IPEND_STATUS_ESTOP_CMD	0020h	Bit 5, Exit by emergency stop command

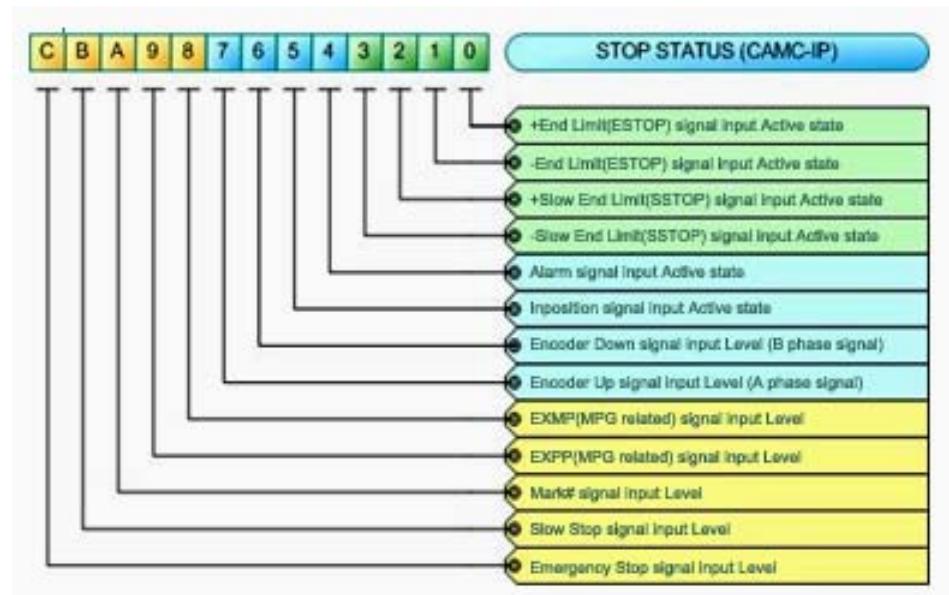
IPEND_STATUS_ALARM_SIGNAL	0040h	Bit 6, Exit by Alarm signal input
IPEND_STATUS_DATA_ERROR	0080h	Bit 7, Exit by data setting error
IPEND_STATUS_DEVIATION_ERROR	0100h	Bit 8, Exit by stall error
IPEND_STATUS_ORIGIN_DETECT	0200h	Bit 9, Exit by home search
IPEND_STATUS_SIGNAL_DETECT	0400h	Bit 10, Exit by signal search (Signal search-1/2 drive exits)
IPEND_STATUS_PRESET_PULSE_DR IVE	0800h	Bit 11, Preset pulse drive exits
IPEND_STATUS_SENSOR_PULSE_D RIVE	1000h	Bit 12, Sensor pulse drive exits
IPEND_STATUS_LIMIT	2000h	Bit 13, Exit by Limit complete stop
IPEND_STATUS_SOFTLIMIT	4000h	Bit 14, Exit by Soft limit
IPEND_STATUS_INTERPOLATION_D RIVE	8000h	Bit 15, Interpolation drive exits

Exit state of axis(PCI-N804/404)

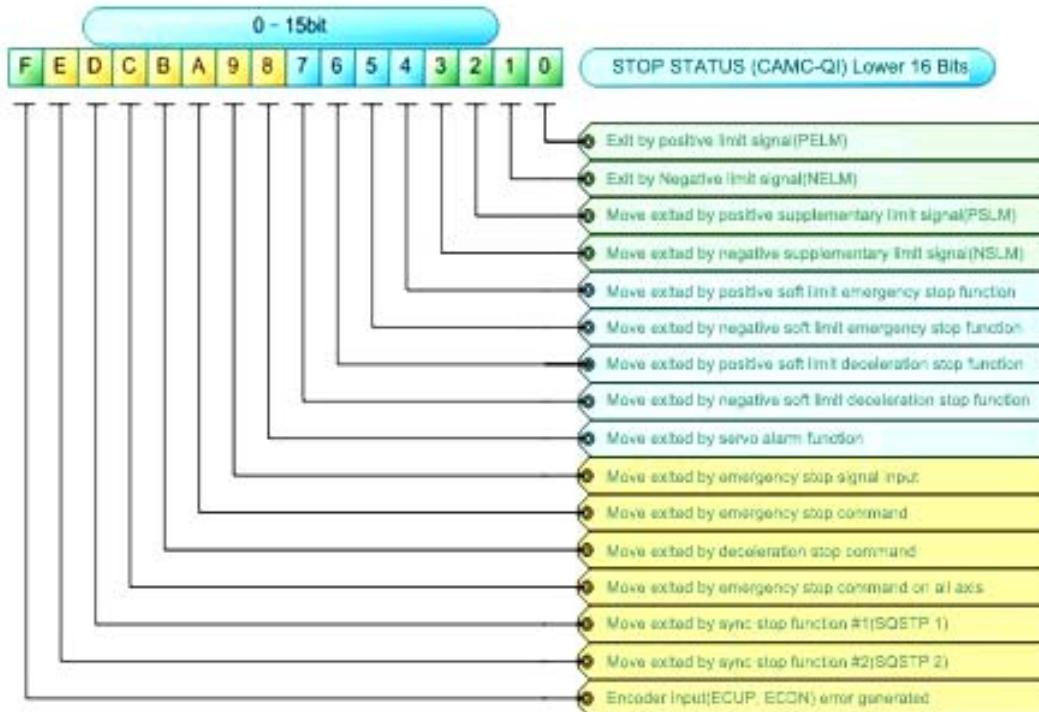
#define	Value	Explanation
	00000000h	Not in move or initial state
QIEND_STATUS_0	00000001h	Bit 0, Exit by positive limit signal(PELM)
QIEND_STATUS_1	00000002h	Bit 1, Exit by negative limit signal(NELM)
QIEND_STATUS_2	00000004h	Bit 2, Move exited by positive supplementary limit signal(PSLM)
QIEND_STATUS_3	00000008h	Bit 3, Move exited by negative supplementary limit signal(NSLM)
QIEND_STATUS_4	00000010h	Bit 4, Move exited by positive soft limit emergency stop function
QIEND_STATUS_5	00000020h	Bit 5, Move exited by negative soft limit emergency stop function
QIEND_STATUS_6	00000040h	Bit 6, Move exited by positive soft limit deceleration stop function
QIEND_STATUS_7	00000080h	Bit 7, Move exited by negative soft limit deceleration stop function
QIEND_STATUS_8	00000100h	Bit 8, Move exited by servo alarm function
QIEND_STATUS_9	00000200h	Bit 9, Move exited by emergency stop signal input

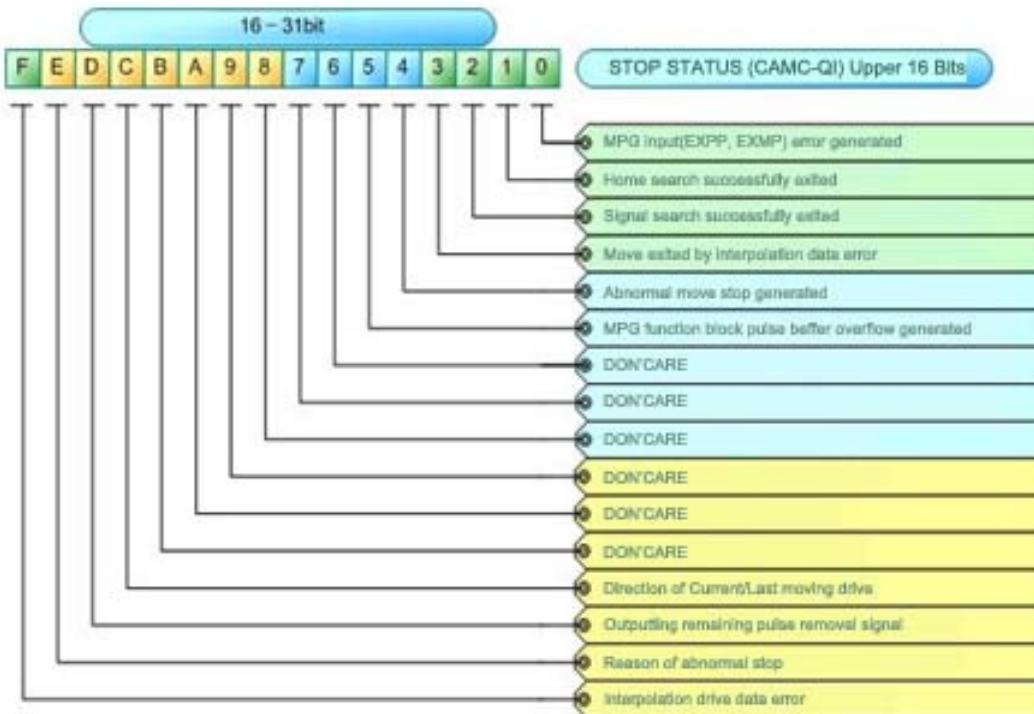
QIEND_STATUS_10	00000400h	Bit 10, Move exited by emergency stop command
QIEND_STATUS_11	00000800h	Bit 11, Move exited by deceleration stop command
QIEND_STATUS_12	00001000h	Bit 12, Move exited by emergency stop command on all axes
QIEND_STATUS_13	00002000h	Bit 13, Move exited by sync stop function #1(SQSTP1)
QIEND_STATUS_14	00004000h	Bit 14, Move exited by sync stop function #2(SQSTP2)
QIEND_STATUS_15	00008000h	Bit 15, Encoder input(ECUP,ECDN) error generated
QIEND_STATUS_16	00010000h	Bit 16, MPG input(EXPP,EXMP) error generated
QIEND_STATUS_17	00020000h	Bit 17, Home search successfully exited
QIEND_STATUS_18	00040000h	Bit 18, Signal search successfully exited
QIEND_STATUS_19	00080000h	Bit 19, Move exited by interpolation data error
QIEND_STATUS_20	00100000h	Bit 20, Abnormal move stop generated
QIEND_STATUS_21	00200000h	Bit 21, MPG function block pulse buffer overflow generated
QIEND_STATUS_22 – 27		Bit 22,– Bit27 DON'CARE
QIEND_STATUS_28	10000000h	Bit 28, Direction of Current/Last moving drive
QIEND_STATUS_29	20000000h	Bit 29, Outputting remaining pulse removal signal
QIEND_STATUS_30	40000000h	Bit 30, Cause of abnormal stop
QIEND_STATUS_31	80000000h	Bit 31, Interpolation drive data error

Definition of Each Bit in Stop Status (SMC-2V03)



Definition of Each Bit in Stop Status (PCI-N804/404)





C Example

```

WORD      endstatus;

// Read the exit state of axis 0.
AxmStatusReadStop (0, &endstatus);

switch(endstatus)
{
case 0:
AfxMessageBox( "Axis 0 has been normally exited." );
break;

case IPEND_STATUS_SLM:
AfxMessageBox( "Axis 0 has been exited by limit deceleration stop signal
           input." );
break;

case IPEND_STATUS_ELM:
AfxMessageBox( "Axis 0 has been exited by limit emergency stop signal
           input." );
break;

case IPEND_STATUS_SSTOP_SIGNAL:
AfxMessageBox( "Axis 0 has been exited by deceleration stop signal
           input." );
break;

case IPEND_STATUS_ESTOP_SIGANL:
AfxMessageBox( "Axis 0 has been exited by emergency stop signal input." );
break;

case IPEND_STATUS_SSTOP_COMMAND:

```

```
AfxMessageBox( "Axis 0 has been exited by deceleration stop command." );
break;

case IPEND_STATUS_ESTOP_COMMAND:

AfxMessageBox( "Axis 0 has been exited by emergency stop command." );
break;

case IPEND_STATUS_ALARM_SIGNAL:

AfxMessageBox( "Axis 0 has been exited by Alarm signal input." );
break;

case IPEND_STATUS_DATA_ERROR:

AfxMessageBox( "Axis 0 has been exited by data setting error." );
break;

case IPEND_STATUS_DEVIATION_ERROR:

AfxMessageBox( "Axis 0 has been exited by stall error." );
break;

case IPEND_STATUS_ORIGIN_DETECT:

AfxMessageBox( "Axis 0 has been exited by home search." );
break;

case IPEND_STATUS_SIGNAL_DETECT:

AfxMessageBox( "Axis 0 has been exited by signal search." );
break;

case IPEND_STATUS_PRESET_PULSE_DRIVE:

AfxMessageBox( "Axis 0 has been exited by Preset pulse drive." );
break;

case IPEND_STATUS_SENSOR_PULSE_DRIVE:

AfxMessageBox( "Axis 0 has been exited by Sensor pulse drive." );
break;

case IPEND_STATUS_LIMIT:

AfxMessageBox( "Axis 0 has been exited by Limit complete stop." );
break;

case IPEND_STATUS_SOFTLIMIT:

AfxMessageBox( "Axis 0 has been exited by Soft Limit." );
break;

case IPEND_STATUS_INTERPOLATION_DRIVE:

AfxMessageBox( "Axis 0 has been exited interpolation drive." );
break;

default:
AfxMessageBox( "Unknown Data." );
break;
}
```

VB Example

```

Dim endstatus As Long

' Read the exit state of axis 0.
AxmStatusReadStop 0, endstatus

Select Case endstatus
Case 0
MsgBox "Axis 0 has been normally exited."

Case IPEND_STATUS_SLM
MsgBox "Axis 0 has been exited by limit deceleration stop signal input."

Case IPEND_STATUS_ELM
MsgBox "Axis 0 has been exited by limit emergency stop signal input."

Case IPEND_STATUS_SSTOP_SIGNAL
MsgBox "Axis 0 has been exited by deceleration stop signal input."

Case IPEND_STATUS_ESTOP_SIGANL
MsgBox "Axis 0 has been exited by emergency stop signal input."

Case IPEND_STATUS_SSTOP_COMMAND
MsgBox "Axis 0 has been exited by deceleration stop command."

Case IPEND_STATUS_ESTOP_COMMAND
MsgBox "Axis 0 has been exited by emergency stop command."

Case IPEND_STATUS_ALARM_SIGNAL
MsgBox "Axis 0 has been exited by Alarm signal input."

Case IPEND_STATUS_DATA_ERROR
MsgBox "Axis 0 has been exited by data setting error."

Case IPEND_STATUS_DEVIATION_ERROR
MsgBox "Axis 0 has been exited by stall error."

Case IPEND_STATUS_ORIGIN_DETECT
MsgBox "Axis 0 has been exited by home search."

Case IPEND_STATUS_SIGNAL_DETECT
MsgBox "Axis 0 has been exited by signal search."

Case IPEND_STATUS_PRESET_PULSE_DRIVE
MsgBox "Axis 0 has been exited by Preset pulse drive."

Case IPEND_STATUS_SENSOR_PULSE_DRIVE
MsgBox "Axis 0 has been exited by Sensor pulse drive."

Case IPEND_STATUS_LIMIT
MsgBox "Axis 0 has been exited by Limit complete stop."

Case IPEND_STATUS_SOFTLIMIT
MsgBox "Axis 0 has been exited by Soft Limit."

Case IPEND_STATUS_INTERPOLATION_DRIVE
MsgBox "Axis 0 has been exited interpolation drive."

Case Else
MsgBox "Unknown Data."
End Select

```

Delphi Example

```

var
endstatus : Word;

```

```

begin
{ Read the exit state of axis 0. }
AxmStatusReadStop (0, @endstatus);

begin
case endstatus of
0:Application.MessageBox('Axis 0 has been normally exited.', 'Ajinextek',
MB_OK);

IPEND_STATUS_SLM:
Application.MessageBox('Axis 0 has been exited by limit deceleration stop
signal input.', 'Ajinextek', MB_OK);

IPEND_STATUS_ELM:
Application.MessageBox('Axis 0 has been exited by limit emergency stop
signal input.', 'Ajinextek', MB_OK);

IPEND_STATUS_SSTOP_SIGNAL:
Application.MessageBox('Axis 0 has been exited by deceleration stop signal
input.', 'Ajinextek', MB_OK);

IPEND_STATUS_ESTOP_SIGANL:
Application.MessageBox('Axis 0 has been exited by emergency stop signal
input.', 'Ajinextek', MB_OK);

IPEND_STATUS_SSTOP_COMMAND:
Application.MessageBox('Axis 0 has been exited by deceleration stop
command.', 'Ajinextek', MB_OK);

IPEND_STATUS_ESTOP_COMMAND:
Application.MessageBox('Axis 0 has been exited by emergency stop
command.', 'Ajinextek', MB_OK);

IPEND_STATUS_ALARM_SIGNAL:
Application.MessageBox('Axis 0 has been exited by Alarm signal input.',
'Ajinextek', MB_OK);

IPEND_STATUS_DATA_ERROR:
Application.MessageBox('Axis 0 has been exited by data setting error.',
'Ajinextek', MB_OK);

IPEND_STATUS_DEVIATION_ERROR:
Application.MessageBox('Axis 0 has been exited by stall error.',
'Ajinextek', MB_OK);

IPEND_STATUS_ORIGIN_DETECT:
Application.MessageBox('Axis 0 has been exited by home search.',
'Ajinextek', MB_OK);

IPEND_STATUS_SIGNAL_DETECT:
Application.MessageBox('Axis 0 has been exited by signal search.',
'Ajinextek', MB_OK);

IPEND_STATUS_PRESET_PULSE_DRIVE:
Application.MessageBox('Axis 0 has been exited by Preset pulse drive.',
'Ajinextek', MB_OK);

IPEND_STATUS_SENSOR_PULSE_DRIVE:
Application.MessageBox('Axis 0 has been exited by Sensor pulse drive.',
'Ajinextek', MB_OK);

IPEND_STATUS_LIMIT:
Application.MessageBox('Axis 0 has been exited by Limit complete stop.',
'Ajinextek', MB_OK);

IPEND_STATUS_SOFTLIMIT:

```

```
Application.MessageBox('Axis 0 has been exited by Soft Limit.',  
    'Ajinextek', MB_OK);  
  
IPEND_STATUS_INTERPOLATION_DRIVE:  
Application.MessageBox('Axis 0 has been exited interpolation drive.',  
    'Ajinextek', MB_OK);  
  
else Application.MessageBox('Unknown Data.', 'Ajinextek', MB_OK);  
end;  
end;  
end;
```

See Also

[AxmStatusReadInMotion](#), [AxmStatusReadDrivePulseCount](#), [AxmStatusReadMotion](#),
[AxmStatusReadMechanical](#), [AxmStatusReadVel](#), [AxmStatusReadPosError](#), [AxmStatusReadDriveDistance](#),
[AxmStatusSetActPos](#), [AxmStatusGetActPos](#), [AxmStatusSetCmdPos](#), [AxmStatusGetCmdPos](#)

AxmStatusReadMechanical

Purpose

Return the state of external sensors and mechanical signals on a specific axis.

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		Channel (axis) number(starting from 0)
Status	out	DWORD*		External sensor and Motion Mechanical signal status

Return AXT_RT_SUCCESS(0000) : Successful execution of API.

[* See error code Table for more information on status error codes](#)

Description

It returns the state of external sensors and Mechanical Signals on a specific axis. The Active Level of each state signal can be verified. The values of the Mechanical Signal state on the axis specified by a user are shown on the table below.

Status

Mechanical Signal Register(SMC-2V03)

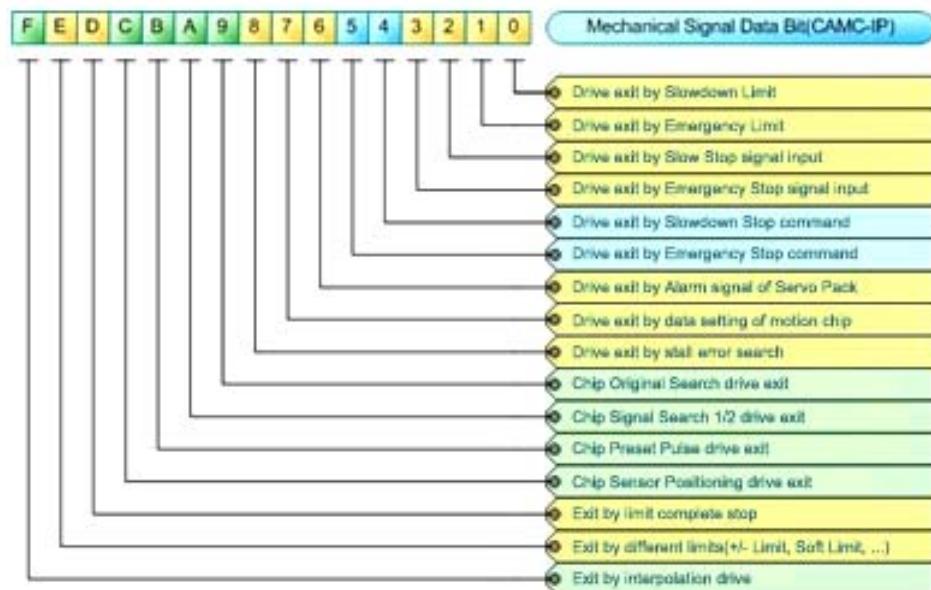
#define	Value	Explanation
IPMECHANICAL_PELM_LEVEL	0001h	Bit 0, +Limit emergency stop signal input Level
IPMECHANICAL_NELM_LEVEL	0002h	Bit 1, -Limit emergency stop signal input Level
IPMECHANICAL_PSLM_LEVEL	0004h	Bit 2, +limit deceleration stop signal input Level
IPMECHANICAL_NSLSM_LEVEL	0008h	Bit 3, -limit deceleration stop signal input Level
IPMECHANICAL_ALARM_LEVEL	0010h	Bit 4, Alarm signal input Level
IPMECHANICAL_INP_LEVEL	0020h	Bit 5, InPos signal input Level
IPMECHANICAL_ENC_DOWN_LEVEL	0040h	Bit 6, Encoder DOWN(B phase) signal input Level
IPMECHANICAL_ENC_UP_LEVEL	0080h	Bit 7, Encoder UP(A phase) signal input Level
IPMECHANICAL_EXMP_LEVEL	0100h	Bit 8, EXMP signal input Level

IPMECHANICAL_EXPP_LEVEL	0200h	Bit 9, EXPP signal input Level
IPMECHANICAL_MARK_LEVEL	0400h	Bit 10, MARK# signal input Level
IPMECHANICAL_SSTOP_LEVEL	0800h	Bit 11, SSTOP signal input Level
IPMECHANICAL_ESTOP_LEVEL	1000h	Bit 12, ESTOP signal input Level
IPMECHANICAL_SYNC_LEVEL	2000h	Bit 13, SYNC signal input Level
IPMECHANICAL_MODE8_16_LEVEL	4000h	Bit 14, MODE8_16 signal input Level

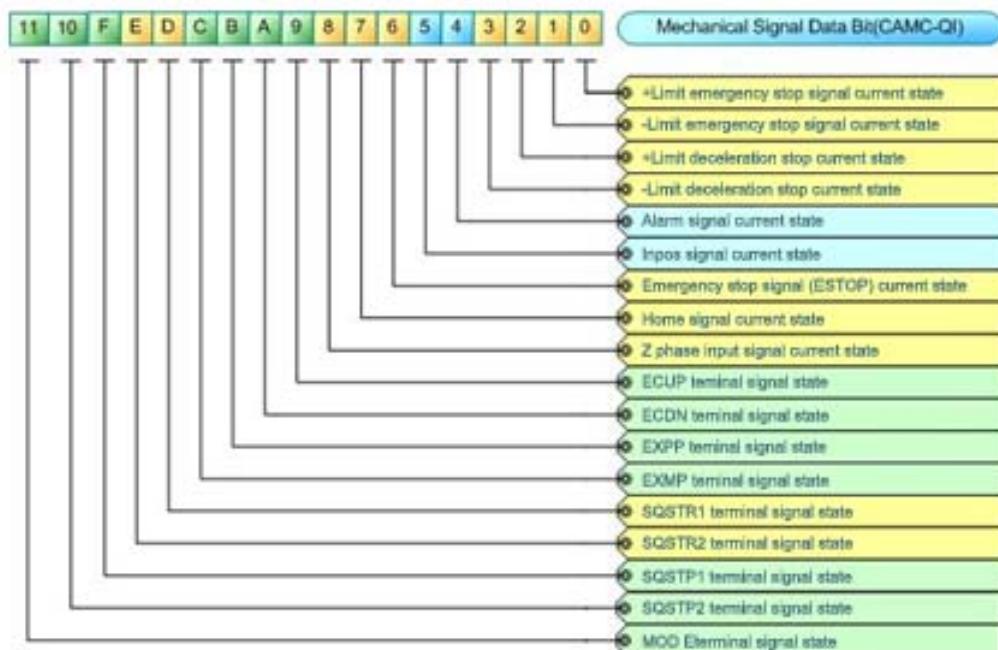
Mechanical Signal Register (PCI-N804/404)

#define	Value	Explanation
QIMECHANICAL_PELM_LEVEL	0001h	Bit 0, +Limit emergency stop signal current state
QIMECHANICAL_NELM_LEVEL	0002h	Bit 1, -Limit emergency stop signal current state
QIMECHANICAL_PSLM_LEVEL	0004h	Bit 2, +limit deceleration stop current state
QIMECHANICAL_NSLM_LEVEL	0008h	Bit 3, -limit deceleration stop current state
QIMECHANICAL_ALARM_LEVEL	0010h	Bit 4, Alarm signal current state
QIMECHANICAL_INP_LEVEL	0020h	Bit 5, InPos signal current state
QIMECHANICAL_ESTOP_LEVEL	0040h	Bit 6, Emergency stop signal (ESTOP) current state
QIMECHANICAL_ORG_LEVEL	0080h	Bit 7, Home signal current state
QIMECHANICAL_ZPHASE_LEVEL	0100h	Bit 8, Z phase input signal current state
QIMECHANICAL_ECUP_LEVEL	0200h	Bit 9, ECUP terminal signal state.
QIMECHANICAL_ECDN_LEVEL	0400h	Bit 10, ECDN terminal signal state.
QIMECHANICAL_EXPP_LEVEL	0800h	Bit 11, EXPP terminal signal state
QIMECHANICAL_EXMP_LEVEL	1000h	Bit 12, EXMP terminal signal state
QIMECHANICAL_SQSTR1_LEVEL	2000h	Bit 13, SQSTR1 terminal signal state
QIMECHANICAL_SQSTR2_LEVEL	4000h	Bit 14, SQSTR2 terminal signal state
QIMECHANICAL_SQSTP1_LEVEL	8000h	Bit 15, SQSTP1 terminal signal state
QIMECHANICAL_SQSTP2_LEVEL	10000h	Bit 16, SQSTP2 terminal signal state
QIMECHANICAL_MODE_LEVEL	20000h	Bit 17, MODE terminal signal state

MeDefinition of Each bit on chanical Signal (SMC-2V03)



Mechanical Signal 각 Bit별 의미 (PCI-N804/404)



C Example

```
// Read sensoring state of each sensor on axis 0
DWORD Status;
AxmStatusReadMechanical (0, &Status);
```

VB Example

```
' Read sensoring state of each sensor on axis 0
Dim Status As Long
AxmStatusReadMechanical 0, Status
```

Delphi Example

```
{ Read sensoring state of each sensor on axis 0 }
var
Status: DWord;

begin
AxmStatusReadMechanical (0, @Status);
end;
```

See Also

[AxmStatusReadInMotion](#), [AxmStatusReadDrivePulseCount](#), [AxmStatusReadMotion](#), [AxmStatusReadStop](#),
[AxmStatusReadVel](#), [AxmStatusReadPosError](#), [AxmStatusReadDriveDistance](#), [AxmStatusSetActPos](#),
[AxmStatusGetActPos](#), [AxmStatusSetCmdPos](#), [AxmStatusGetCmdPos](#)

AxmStatusReadVel

Purpose

Read current moving velocity on a specific axis.

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	-	Channel (axis) number(starting from 0)
Velocity	out	double*		Current velocity

Return AXT_RT_SUCCESS(0000) : Successful execution of API.

[* See error code Table for more information on status error codes](#)

Description

It reads current moving velocity on the axis specified by a user. The unit of the velocity is unit/sec.

C Example

```
double    velocity;
AxmMoveStartPos(0, 1000, 100, 200, 200);
// Return the current moving velocity on a specific axis
AxmStatusReadVel (0, &Velocity);
```

VB Example

```
Dim velocity As Double
AxmMoveStartPos 0, 1000, 100, 200, 200
' Return the current moving velocity on a specific axis
AxmStatusReadVel 0, Velocity
```

Delphi Example

```
var
Velocity : Double;
begin
AxmMoveStartPos(0, 1000, 100, 200, 200);
{ Return the current moving velocity on a specific axis }
```

```
AxmStatusReadVel (0, @Velocity);  
end;
```

See Also

[AxmStatusReadInMotion](#), [AxmStatusReadDrivePulseCount](#), [AxmStatusReadMotion](#),
[AxmStatusReadStop](#), [AxmStatusReadMechanical](#), [AxmStatusReadPosError](#), [AxmStatusReadDriveDistance](#),
[AxmStatusSetActPos](#), [AxmStatusGetActPos](#), [AxmStatusSetCmdPos](#), [AxmStatusGetCmdPos](#).

AxmStatusReadPosError

Purpose

Return the difference between Command Position and Actual Position on a specific axis.

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		Channel (axis) number(starting from 0)
Error	out	double*		Difference between Command Position and Actual Position

Return AXT_RT_SUCCESS(0000) : Successful execution of API.

[* See error code Table for more information on status error codes](#)

Description

You can check if the output pulse from the axis controller specified by a user is normally operating by returning the difference between internal Command Position and Actual Position(Encoder). Step motor can also be verified if an Encoder is attached as a supplementary. This API cannot be used without Encoder.

C Example

```
// Verify the difference between Cmd Pos and Act Pos on axis 0
double Error;
AxmStatusReadPosError (0, &Error);
```

VB Example

```
' Verify the difference between Cmd Pos and Act Pos on axis 0
Dim Error As Double
AxmStatusReadPosError 0, Error
```

Delphi Example

```
{ Verify the difference between Cmd Pos and Act Pos on axis 0 }
var
Error: Double;

begin
AxmStatusReadPosError (0, @Error);
end;
```

See Also

[AxmStatusReadInMotion](#), [AxmStatusReadDrivePulseCount](#), [AxmStatusReadMotion](#), [AxmStatusReadStop](#),
[AxmStatusReadMechanical](#), [AxmStatusReadVel](#), [AxmStatusReadDriveDistance](#), [AxmStatusSetActPos](#),
[AxmStatusGetActPos](#), [AxmStatusSetCmdPos](#), [AxmStatusGetCmdPos](#)

AxmStatusReadDriveDistance

Purpose

Return the position driving distance from the start point to the end point of the current motion on a specific axis

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		Channel (axis) number(starting from 0)
Unit	out	double*		Driving distance

Return AXT_RT_SUCCESS(0000) : Successful execution of API.

[* See error code Table for more information on status error codes](#)

Description

After motion move is completed, it maintains the previous driving distance first then it updates the value of driving distance when a new motion move starts.

Note: In case of SMC-2V03 module, it has counter value after move exit; in case of N404 and N804, counter value is only shown during move and it is cleared after move exit.

C Example

```
// Set the current position to '0' on axis 0 and verify the output pulse
double Unit;
AxmStatusReadDriveDistance (0, &Unit);
```

VB Example

```
' Set the current position to '0' on axis 0 and verify the output pulse
Dim Unit As Double
AxmStatusReadDriveDistance 0, Unit
```

Delphi Example

```
{ Set the current position to '0' on axis 0 and verify the output pulse }
var
  Unit: Double;

begin
  AxmStatusReadDriveDistance (0, @Unit);
end;
```

See Also

[AxmStatusReadInMotion](#), [AxmStatusReadDrivePulseCount](#), [AxmStatusReadMotion](#), [AxmStatusReadStop](#),
[AxmStatusReadMechanical](#), [AxmStatusReadVel](#), [AxmStatusReadPosError](#), [AxmStatusSetActPos](#),
[AxmStatusGetActPos](#), [AxmStatusSetCmdPos](#), [AxmStatusGetCmdPos](#)

AxmStatusSetActPos

Purpose

Set Actual position(Encoder value) on a specific axis.

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	-	Channel (axis) number(starting from 0)
Pos	In	double		Act position

Return AXT_RT_SUCCESS(0000) : Successful execution of API.

[* See error code Table for more information on status error codes](#)

Description

It can be forced to set Actual position on the axis specified by a user. It is mainly used to set the current position to 0.

C Example

```
// Set the current position to the encoder position '0' on axis 0
AxmStatusSetActPos (0, 0);

// Verify encoder position on axis 0
double Pos;
AxmStatusGetActPos (0, &Pos);
```

VB Example

```
' Set the current position to the encoder position '0' on axis 0
AxmStatusSetActPos 0, 0

' Verify encoder position on axis 0
Dim Pos As Double
AxmStatusGetActPos 0, Pos
```

Delphi Example

```
{ Verify encoder position on axis 0 }
var
  Pos: Double;

Begin
  { Set the current position to the encoder position '0' on axis 0 }
```

```
AxmStatusSetActPos ( 0, 0 );  
AxmStatusGetActPos ( 0, @Pos );  
end;
```

See Also

[AxmStatusReadInMotion](#), [AxmStatusReadDrivePulseCount](#), [AxmStatusReadMotion](#),
[AxmStatusReadStop](#), [AxmStatusReadMechanical](#), [AxmStatusReadVel](#), [AxmStatusReadPosError](#),
[AxmStatusReadDriveDistance](#), [AxmStatusGetActPos](#), [AxmStatusSetCmdPos](#), [AxmStatusGetCmdPos](#)

AxmStatusGetActPos

Purpose

Return Actual position(Encoder value) on a specific axis.

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		Channel (axis) number(starting from 0)
Pos	out	double*		Act position

Return AXT_RT_SUCCESS(0000) : Successful execution of API.

[* See error code Table for more information on status error codes](#)

Description

It returns the Actual position read by the encoder on the axis specified by a user.

C Example

```
// Set the current position to the encoder position '0' on axis 0
AxmStatusSetActPos (0, 0);

// Verify encoder position on axis 0
double Pos;
AxmStatusGetActPos (0, &Pos);
```

VB Example

```
' Set the current position to the encoder position '0' on axis 0
AxmStatusSetActPos 0, 0

' Verify encoder position on axis 0
Dim Pos As Double
AxmStatusGetActPos 0, Pos
```

Delphi Example

```
{ Verify encoder position on axis 0}
var
  Pos: Double;

begin
  { Set the current position to the encoder position '0' on axis 0}
  AxmStatusSetActPos (0, 0);
```

```
AxmStatusGetActPos (0, @Pos);  
end;
```

See Also

[AxmStatusReadInMotion](#), [AxmStatusReadDrivePulseCount](#), [AxmStatusReadMotion](#),
[AxmStatusReadStop](#), [AxmStatusReadMechanical](#), [AxmStatusReadVel](#), [AxmStatusReadPosError](#),
[AxmStatusReadDriveDistance](#), [AxmStatusSetActPos](#), [AxmStatusSetCmdPos](#),
[AxmStatusGetCmdPos](#),

AxmStatusSetCmdPos

Purpose

Set Command position on a specific axis.

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		Channel (axis) number(starting from 0)
Pos	In	double		Cmd position

Return AXT_RT_SUCCESS(0000) : Successful execution of API.

[* See error code Table for more information on status error codes](#)

Description

It can be forced to set Command position on the axis specified by a user. It is mainly used to set the current position to 0.

C Example

```
// Set the current position to the command position '0' on axis 0
AxmStatusSetCmdPos (0, 0);

// Verify command position on axis 0
double Pos;
AxmStatusGetCmdPos (0, &Pos);
```

VB Example

```
' Set the current position to the command position '0' on axis 0
AxmStatusSetCmdPos 0, 0

' Verify command position on axis 0
Dim Pos As Double
AxmStatusGetCmdPos 0, Pos
```

Delphi Example

```
{ Verify command position on axis 0 }
var
  Pos: Double;

begin
  { Set the current position to the command position '0' on axis 0 }
```

```
AxmStatusSetCmdPos (0, 0);  
AxmStatusGetCmdPos (0, @Pos);  
end;
```

See Also

[AxmStatusReadInMotion](#), [AxmStatusReadDrivePulseCount](#), [AxmStatusReadMotion](#),
[AxmStatusReadStop](#), [AxmStatusReadMechanical](#), [AxmStatusReadVel](#), [AxmStatusReadPosError](#),
[AxmStatusReadDriveDistance](#), [AxmStatusSetActPos](#), [AxmStatusGetActPos](#), [AxmStatusGetCmdPos](#),

AxmStatusGetCmdPos

Purpose

Return Command position on a specific axis.

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		Channel (axis) number(starting from 0)
Pos	out	double*		Cmd Position

Return AXT_RT_SUCCESS(0000) : Successful execution of API.

[* See error code Table for more information on status error codes](#)

Description

It returns the internal Command position of the controller on the axis specified by user.

C Example

```
// Set the current position to the command position '0' on axis 0
AxmStatusSetCmdPos (0, 0);

// Verify command position on axis 0
double Pos;
AxmStatusGetCmdPos (0, &Pos);
```

VB Example

```
' Set the current position to the command position '0' on axis 0
AxmStatusSetCmdPos 0, 0

' Verify command position on axis 0
Dim Pos As Double
AxmStatusGetCmdPos 0, Pos
```

Delphi Example

```
{ Verify command position on axis 0 }
var
  Pos: Double;

begin
  { Set the current position to the command position '0' on axis 0 }
  AxmStatusSetCmdPos (0, 0);
```

```
AxmStatusGetCmdPos ( 0 , @Pos ) ;  
end;
```

See Also

[AxmStatusReadInMotion](#), [AxmStatusReadDrivePulseCount](#), [AxmStatusReadMotion](#),
[AxmStatusReadStop](#), [AxmStatusReadMechanical](#), [AxmStatusReadVel](#),
[AxmStatusReadPosError](#), [AxmStatusReadDriveDistance](#), [AxmStatusSetActPos](#),
[AxmStatusGetActPos](#), [AxmStatusSetCmdPos](#).

Home Search API

Function	Description
<u>AxmHomeSetSignalLevel</u>	Sets home sensor level of a specific axis.
<u>AxmHomeGetSignalLevel</u>	Returns home sensor level of a specific axis.
<u>AxmHomeReadSignal</u>	Returns state of home signal input of a specific axis.
<u>AxmHomeSetMethod</u>	To execute home search of the corresponding axis, the parameters related to home search must be preset. If initialization has been normally executed using AxmMotLoadParaAll API setting file, no further setup is required.. Home search method setting includes searching direction, the signal to be used as home, home sensor active level, and whether encoder Z phase is detected, etc. (Refer to the description part of AxmMotLoadParaAll for details)
<u>AxmHomeGetMethod</u>	To execute home search of the corresponding axis, the parameters related to home search must be preset. Returns the home-related parameters that were previously set.
<u>AxmHomeSetVel</u>	It detects home through multi-level steps to search it quickly and accurately. At this point, it sets the velocity to be used for each step. Time and accuracy of home search is determined based on the setting of these velocities. You can set home search velocities while adjusting velocities in each step accordingly. (Refer to the description part of AxmMotLoadParaAll for details)
<u>AxmHomeGetVel</u>	It detects home through multi-level steps to search it quickly and accurately. At this point, it sets the velocity to be used for each step. Time and accuracy of home search is determined based on the setting of these velocities. It gets each velocity previously set.
<u>AxmHomeSetStart</u>	If home search starting API is executed, the thread to execute home search of the corresponding axis is automatically created within the library. Then it sequentially executes home search and is exited.
<u>AxmHomeSetResult</u>	After home search is successfully executed using home search API, the search result is set to HOME SUCCESS. This API can set the result at user's discretion without executing home search.

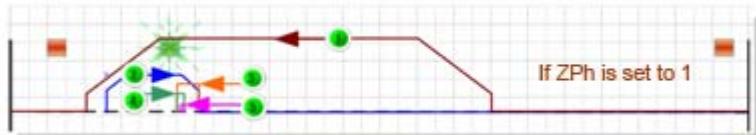
	In practice, if an exceptional situation occurs during a normal move, the user can set the home search result to HOME_ERR_USER_BREAK or so to recover executing home search again.
AxmHomeGetResult	Verifies the search result of home search API. If home search starts, it is set to HOME_SEARCHING, and if it fails, the cause of failure is set. You can remove the cause of failure and restart home search.
AxmHomeGetRate	You can check the progress rate after home search starts. After home search is completed, it returns 100 regardless of whether or not it was successful. You can verify whether or not home search was successful using GetHome Result API.

For equipment move, home search needs to be executed. The important thing in the parameters related to home search is the velocity used for home search. dVelFirst is the initial velocity and dVelSecond is the velocity at which home sensor escapes to the opposite direction after its first detection. dVelThird is the velocity used for second home search. dVelLast is the velocity used for the very last home search. If the value of dVelLast is too big, position errors may occur when home search is completed. On the other hand, if the value is too small, it takes too much time for home search. Therefore, setting a proper value is important.

Home Search Variables	Variable Description
HmSig	Sets a signal to use as home sensor (e.g.) Home Sensor(4), +Limit Sensor(0), etc.
HmLev	Sets the Active Level of Home Sensor (e.g.) 0: A contact, 1: B contact
HmDir	Sets the initial search direction in home sensor search (e.g.) 1: (+)direction, 0: (-)direction
Zphas	Sets whether to detect encoder zphase after home sensor search
VelFirst	Velocity of Initial high-speed detection in home search (when the initial home sensor is not detected)
VelSecond	Velocity to escape to the opposite direction after the first sensor detection in home search
VelThird	Velocity to execute search again after the first sensor detection in home search
VelLast	Sets the last detection velocity in home search [determine the accuracy of home search]
AccFirst	Initial high-speed detection acceleration in home search (The unit of acceleration is PPS[Pulses/sec] when Unit/pulse is set to 1/1)
AccSecond	Acceleration to escape to the opposite direction after the first sensor detection in home search (The unit of acceleration is PPS[Pulses/sec] when Unit/pulse is set to 1/1)
HClrTim	Standby time before Cmd, Act(Encoder) position are cleared after home search

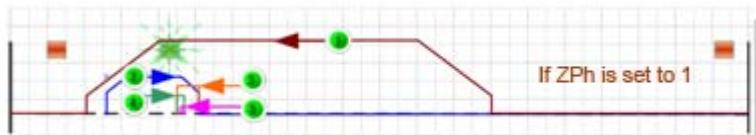
HOffset	Moving offset value when resetting mechanical home after home search is completed
NSWLimit	Sets (-) Software Limit value to apply after home search is completed
PSWLimit	Sets (-) Software Limit value to apply after home search is completed (If NSWLimit is greater than or equal to PSWLimit, S/W Limit won't be set)

When Zph is 1



By detecting Z phase Down Edge in Step 4 and detecting Z phase Up Edge again in Step 5, it enhances the velocity and accuracy of home search.

Understanding home search velocity setting per each step



1 Use VelFirst, AccFirst

High-speed detection of home sensor in the direction of HmDir and deceleration stop at HAccF deceleration

2 Use VelSecond, AccSecond

Detection of Down edge of home sensor in the opposite direction of HmDir and deceleration stop at HAccF deceleration

3 Use VelThird

Detection of Up Edge of home sensor in the direction of HmDir and emergency stop

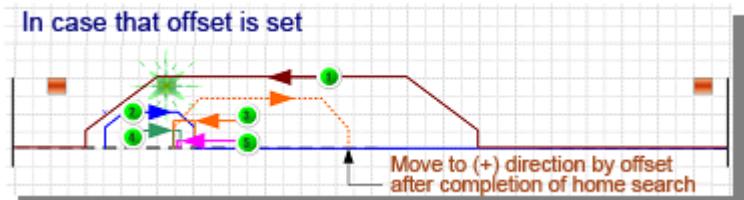
4 Use VelLast

Detection of Down Edge of Z phase in the opposite direction of HmDir and emergency stop

5 Use VelLast

Detection of Up Edge of Z phase in the direction of HmDir and emergency stop

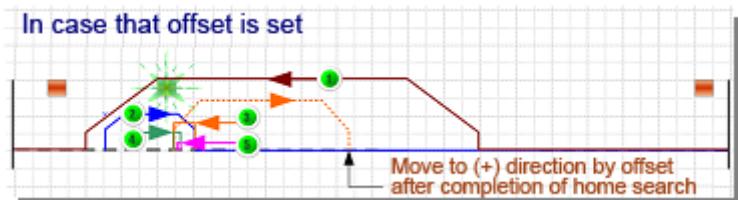
In case that home needs to be set by moving to a specific position after mechanical home sensor detection is completed, if a value is assigned to HOffset, it automatically resets home after moving by a corresponding Offset. You should be careful not to exceed the range between (+) and (-) limits. Otherwise, home search cannot be executed normally.



After Offset movement
completing with automatic it

Soft limit can be set after home search is normally completed. Soft limit can be used when limit sensor is not mechanically equipped or when limit needs to be modified depending on situations. Remember that the value of PLimit(+) must be greater than that of NLimit(–) in order to set soft limit normally. If NSwLimit is greater than or equal to PSwLimit, Soft Limit will not be set.

- Note:
1. During home search, software limit is enabled again when software limit is set in advance even if it is stopped by the user or outside.
 2. During home search, software limit is disabled at the start when software limit is set in advance; when home search is done, enable software limit again.



AxmHomeSetSignalLevel

Purpose

Set output level of home input signal of a specific axis.

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	-	Channel (axis) number(starting from 0)
Level	In	DWORD	1	Active Level of Home signal

Level

#define	Value	Explanation
LOW	00h	B contact (Normal Close)
HIGH	01h	A contact (Normal Open)

Return AXT_RT_SUCCESS(0000) : Successful execution of API.

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

Description

Caution : Even if the level is incorrectly set in the – direction, it can still move in the + direction; It may cause a problem during home search.

It sets home input level of the axis specified by user. The chips in SMC-2V03, PCI-N804/404 use universal input 0 as home signal of encoder. In order to read home level, you can use AxmHomeReadSignal API.

C Example

```
/ Set home input signal level on axis 0
AxmHomeSetSignalLevel(0, HIGH);

// Verify home input signal level on axis 0
DWORD state;

AxmHomeGetSignalLevel(0, &state);
```

VB Example

```
' Set home input signal level on axis 0
AxmHomeSetSignalLevel 0, HIGH

'Verify home input signal level on axis 0
Dim state As Long

AxmHomeGetSignalLevel 0, state
```

Delphi Example

```
{ Verify home input signal level on axis 0 }
var
state: DWord;

Begin
{ Set home input signal level on axis 0 }
AxmHomeSetSignalLevel (0, HIGH);
AxmHomeGetSignalLevel (0, @state);
end;
```

See Also

[AxmHomeGetSignalLevel](#), [AxmHomeReadSignal](#), [AxmHomeSetMethod](#), [AxmHomeGetMethod](#),
[AxmHomeSetVel](#), [AxmHomeGetVel](#), [AxmHomeSetStart](#), [AxmHomeSetResult](#),
[AxmHomeGetResult](#), [AxmHomeGetRate](#), [AxmHomeGetResult](#), [AxmHomeGetRate](#)

AxmHomeGetSignalLevel

Purpose

Returns output level settings of home signal of a specific axis.

Format

C

```
DWORD AxmHomeGetSignalLevel (long IAxisNo, DWORD *upLevel);
```

Visual Basic

```
Function AxmHomeGetSignalLevel (ByVal IAxisNo As Long, ByRef upLevel As Long) As Long
```

Delphi

```
function AxmHomeGetSignalLevel (IAxisNo : LongInt; upLevel : PDWord) : DWord; stdcall;
```

Input / Output

Name	in/out	Format	Init Value	Explanation
AxisNo	in	long		Channel (axis) number(starting from 0)
Level	out	DWORD*	1	Active Level of Home signal

Level

#define	Value	Explanation
LOW	00h	B contact (Normal Close)
HIGH	01h	A contact (Normal Open)

Return AXT_RT_SUCCESS(0000) : Successful execution of API.

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

Description.

It returns the home input level set by '[AxmHomeSetSignalLevel](#)' ,

C Example

```
// Set home input signal level on axis
AxmHomeSetSignalLevel (0, HIGH);
AxmHomeSetSignalLevel// Verify home input signal level on axis 0
DWORD state;

AxmHomeGetSignalLevel (0, &state);
```

VB Example

```
' Set home input signal level on axis
AxmHomeSetSignalLevel 0, HIGH

'Verify home input signal level on axis 0
Dim state As Long

AxmHomeGetSignalLevel 0, state
```

Delphi Example

```
{ Verify home input signal level on axis 0 }
var
state: DWord;

Begin
{ Set home input signal level on axis }
AxmHomeSetSignalLevel (0, HIGH);
AxmHomeGetSignalLevel (0, @state);
end;
```

See Also

[AxmHomeSetSignalLevel](#), [AxmHomeReadSignal](#), [AxmHomeSetMethod](#), [AxmHomeGetMethod](#),
[AxmHomeSetVel](#), [AxmHomeGetVel](#), [AxmHomeSetStart](#), [AxmHomeSetResult](#),
[AxmHomeGetResult](#), [AxmHomeGetRate](#), [AxmHomeGetResult](#), [AxmHomeGetRate](#)

AxmHomeReadSignal

Purpose

Return the input state of home signal of a specific axis.

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		Channel (axis) number(starting from 0)
Status	out	DWORD*		Input state of a signal

Status

#define	Value	Explanation
INACTIVE	00h	No-activation
ACTIVE	01h	Activation

Return AXT_RT_SUCCESS(0000) : Successful execution of API.

[* See error code Table for more information on status error codes](#)

Description

It returns the input state of the current home signal.

C Example

```
// Verify home signal input on axis 0
DWORD upStatus;
AxmHomeReadSignal (0, &upStatus);
```

VB Example

```
' Verify home signal input on axis 0
Dim upStatus As Long
AxmSignalReadInpos 0, upStatus
```

Delphi Example

```
{ Verify home signal input on axis 0 }
var
  upStatus: DWord;
begin
```

```
AxmHomeReadSignal (0, @upStatus);  
end;
```

See Also

[AxmHomeSetSignalLevel](#), [AxmHomeGetSignalLevel](#), [AxmHomeSetMethod](#), [AxmHomeGetMethod](#),
[AxmHomeSetVel](#), [AxmHomeGetVel](#), [AxmHomeSetStart](#), [AxmHomeSetResult](#),
[AxmHomeGetResult](#), [AxmHomeGetRate](#), [AxmHomeGetResult](#), [AxmHomeGetRate](#)

AxmHomeSetMethod

Purpose

To execute home search of the corresponding axis, the parameters related to home search must be preset. If initialization has been normally executed using AxmMotLoadParaAll API setting file, no further setup is required..

Home search method setting includes searching direction, the signal to be used as home, home sensor active level, and whether encoder Z phase is detected, etc. (Refer to the description part of AxmMotLoadParaAll for details)

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		Channel(axis) number (start from 0) to verify whether an initial value has been set using AxmMotLoadParaAll SMC-2V03, PCI-N804/404 file
HmDir	In	long	0	Sets the direction to execute home search at initial stage
uHomeSigna l	In	DWORD	4	Sets the signal to be used for home search
uZphas	In	DWORD	0	Sets whether encoder Z phasie is detected after the first home search is completed
dHomeClrTi me	In	double	1000	Sets the standby time to clear the command position and the encoder position after home search is completed [per mSec]
dpHomeOffs	In	double	0	Position at which home will be reset after

et				home search is completed and is moved to mechanical home
----	--	--	--	--

IHmDir

#define	Value	Explanation
DIR_CCW	00h	Clockwise
DIR_CW	01h	Counterclockwise

uHomeSignal

#define	Value	Explanation
PosEndLimit	00h	+Elm(End limit) + direction limit sensor signal
NegEndLimit	01h	-Elm(End limit) - direction limit sensor signal
HomeSensor	04h	IN0(ORG) home sensor signal

uZphas

#define	Value	Explanation
DISABLE	00h	Z phase detection non-used
ENABLE	01h	Z phase detection used

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

Description

To execute home search of the corresponding axis, the parameters related to home search must be preset. If initialization has been normally executed using AxmMotLoadParaAll API setting file, no further setup is required. Home search method setting includes searching direction, the signal to be used as home, and whether encoder Z phase is detected, etc. Home sensor active level is independently set using AxmSignalSetHomeLevel API. (Refer to the description part of AxmMotLoadParaAll for details) Even if the level is incorrectly set in the – direction, it can still move in the + direction; It may cause a problem during home search.

C Example

```
long lHmDir = 1;
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);

// Set home Search in the (-) direction, home sensor used, home sensor
// type A contact, Z phase detection non-used, offset non-moved

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 = -1
lHmsig = 4
dwZphas = 0
` Set home Search in the (-) direction, home sensor used, home sensor type
  A contact, Z phase detection non-used, offset non-moved

AxmHomeSetMethod 0, lHmDir, lHmsig, dwZphas, dwHClrTim, dwHOffset
AxmHomeSetVel 0, 10000, 5000, 500, 50, 40000, 10000
AxmHomeSetStart 0
AxmHomeGetMethod0, lHmDir, lHmsig, dwZphas, dwHClrTim, dwHOffset

```

Delphi Example

```

var
  dwZphas, lHmsig : DWORD;
  dwHOffset, dwHClrTim : Double;
  lHmDir : LongInt;
  velocity : Double;

begin

lHmDir := -1;
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);
{ Set home Search in the (-) direction, home sensor used, home sensor
  type A contact, Z phase detection non-used, offset non-moved }
AxmHomeGetMethod (0, @lHmDir, @lHmsig, @dwZphas, @dwHClrTim, @dwHOffset);
End;

```

See Also

[AxmHomeSetSignalLevel](#), [AxmHomeGetSignalLevel](#), [AxmHomeReadSignal](#), [AxmHomeGetMethod](#),
[AxmHomeSetVel](#), [AxmHomeGetVel](#), [AxmHomeSetStart](#), [AxmHomeSetResult](#),
[AxmHomeGetResult](#), [AxmHomeGetRate](#), [AxmHomeGetResult](#), [AxmHomeGetRate](#)

AxmHomeGetMethod

Purpose

To execute home search of the corresponding axis, the parameters related to home search must be preset. If initialization has been normally executed using AxmMotLoadParaAll API setting file, no further setup is required. It reads searching direction, the signal to be used as home, and whether encoder Z phase is detected, etc. in home search method setting. (Refer to the description part of AxmMotLoadParaAll for details)

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/o ut	Format	Init Value	Explanation
AxisNo	in	long		Channel(axis) number (start from 0) to verify whether an initial value has been set using AxmMotLoadParaAll SMC-2V03, PCI-N804/404 file
HmDir	out	long*	0	Sets the direction to execute home search at initial stage
uHomeSig nal	out	DWORD*	1	Sets the signal to be used for home search
uZphas	out	DWORD*	0	Sets whether encoder Z phasie is detected after the first home search is completed
dHomeClr Time	out	double*	1000	Sets the standby time to clear the command position and the encoder position after home search is completed [per mSec]
dpHomeO ffset	out	double*	0	Position at which home will be reset after home search is completed and is moved to mechanical home

lHmDir

#define	Value	Explanation
DIR_CCW	00h	Counterclockwise
DIR_CW	01h	Clockwise

uHomeSignal

#define	Value	Explanation
PosEndLimit	00h	+Elm(End limit) + direction limit sensor signal
NegEndLimit	01h	-Elm(End limit) - direction limit sensor signal
HomeSensor	04h	IN0(ORG) home sensor signal

uZphas

#define	Value	Explanation
DISABLE	00h	Z phase detection non-used
ENABLE	01h	Z phase detection used

Return AXT_RT_SUCCESS(0000) : Successful execution of API.

[* See error code Table for more information on status error codes](#)**Description**

To execute home search of the corresponding axis, the parameters related to home search must be preset. If initialization has been normally executed using AxmMotLoadParaAll API setting file, no further setup is required. Home search method setting includes searching direction, the signal to be used as home, and whether encoder Z phase is detected, etc. Home sensor active level is independently set using AxmSignalGetHomeLevel API. (Refer to the description part of AxmMotLoadParaAll for details)

C Example

```

long lHmDir = 1;
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);

// Set home search in the (-) direction, home sensor used, home sensor type A
// contact, Z phase detection non-used, offset non-moved

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 = -1
lHmsig = 4

```

```

dwZphas = 0
' Set home search in the (-) direction, home sensor used, home sensor type A
    contact, Z phase detection non-used, offset non-moved

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 := -1;
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);
{ Set home search in the (-) direction, home sensor used, home sensor type A
    contact, Z phase detection non-used, offset non-moved }
AxmHomeGetMethod (0, @lHmDir, @lHmsig, @dwZphas, @dwHClrTim, @dwHOffset);
End;

```

See Also

[AxmHomeSetSignalLevel](#), [AxmHomeGetSignalLevel](#), [AxmHomeReadSignal](#), [AxmHomeSetMethod](#),
[AxmHomeSetVel](#), [AxmHomeGetVel](#), [AxmHomeSetStart](#), [AxmHomeSetResult](#),
[AxmHomeGetResult](#), [AxmHomeGetRate](#), [AxmHomeGetResult](#), [AxmHomeGetRate](#)

AxmHomeSetVel

Purpose

It detects home through multi-level steps to search it quickly and accurately. At this point, it sets the velocity to be used for each step. Time and accuracy of home search is determined based on the setting of these velocities. You can set the home search velocities while adjusting velocities in each step accordingly. (Refer to the description part of AxmMotLoadParaAll for details)

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	–	Channel (axis) number(starting from 0)
dVelFirst	In	double	10000	Velocity at which to move in the direction of home search when home sensor is not detected
dVelSecond	In	double	10000	Velocity at which to move in the opposite direction of home search when home sensor is detected
dVelThird	In	double	2000	Moving velocity to be used for the second search after the first home sensor detection
dvelLast	In	double	100	Moving velocity to be used for the last home search (Velocity at which to use Z phase detection, detailed searching velocity of home sensor)
dAccFirst	In	double	40000	Ac/deceleration to be used when home sensor is not detected and it is moving in the direction of home search (The unit of

				deceleration is PPS[Pulses/sec] when Unit/pulse is set to 1/1)
dAccSecond	In	double	40000	Ac/deceleration to be used when home sensor is detected and it is moving in the opposite direction of home search (The unit of deceleration is PPS[Pulses/sec] when Unit/pulse is set to 1/1)

Return AXT_RT_SUCCESS(0000) : Successful execution of API.

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

Description

It detects home through multi-level steps to search it quickly and accurately. At this point, it sets the velocity to be used for each step. Time and accuracy of home search is determined based on the setting of these velocities. You can set the home search velocity while adjusting it in each step accordingly. (Refer to the description part of AxmMotLoadParaAll for details)

C Example

```
long lHmDir = 1;
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);

// Set home search in the (-) direction, home sensor used, home sensor
// type A contact, Z phase detection non-used, offset non-moved

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 = -1
lHmsig = 4
dwZphas = 0
' Set home search in the (-) direction, home sensor used, home sensor type
' A contact, Z phase detection non-used, offset non-moved

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 := -1;
  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);
{ Set home search in the (-) direction, home sensor used, home sensor type
  A contact, Z phase detection non-used, offset non-moved}
  AxmHomeGetMethod (0, @lHmDir, @lHmsig, @dwZphas, @dwHClrTim, @dwHOffset);
End;
```

See Also

[AxmHomeSetSignalLevel](#), [AxmHomeGetSignalLevel](#), [AxmHomeReadSignal](#), [AxmHomeSetMethod](#),
[AxmHomeGetMethod](#), [AxmHomeSetVel](#), [AxmHomeGetVel](#), [AxmHomeSetStart](#), [AxmHomeSetResult](#),
[AxmHomeGetResult](#), [AxmHomeGetRate](#), [AxmHomeGetResult](#), [AxmHomeGetRate](#)

AxmHomeGetVel

Purpose

It detects home through multi-level steps to search it quickly and accurately, At this point, it returns the velocity to be used for each step.

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	–	Channel (axis) number(starting from 0)
dVelFirst	out	double*	10000	Velocity at which to move in the direction of home search when home sensor is not detected
dVelSecond	out	double*	10000	Velocity at which to move in the opposite direction of home search when home sensor is detected
dVelThird	out	double*	2000	Moving velocity to be used for the second search after the first home sensor detection
dvelLast	out	double*	100	Moving velocity to be used for the last home search (Velocity at which to use Z phase detection, detailed searching velocity of home sensor)
dAccFirst	out	double*	40000	Ac/deceleration to be used when home sensor is not detected and it is moving in the direction of home search (The unit of deceleration is PPS[Pulses/sec] when Unit/pulse is set to 1/1)

dAccSecond	out	double*	40000	Ac/deceleration to be used when home sensor is detected and it is moving in the opposite direction of home search (The unit of deceleration is PPS[Pulses/sec] when Unit/pulse is set to 1/1)
------------	-----	---------	-------	---

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

Description

Tip : It is possible to get only necessary data by inserting NULL into unnecessary variables.

It detects home through multi-level steps to search it quickly and accurately, At this point, it sets the velocity to be used for each step. Time and accuracy of home search is determined based on the setting of these velocities. You can set the home search velocity while adjusting it in each step accordingly. (Refer to the description part of AxmMotLoadParaAll for details)

C Example

```
long lHmDir = 1;
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);

// Set to home search in the (-) direction, home sensor used, home sensor
// type A contact, Z phase detection non-used, offset non-moved

AxmHomeGetMethod (0, &lHmDir, &lHmsig, &dwZphas, &dwHClrTim, &
                  dwHOffset);
AxmHomeGetVel(0, & dVelFirst, & dVelSecond, & dVelThird, & dVelLast,
              &dAccFirst, & dAccSecond);
```

VB Example

```
Dim dpS1Offset As Double
Dim dS1ORange As Double
Dim bS1HomeUse, bGtryOn, bMaS1Change 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 = -1
lHmsig = 4
dwZphas = 0

' Set to home search in the (-) direction, home sensor used, home sensor
// type A contact, Z phase detection non-used, offset non-moved
```

```

AxmHomeSetMethod0, lHmDir, lHmsig, dwZphas, dwHClrTim, dwHOffset
AxmHomeSetVel 0, 10000, 5000, 500, 50, 40000, 10000
AxmHomeSetStart 0

AxmHomeGetMethod0, lHmDir, lHmsig, dwZphas, dwHClrTim, dwHOffset
AxmHomeGetVel 0, dVelFirst, dVelSecond, dVelThird, dVelLast, dAccFirst,
dAccSecond

```

Delphi Example

```

var
  dwZphas, bSlHomeUse, bGtryOn, bMaSlChange : DWORD;
  dwHOffset, dwHClrTim, dpSlOffset,dSlORange : Double;
  dVelFirst, dVelSecond, dVelThird,dVelLast, dAccFirst,dAccSecond : Double;
  lHmDir, lHmsig : LongInt;
  velocity : Double;

begin

lHmDir := -1;
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);

{ Set to home search in the (-) direction, home sensor used, home sensor
  type A contact, Z phase detection non-used, offset non-moved }

AxmHomeGetMethod (0, @lHmDir, @lHmsig, @dwZphas, @dwHClrTim, @dwHOffset);
AxmHomeGetVel (0, @dVelFirst, @dVelSecond, @dVelThird, @dVelLast,
@dAccFirst, @dAccSecond);
End;

```

See Also

[AxmHomeSetSignalLevel](#), [AxmHomeGetSignalLevel](#), [AxmHomeReadSignal](#), [AxmHomeSetMethod](#),
[AxmHomeGetMethod](#), [AxmHomeSetVel](#), [AxmHomeSetStart](#), [AxmHomeSetResult](#),
[AxmHomeGetResult](#), [AxmHomeGetRate](#), [AxmHomeGetResult](#), [AxmHomeGetRate](#)

AxmHomeSetStart

Purpose

If home search starting API is executed, the thread to execute home search of the corresponding axis is automatically created within the library. Then it sequentially executes home search and is exited

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	-	Channel (axis) number(starting from 0) (0 ~ n-1)

Return AXT_RT_SUCCESS(0000) : Successful execution of API.

[* See error code Table for more information on status error codes](#)

Description

If home search starting API is executed, the thread to execute home search of the corresponding axis is automatically created within the library. Then it sequentially executes home search and is exited.

Note: Even if limit sensor of opposite direction against process direction is on, it does not work without the activation of sensor of process direction. If home search is started and limit sensor of process direction is on, it moves to the next step as it thought limit sensor is detected.

C Example

```
DWORD uHomeResult;
if(AxmHomeSetStart (0) == AXT_RT_SUCCESS) // Execute home search
printf("Starts home search of axis 0.");
else{
AxmHomeGetResult (0, & uHomeResult);
printf("Home search of axis 0 cannot start. [Error Code:%d]",
uHomeResult);
}
```

VB Example

```
Dim nCount As Integer
Dim strData As String
Dim uHomeResult As Long
If (AxmHomeSetStart (0) = AXT_RT_SUCCESS )Then      ' Execute home search
MsgBox "Starts home search of axis 0.", vbOKCancel
```

```
Else
AxmHomeGetResult 0, uHomeResult
nCount= uHomeResult
strData = "Home search of axis 0 cannot start. Error Code" + CStr(nCount)
        + "."
MsgBox strData
End If
```

Delphi Example

```
var
nCount : Word; strData : String;
uHomeResult: Dword

begin
if (AxmHomeSetStart (0) = AXT_RT_SUCCESS ) then { Execute home search }
Application.MessageBox('Starts home search of axis 0.', 'Ajinextek',
MB_OK)
else
begin
AxmHomeGetResult (0, @ uHomeResult);
nCount := uHomeResult;
strData := 'Home search of axis 0 cannot start. Error Code ' +
          IntToStr(nCount) + '.';
Application.MessageBox (PCHAR(strData), 'Ajinextek', MB_OK);
end;
End;
```

See Also

[AxmHomeSetSignalLevel](#), [AxmHomeGetSignalLevel](#), [AxmHomeReadSignal](#), [AxmHomeSetMethod](#),
[AxmHomeGetMethod](#), [AxmHomeSetVel](#), [AxmHomeGetVel](#), [AxmHomeSetStart](#), [AxmHomeSetResult](#),
[AxmHomeGetResult](#), [AxmHomeGetRate](#), [AxmHomeGetResult](#), [AxmHomeGetRate](#)

AxmHomeSetResult

Purpose

It outputs the search result after home search is successfully executed using home search API. After home search is successfully executed using home search API, the search result is set to HOME SUCCESS. This API can set the result at user's discretion without executing home search.

In practice, if an unexpected situation occurs during a normal move, the user can set the home search result to HOME_ERR_USER_BREAK or so to recover executing home research.

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		Channel(axis) number (start from 0) to set home search result at user's discretion
HomeResult	out	double		Home search result

HomeResult

#define	Value	Explanation
HOME_SUCCESS	01h	Home search has been successfully exited
HOME_SEARCHING	02h	Home search is currently in progress
HOME_ERR_GNT_RANGE	10h	Home search results of Master axis and Slave axis of gantry move axes are out of OffsetRange
HOME_ERR_USER_BREAK	11h	User executed stop command during home search
HOME_ERR_VELOCITY	12h	At least one value among 4 home searching velocity setting values is less than or equal to 0
HOME_ERR_AMP_FAULT	13h	Servo pack alarm has occurred during home search
HOME_ERR_NEG_LIMIT	14h	(-) limit sensor has been detected during the home sensor search in (+) direction

HOME_ERR_POS_LIMIT	15h	(+) limit sensor has been detected during the home sensor search in (-) direction
HOME_ERR_NOT_DETECT	16h	Home sensor has not been detected
HOME_ERR_UNKNOWN	FFh	Home search has been attempted using unknown channel(axis) number(start from 0)

Return AXT_RT_SUCCESS(0000) : Successful execution of API.

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

Description

After home search is successfully executed using home search API, the search result is set to HOME SUCCESS. This API can set the result at user's discretion without executing home search.

In practice, if an unexpected situation occurs during a normal move, the user can set the home search result to HOME_ERR_USER_BREAK or so to recover executing home research.

C Example

```
if(AxmHomeSetResult (0, HOME_ERR_USER_BREAK) == AXT_RT_SUCCESS){
    printf("Home search result on axis 0 has been set to USER BREAK.");
} else{
    printf("Home search result setting on axis 0 has failed.");
}

BOOL bRun = TRUE;
DWORD uHomeResult;
AxmHomeSetStart (0);

while(bRun)
{
    // Output the search result after home search is successfully executed
    // using home search API.
    AxmHomeGetResult (0, &uHomeResult);
    switch(uHomeResult)
    {
        case HOME_ERR_UNKNOWN: printf("Unknown axis number"); bRun=FALSE; break;
        case HOME_ERR_GNT_RANGE: printf("Out of Gantry Offset"); bRun=FALSE;
            break;
        case HOME_ERR_USER_BREAK: printf("Home search stops."); bRun=FALSE;
            break;
        case HOME_ERR_VELOCITY: printf("Velocity has not been set.");
            bRun=FALSE; break;
        case HOME_ERR_LAMPFAULT: printf("Servo pack alarm has occurred");
            bRun = FALSE; break;
        case HOME_ERR_NEG_LIMIT: printf("(+) limit sensor has been detected");
            bRun = FALSE; break;
        case HOME_ERR_POS_LIMIT: printf("(-) limit sensor has been detected");
            bRun = FALSE; break;
        case HOME_SEARCHING: printf("Home search is now in progress"); Sleep(100);
            break;
        case HOME_SUCCESS: printf("Home search was successful");
            bRun = FALSE; break;
    }
}
```

VB Example

```

If (AxmHomeSetResult (0, HOME_ERR_USER_BREAK) = AXT_RT_SUCCESS )Then
  MsgBox "Home search result on axis 0 has been set to USER BREAK.", vbOKCancel
Else
  MsgBox "Home search result setting on axis 0 has failed.", vbOKCancel
End If

Dim bFlag As Byte
Dim uID As Long
Dim uHomeResult As Long
AxmHomeSetStart 0
'Output the search result after home search is successfully executed using
'home search API.
bFlag = True
Do While bFlag
  AxmHomeGetResult 0, uHomeResult
  uID = uHomeResult
  Select Case uID
  Case HOME_ERR_UNKNOWN:
    MsgBox "Unknown axis number", vbOKCancel
    bFlag = False
  Case HOME_ERR_GNT_RANGE:
    MsgBox "Out of Gantry Offset", vbOKCancel
    bFlag = False
  Case HOME_ERR_USER_BREAK:
    MsgBox "Home search stops", vbOKCancel
    bFlag = False
  Case HOME_ERR_VELOCITY:
    MsgBox "Velocity has not been set.", vbOKCancel
    bFlag = False
  Case HOME_ERR_AMPFAULT:
    MsgBox "Servo pack alarm has occurred", vbOKCancel
    bFlag = False
  Case HOME_ERR_NEG_LIMIT:
    MsgBox "(-) limit sensor has been detected", vbOKCancel
    bFlag = False
  Case HOME_ERR_POS_LIMIT:
    MsgBox "(+) limit sensor has been detected", vbOKCancel
    bFlag = False
  Case HOME_SEARCHING:
    MsgBox "Home search is now in progress", vbOKCancel
    bFlag = False
  Case HOME_SUCCESS:
    MsgBox "Home search was successful", vbOKCancel
    bFlag = False
  End Select
Loop

```

Delphi Example

```

begin
if (AxmHomeSetResult (0, HOME_ERR_USER_BREAK) = AXT_RT_SUCCESS) then
  Application.MessageBox('Home search result on axis 0 has been set to USER
  BREAK.', 'Ajinextek', MB_OK)
else
  Application.MessageBox('Home search result setting on axis 0 has failed.',
  'Ajinextek', MB_OK);
End;

var
uID : Word; bFlag : Boolean; bFlag := True;
uHomeResult: Dword

```

```

begin
AxmHomeSetStart (0);
{ Output the search result after home search is successfully executed
  using home search API.}
while bFlag do
begin
AxmHomeGetResult (0, @uHomeResult);
uID := uHomeResult;
case (uID) of
HOME_ERR_UNKNOWN: begin
Application.MessageBox ('Unknown axis number.', 'Ajinextek', MB_OK);
  bFlag := False; end;
HOME_ERR_GNT_RANGE: begin
Application.MessageBox ('Out of Gantry Offset', 'Ajinextek', MB_OK);
  bFlag := False; end;
HOME_ERR_USER_BREAK: begin
Application.MessageBox ('Home search stops.', 'Ajinextek', MB_OK);
  bFlag := False; end;
HOME_ERR_VELOCITY: begin
Application.MessageBox ('Velocity has not been set.', 'Ajinextek', MB_OK);
  bFlag := False; end;
HOME_ERR_AMP_FAULT: begin
Application.MessageBox ('Servo pack alarm has occurred.', 'Ajinextek',
  MB_OK); bFlag := False; end;
HOME_ERR_NEG_LIMIT: begin
Application.MessageBox ('(-) limit sensor has been detected.',
  'Ajinextek', MB_OK); bFlag := False; end;
HOME_ERR_POS_LIMIT: begin
Application.MessageBox ('(+) limit sensor has been detected', 'Ajinextek',
  MB_OK); bFlag := False; end;
HOME_SEARCHING: begin
Application.MessageBox ('Home search is now in progress', 'Ajinextek',
  MB_OK); bFlag := False; end;
HOME_SUCCESS: begin
Application.MessageBox ('Home search was successful', 'Ajinextek', MB_OK);
  bFlag := False; end;
end; end;
End;

```

See Also

[AxmHomeSetSignalLevel](#), [AxmHomeGetSignalLevel](#), [AxmHomeReadSignal](#), [AxmHomeSetMethod](#),
[AxmHomeGetMethod](#), [AxmHomeSetVel](#), [AxmHomeGetVel](#), [AxmHomeSetStart](#),
[AxmHomeGetResult](#), [AxmHomeGetRate](#), [AxmHomeGetResult](#), [AxmHomeGetRate](#)

AxmHomeGetResult

Purpose

It verifies the search result of home search API. If home search starts, it is set to HOME_SEARCHING, and if it fails, the cause of failure is set. You can eliminate the cause of failure and restart home search.

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		Channel(axis) number (start from 0) to set home search result at user's discretion
HomeResult	out	DWORD *		Home search result

HomeResult

#define	Value	Explanation
HOME_SUCCESS	01h	Home search has been successfully exited
HOME_SEARCHING	02h	Home search is currently in progress
HOME_ERR_GNT_RANGE	10h	Home search results of Master axis and Slave axis of gantry move axes are out of OffsetRange
HOME_ERR_USER_BREAK	11h	User executed stop command during home search
HOME_ERR_VELOCITY	12h	At least one value among 4 home searching velocity setting values is less than or equal to 0
HOME_ERR_AMP_FAULT	13h	Servo pack alarm has occurred during home search
HOME_ERR_NEG_LIMIT	14h	(-) limit sensor has been detected during the home sensor search in (+) direction
HOME_ERR_POS_LIMIT	15h	(+) limit sensor has been detected during the home sensor search in (-) direction

HOME_ERR_NOT_DETECT	16h	Home sensor has not been detected
HOME_ERR_UNKNOWN	FFh	Home search has been attempted using unknown channel(axis) number(start from 0)

Return AXT_RT_SUCCESS(0000) : Successful execution of API.

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

* Tip: After home search has succeeded, if the home search is executed again, its result is initialized and reset. If it is stopped while executing once again using stop command([AxmMoveSStop](#)) API, it is set to HOME_USER_BREAK.

Description

It verifies the search result of home search API. If home search starts, it is set to HOME_SEARCHING, and if it fails, the cause of failure is set. You can eliminate the cause of failure and restart home search.

C Example

```
if(AxmHomeSetResult (0, HOME_ERR_USER_BREAK) == AXT_RT_SUCCESS){
    printf("Home search result on axis 0 has been set to USER BREAK.");
} else{
    printf("Home search result setting on axis 0 has failed.");
}

BOOL bRun = TRUE;
DWORD uHomeResult;
AxmHomeSetStart (0);

while(bRun)
{
    // Verify home search result
    AxmHomeGetResult(0, &uHomeResult);
    switch(uHomeResult)
    {
        case HOME_ERR_UNKNOWN: printf("Unknown axis number");
            bRun = FALSE; break;
        case HOME_ERR_GNT_RANGE: printf("Out of Gantry Offset");
            bRun = FALSE; break;
        case HOME_ERR_USER_BREAK: printf("Home search stops");
            bRun = FALSE; break;
        case HOME_ERR_VELOCITY: printf("Velocity has not been set.");
            bRun = FALSE; break;
        case HOME_ERR_AMP_FAULT: printf("Servo pack alarm has occurred");
            bRun = FALSE; break;
        case HOME_ERR_NEG_LIMIT: printf("(--) limit sensor has been detected");
            bRun = FALSE; break;
        case HOME_ERR_POS_LIMIT: printf("(+) limit sensor has been detected");
            bRun = FALSE; break;
        case HOME_SEARCHING: printf("Home search is now in progress");Sleep(100);
            break;
        case HOME_SUCCESS: printf("Home search was successful");
            bRun = FALSE; break;
    }
}
```

VB Example

```
If (AxmHomeSetResult (0, HOME_ERR_USER_BREAK) = AXT_RT_SUCCESS )Then
```

```

MsgBox "Home search result on axis 0 has been set to USER BREAK.", vbOKCancel
Else
MsgBox "Home search result setting on axis 0 has failed.", vbOKCancel
End If

Dim bFlag As Byte
Dim uID As Long
Dim uHomeResult As Long
AxmHomeSetStart 0

bFlag = True
Do While bFlag
' Verify home search result
AxmHomeGetResult 0, uHomeResult
uID = uHomeResult
Select Case uID
Case HOME_ERR_UNKNOWN:
MsgBox "Unknown axis number", vbOKCancel
bFlag = False
Case HOME_ERR_GNT_RANGE:
MsgBox "Out of Gantry Offset", vbOKCancel
bFlag = False
Case HOME_ERR_USER_BREAK:
MsgBox "Home search stops", vbOKCancel
bFlag = False
Case HOME_ERR_VELOCITY:
MsgBox "Velocity has not been set.", vbOKCancel
bFlag = False
Case HOME_ERR_AMP_FAULT:
MsgBox "Servo pack alarm has occurred", vbOKCancel
bFlag = False
Case HOME_ERR_NEG_LIMIT:
MsgBox "(-) limit sensor has been detected", vbOKCancel
bFlag = False
Case HOME_ERR_POS_LIMIT:
MsgBox "(+) limit sensor has been detected", vbOKCancel
bFlag = False
Case HOME_SEARCHING:
MsgBox "Home search is now in progress", vbOKCancel
bFlag = False
Case HOME_SUCCESS:
MsgBox "Home search was successful", vbOKCancel
bFlag = False
End Select
Loop

```

Delphi Example

```

begin
if (AxmHomeSetResult (0, HOME_ERR_USER_BREAK) = AXT_RT_SUCCESS) then
Application.MessageBox('Home search result on axis 0 has been set to USER
BREAK.', 'Ajinextek', MB_OK)
else
Application.MessageBox('Home search result setting on axis 0 has failed.',
'Ajinextek', MB_OK);
End;

var
uID : Word; bFlag : Boolean; bFlag := True;
uHomeResult: Dword

begin
AxmHomeSetStart (0);

while bFlag do

```

```
begin
{ Verify home search result }
AxmHomeGetResult(0, @uHomeResult);
uID := uHomeResult;
case (uID) of
HOME_ERR_UNKNOWN: begin
Application.MessageBox ('Unknown axis number.', 'Ajinextek', MB_OK);
bFlag := False; end;
HOME_ERR_GNT_RANGE: begin
Application.MessageBox ('Out of Gantry Offset', 'Ajinextek', MB_OK);
bFlag := False; end;
HOME_ERR_USER_BREAK: begin
Application.MessageBox ('Home search stops.', 'Ajinextek', MB_OK);
bFlag := False; end;
HOME_ERR_VELOCITY: begin
Application.MessageBox ('Velocity has not been set.', 'Ajinextek', MB_OK);
bFlag := False; end;
HOME_ERR_AMPFAULT: begin
Application.MessageBox ('Servo pack alarm has occurred.', 'Ajinextek',
MB_OK); bFlag := False; end;
HOME_ERR_NEG_LIMIT: begin
Application.MessageBox ('(-) limit sensor has been detected.',
'Ajinextek', MB_OK); bFlag := False; end;
HOME_ERR_POS_LIMIT: begin
Application.MessageBox ('(+) limit sensor has been detected', 'Ajinextek',
MB_OK); bFlag := False; end;
HOME_SEARCHING: begin
Application.MessageBox ('Home search is now in progress', 'Ajinextek',
MB_OK); bFlag := False; end;
HOME_SUCCESS: begin
Application.MessageBox ('Home search was successful', 'Ajinextek', MB_OK);
bFlag := False; end;
end; end;
End;
```

See Also

[AxmHomeSetSignalLevel](#), [AxmHomeGetSignalLevel](#), [AxmHomeReadSignal](#), [AxmHomeSetMethod](#),
[AxmHomeGetMethod](#), [AxmHomeSetVel](#), [AxmHomeGetVel](#), [AxmHomeSetStart](#), [AxmHomeSetResult](#),
[AxmHomeGetResult](#), [AxmHomeGetRate](#), [AxmHomeGetRate](#)

AxmHomeGetRate

Purpose

You can check the progress rate after home search starts. After home search is completed, it returns 100 regardless of whether or not it was successful. You can verify whether or not home search was successful using GetHome Result API.

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 ; upHomeStepN umber : PDWord) :  
DWord; stdcall;
```

Input / output

Name	in/out	Format	Init Value	Explanation
AxisNo	in	long		Channel(axis) number (start from 0) to verify home search progress rate
HomeMainStepNumber	out	DWORD*		Main Step progress rate(0 – 100) – If gantry is FALSE and upHomeMainStep Number : 0, only the selected axis is in progress and the home progress rate displays upHomeStepNumber. – If gantry is TRUE and upHomeMainStep Number : 0, the master home is in progress and the master home progress rate displays upHomeStepNumber. – If gantry is TRUE and upHomeMainStep Number : 10, the slave home is in progress and the master home progress rate displays upHomeStepNumber.
HomeStepNumber	out	DWORD*		Verifies home search progress rate

HomeStepNumber

upHomeStepNumber	Value	Explanation
DWORD value	0	When the library is not initialized; when it is an invalid channel(axis) number (start from 0); when home search is not in progress (unit:%)
DWORD value	1~99	Home search progress rate(unit:%)
DWORD value	100	When home search is completed(unit:%)

Return AXT_RT_SUCCESS(0000) : Successful execution of API.

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

Description

You can check the progress rate after home search starts. After home search is completed, it returns 100 regardless of whether or not it was successful. You can verify whether or not home search was successful using GetHome Result API.

C Example

```

BOOL bRun = TRUE;
DWORD uHomeResult, uHomeStepNumber, uHomeMainStepNumber;
AxmHomeSetStart (0);

while(bRun)
{
AxmHomeGetResult (0, &uHomeResult);
switch(uHomeResult)
{
case HOME_ERR_UNKNOWN: printf("Unknown axis number");
bRun = FALSE; break;
case HOME_ERR_GNT_RANGE: printf("Out of Gantry Offset");
bRun = FALSE; break;
case HOME_ERR_USER_BREAK: printf("Home search stops");
bRun = FALSE; break;
case HOME_ERR_VELOCITY: printf("Velocity has not been set.");
bRun = FALSE; break;
case HOME_ERR_AMPFAULT: printf("Servo pack alarm has occurred");
bRun = FALSE; break;
case HOME_ERR_NEG_LIMIT: printf("(--)limit sensor has been detected");
bRun = FALSE; break;
case HOME_ERR_POS_LIMIT: printf("(+)limit sensor has been detected");
bRun = FALSE; break;
case HOME_SEARCHING:
// Verify progress rate
AxmHomeGetRate( 0, &uHomeMainStepNumber , &uHomeStepNumber );
printf("\tHome search is now %d in progress", uHomeStepNumber);
Sleep(100);
break;
case HOME_SUCCESS: printf("Home search was successful");
bRun = FALSE break;
}
}

```

VB Example

```

Dim bFlag As Byte
Dim uID As Long
Dim uHomeResult, uHomeMainStepNumber, uHomeStepNumber As Long

```

```

bFlag = True
AxmHomeSetStart0
Do While bFlag
  AxmHomeGetResult 0, uHomeResult
  uID      = uHomeResult
  Select Case uID
    Case HOME_ERR_UNKNOWN:
      MsgBox "Unknown axis number", vbOKCancel
      bFlag = False
    Case HOME_ERR_GNT_RANGE:
      MsgBox "Out of Gantry Offset", vbOKCancel
      bFlag = False8
    Case HOME_ERR_USER_BREAK:
      MsgBox "Home search stops", vbOKCancel
      bFlag = False
    Case HOME_ERR_VELOCITY:
      MsgBox "Velocity has not been set.", vbOKCancel
      bFlag = False
    Case HOME_ERR_AMP_FAULT:
      MsgBox "Servo pack alarm has occurred", vbOKCancel
      bFlag = False
    Case HOME_ERR_NEG_LIMIT:
      MsgBox "(-)limit sensor has been detected", vbOKCancel
      bFlag = False
    Case HOME_ERR_POS_LIMIT:
      MsgBox "(+)limit sensor has been detected", vbOKCancel
      bFlag = False
    Case HOME_SEARCHING:
      ' Verify progress rate
      AxmHomeGetRate 0, uHomeMainStepNumber , uHomeStepNumber
      nCount = uHomeStepNumber
      strData = "Home search is now in progress. The rate is " +
                CStr(nCount) + "."
      MsgBox strData
      bFlag = False
    Case HOME_SUCCESS:
      MsgBox "Home search was successful", 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 ('Unknown axis number.', 'Ajinextek', MB_OK);
        bFlag := False; end;
      HOME_ERR_GNT_RANGE: begin
        Application.MessageBox ('Out of Gantry Offset', 'Ajinextek', MB_OK);
        bFlag := False; end;
      HOME_ERR_USER_BREAK: begin
        Application.MessageBox ('Home search stops.', 'Ajinextek', MB_OK);
        bFlag := False; end;
    end;
  end;
end.

```

```
bFlag := False; end;
HOME_ERR_VELOCITY: begin
Application.MessageBox ('Velocity has not been set.', 'Ajinextek', MB_OK);
    bFlag := False; end;
HOME_ERR_AMPFAULT: begin
Application.MessageBox ('Servo pack alarm has occurred.', 'Ajinextek',
    MB_OK);
    bFlag := False; end;
HOME_ERR_NEG_LIMIT: begin
Application.MessageBox ('(-)limit sensor has been detected.', 'Ajinextek',
    MB_OK); bFlag := False; end;
HOME_ERR_POS_LIMIT: begin
Application.MessageBox ('(+)limit sensor has been detected', 'Ajinextek',
    MB_OK);
    bFlag := False; end;
HOME_SEARCHING: begin
{ Verify progress rate }
AxmHomeGetRate( 0, @ uHomeMainStepNumber ,@uHomeStepNumber );
nCount := uHomeStepNumber ; { // Verify home search rate on axis 0 }
strData := ' Home search is now in progress. The rate is ' +
    IntToStr(nCount) + '.';
Application.MessageBox (PCHAR(strData), 'Ajinextek', MB_OK);
bFlag := False; end;
HOME_SUCCESS: begin
Application.MessageBox ('Home search was successful', 'Ajinextek', MB_OK);
    bFlag := False; end;
end; end;

End;
```

See Also

[AxmHomeSetSignalLevel](#), [AxmHomeGetSignalLevel](#), [AxmHomeReadSignal](#), [AxmHomeSetMethod](#),
[AxmHomeGetMethod](#), [AxmHomeSetVel](#), [AxmHomeGetVel](#), [AxmHomeSetStart](#), [AxmHomeSetResult](#),
[AxmHomeGetResult](#), [AxmHomeGetRate](#), [AxmHomeGetResult](#),

Position Move API

In this chapter, we introduce APIs related with not Interpolation Control but Single Axis, Multi Axis motion control. Single axis motion means the work that independently controls only one axis, while multi axis motion means the work that controls more than one axis. Multi axes that are not interpolation control set velocity and acceleration, and perform moving tasks. Also, if necessary, they stop motions using stop API.

The APIs are the following:

Function	Description
AxmMoveStartPos	Moves at preset velocity and acceleration rate up to the position that was set as an absolute coordinate of a specific axis. Velocity profile is set in AxmMotSetProfileMode API. Exits API at the point of pulse out.
AxmMovePos	Moves at preset velocity and acceleration rate up to the position that was set as an absolute coordinate of a specific axis. Velocity profile is set in AxmMotSetProfileMode API. Exits API at the point that pulse out is exited.
AxmMoveVel	Constantly moves in velocity mode at the velocity and acceleration rate preset on a specific axis. Exits API at the point that pulse out starts.
AxmMoveStartMultiVel	Constantly moves in velocity mode at the velocity and acceleration rate preset on specific multi axes. Exits API at the point that pulse out starts.
AxmMoveSignalSearch	Moves until the signal set to a specific velocity and acceleration rate is detected. Exits API at the point of pulse out.
AxmMoveSignalCapture	Moves in order to detect a signal set on a spcific axis and store the position
AxmMoveSignalCapture	Verifies the position value stored in ' AxmMoveSignalCapture ' API.
AxmMoveStartMultiPos	Moves at preset velocity and acceleration rate up to the positions that was set as absolute coordinates of specific axes. Velocity profile is set in AxmMotSetProfileMode API. Exits API at the point that pulse out is exited.
AxmMoveMultiPos	Moves at preset velocity and acceleration rate up to the positions that was set as absolute coordinates of specific axes. Velocity profile is set in AxmMotSetProfileMode API.

	Exits API at the point of pulse out.
AxmMoveStop	Deceleration stop at the deceleration that was set on a specific axis (The unit of deceleration is PPS[Pulses/sec] w Unit/pulse is set to 1/1)
AxmMoveEStop	Emergency stop a specific axis
AxmMoveSStop	Deceleration stop a specific axis

- * Single axis move is generally the most fundamental move in motion move. Single axis drive is composed of position drive, velocity drive, signal search drive, and home search drive.
- * Velocity drive is the move such that it keeps moving based on preset velocity acceleration (The unit of acceleration is PPS[Pulses/sec] when Unit/pulse is set to 1/1) and acceleration time until exit command is called
- * Signal search drive is the move such that it initially moves at a uniform speed after presetting searching targets such as limit signal or universal input/output signal, and it stops when the preset signal is detected. Signal search drive consists of [AxmMoveSignalSearch](#) API which decelerates or emergency stops if a specific signal is detected and [AxmMoveSignalCapture](#) API which stores the position. There are 8 searchable signals as shown in the table below. Since each signal contains Up Edge and Down Edge, there are a total of 16 signals.
- * Generally in equipment move systems, there are many instances that either many axes have to be moved to a specific position at once or each axis has to move in relation with each other in some patterns. Multi axis move APIs are provided in accordance with these moves. Settings for multi axis move can be done by following the single axis move settings for all axes, as previously discussed. For example, if there are two axes, you can set the same settings as that of single axis move API to each of axis 0 and axis 1. Multi axis move APIs are divided into the following: multi axis position move that enables many axes to move at once to a specific position at a specific velocity, two axes interpolation move up to a specific position, continuous interpolation move that enables many axes to move according to specific paths.

AxmMoveStartPos

Purpose

Moves at preset velocity and acceleration rate up to a preset position of a specific axis.
Exits API at the point that pulse out starts.

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	–	Channel(axis) number (start from 0)
Pos	in	double	–	Moving distance
Vel	in	double		Velocity
Accel	in	double		Acceleration (The unit of acceleration is PPS[Pulses/sec^2] when Unit/pulse is set to 1/1)
Decel	in	double		Deceleration (The unit of deceleration is PPS[Pulses/sec^2] when Unit/pulse is set to 1/1)

Return AXT_RT_SUCCESS(0000) : Successful execution of API.

[* See error code Table for more information on status error codes](#)

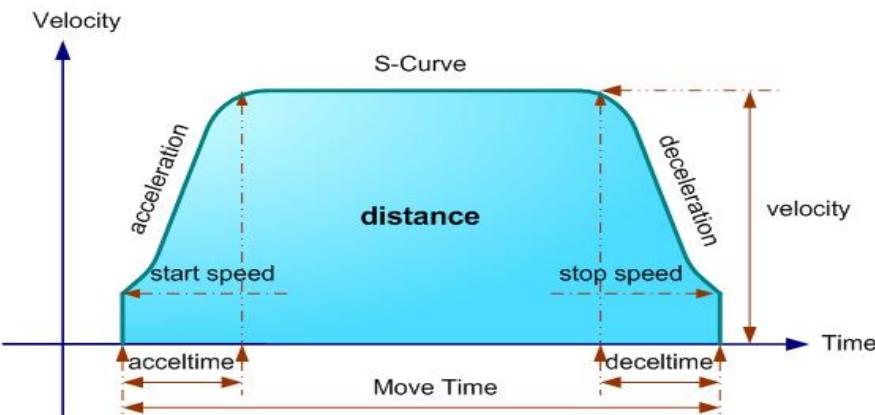
Description

It executes move by a specific distance (absolute/relative) from the current position of a specific axis. First choose absolute/relative using [AxmMotSetAbsRelMode](#) API and choose a profile using [AxmMotSetProfileMode](#) API. In absolute position mode, the setting position value becomes the absolute position value to move to, while in relative position mode, it becomes the moving distance. For example, in absolute position mode, if the position value has been set to ‘100’, it moves to the position ‘100’ and stop. In relative position mode, it moves from the current position by ‘100’.

- ▶ You can verify whether it is in motion using [AxmStatusReadInMotion](#) API.
- ▶ When using [AxmSignalSetInpos](#) API, if Inpos input signal is set to Enable, it won’t return until INP input becomes ON although command pulse out has been completed as it is regarded that motion has

not been completed. Please refer to [AxmSignalSetInposAPI](#) for details.

- In case of using master, slave by using [AxmGantrySetEnable](#), if a command is executed on the slave axis, the command will be ignored.

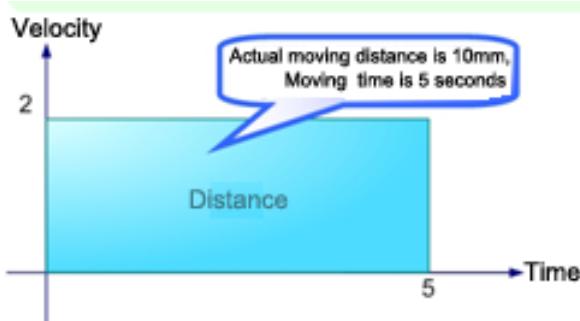


Understanding velocity

$$\text{Velocity} = \text{change in Distance} / \text{Change in Time}$$

$$\text{Distance} = \text{Velocity} \times \text{Time}$$

Velocity is the ratio of changes in distance to changes in time, so the unit of velocity uses mm/s or so. Note that mm is the moving distance in millimeters and s is the time in seconds. Therefore, assuming that the velocity is 2 mm/s, it means when time changes by 1 second, distance changes by 2mm.



Understanding acceleration (The unit of acceleration is PPS[Pulses/sec] when Unit/pulse is set to 1/1)

If acceleration is 0, it means no change in velocity. That is, it moves at a constant velocity. The unit of acceleration is mm/s². Note that mm is the moving distance in millimeters and s² is square of the time.

$$\text{Acceleration} = \text{Change in Velocity} / \text{Change in Velocity}$$

C Example

```
// Trapezoidal move by 2000 at velocity 100 and acceleration rate 200 on  
// the absolute coordinates of axis 0. Once the move starts, it exits API.  
AxmMoveStartPos (0, 2000, 100, 200, 200);
```

VB Example

```
'Trapezoidal move by 2000 at velocity 100 and acceleration rate 200 on the  
'absolute coordinates of axis 0. Once the move starts, it exits API.  
AxmMoveStartPos 0, 2000, 100, 200, 200
```

Delphi Example

```
begin  
{ Trapezoidal move by 2000 at velocity 100 and acceleration rate 200 on  
  the absolute coordinates of axis 0. Once the move starts, it exits  
  API. }  
AxmMoveStartPos (0, 2000, 100, 200, 200);  
End;
```

See Also

[AxmMovePos](#), [AxmMoveVel](#), [AxmMoveStartMultiVel](#), [AxmMoveSignalSearch](#), [AxmMoveSignalCapture](#),
[AxmMoveGetCapturePos](#), [AxmMoveStartMultiPos](#), [AxmMoveMultiPos](#), [AxmMoveStop](#),
[AxmMoveEStop](#), [AxmMoveSStop](#)

AxmMovePos

Purpose

Moves at preset velocity and acceleration rate up to a preset position of a specific axis.
Exits API at the point that pulse out is exited.

Format

C

```
DWORD AxmMovePos (long IAxisNo, double dPos, double dVel, double dAccel, double dDecel);
```

Visual Basic

```
Function AxmMovePos (ByVal IAxisNo As Long, ByVal dPos As Double, ByVal dVel As Double, ByVal dAccel As Double, ByVal dDecel As Double) As Long
```

Delphi

```
function AxmMovePos (IAxisNo : LongInt; dPos : Double; dVel : Double; dAccel : Double; dDecel : Double) : DWord;
  stdcall;
```

Input/ Output

Name	in/out	Format	Init Value	Explanation
AxisNo	in	long		Channel (axis) number(starting from 0)
Pos	in	double		Moving distance
Vel	in	double		Velocity
Accel	in	double		Acceleration (The unit of acceleration is PPS[Pulses/sec^2] when Unit/pulse is set to 1/1)
Decel	in	double		Deceleration (The unit of deceleration is PPS[Pulses/sec^2] when Unit/pulse is set to 1/1)

Return AXT_RT_SUCCESS(0000) : Successful execution of API.

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

Description

It executes move by a specific distance (absolute/relative) from the current position on a specific axis.

First choose absolute/relative using [AxmMotSetAbsRelMode](#) API and choose a profile using

[AxmMotSetProfileMode](#) API. It moves at preset velocity and acceleration up to a preset position of a specific axis. It exits API at the point that pulse out is exited. Since it will occupy CPU process until from the time API starts to the moment the move is exited, special caution may be required in its usage. You cannot verify whether it is in motion using [AxmStatusReadInMotion](#) API.

In absolute position mode, the setting position value becomes the absolute position value to move to, while in relative position mode, it becomes the moving distance.

For example, in absolute position mode, if the position value has been set to '100', it moves to the position '100' and stop. In relative position mode, it moves from the current position by '100'.

- ▶ When using [AxmSignalSetInpos](#)API, if Inpos input signal is set to Enable, it won't return until INP input becomes ON although command pulse out has been completed as it is regarded that motion has not been completed. Please refer to [AxmSignalSetInpos](#)API for details. **
- ▶ In case of using master, slave using by [AxmGantrySetEnable](#), if a command is executed on the slave axis, the command will be ignored.

C Example

```
// Trapezoidal move by 2000 at velocity 100 and acceleration rate 200 on
// the absolute coordinates of axis 0. It waits until the move exits,
// and then exits API.
AxmMovePos (0, 2000, 100, 200, 200);
```

VB Example

```
' Trapezoidal move by 2000 at velocity 100 and acceleration rate 200 on
' the absolute coordinates of axis 0. It waits until the move exits,
' and then exits API.
AxmMovePos 0, 2000, 100, 200, 200
```

Delphi Example

```
begin
{ Trapezoidal move by 2000 at velocity 100 and acceleration rate 200 on
  the absolute coordinates of axis 0. It waits until the move exits,
  and then exits API. }
AxmMovePos (0, 2000, 100, 200, 200);
End;
```

See Also

[AxmMoveStartPos](#), [AxmMoveVel](#), [AxmMoveStartMultiVel](#), [AxmMoveSignalSearch](#),
[AxmMoveSignalCapture](#), [AxmMoveGetCapturePos](#), [AxmMoveStartMultiPos](#), [AxmMoveMultiPos](#),
[AxmMoveStop](#), [AxmMoveEStop](#), [AxmMoveSStop](#)

AxmMoveVel

Purpose

Velocity mode move API that moves at a preset velocity on a specific axis.

Exits API at the point that pulse out starts.

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		Channel (axis) number(starting from 0)
Vel	in	double		Velocity(+direction: more than 0 , - direction: less than 0)
Accel	in	double		Acceleration (The unit of acceleration is PPS[Pulses/sec^2] when Unit/pulse is set to 1/1)
Decel	in	double		Deceleration (The unit of deceleration is PPS[Pulses/sec^2] when Unit/pulse is set to 1/1)

Return AXT_RT_SUCCESS(0000) : Successful execution of API.

[* See error code Table for more information on status error codes](#)

Description

It initially accelerates up to the working velocity and maintain it. It can be stopped by stop API calls or external sensor signals, and it continues the motion in the specific direction. You can verify whether it is in motion using [AxmStatusReadInMotionAPI](#).

- In case of using master, slave by using [AxmGantrySetEnable](#) , if a command is executed on the slave axis, the command will be ignored.

C Example

```
// Velocity drive at velocity 100 and acceleration rate 200 on the
// absolute coordinates of axis 0.
AxmMoveVel (0, 100, 200, 200);
```

VB Example

```
' Velocity drive at velocity 100 and acceleration rate 200 on the absolute
    coordinates of axis 0.
AxmMoveVel 0, 100, 200, 200
```

Delphi Example

```
begin
{Velocity drive at velocity 100 and acceleration rate 200 on the absolute
    coordinates of axis 0. }
AxmMoveVel (0, 100, 200, 200);
End;
```

See Also

[AxmMoveStartPos](#), [AxmMovePos](#), [AxmMoveStartMultiVel](#), [AxmMoveSignalSearch](#), [AxmMoveSignalCapture](#),
[AxmMoveGetCapturePos](#), [AxmMoveStartMultiPos](#), [AxmMoveMultiPos](#), [AxmMoveStop](#), [AxmMoveEStop](#),
[AxmMoveSStop](#)

AxmMoveStartMultiVel

Purpose

Velocity mode move API that moves at a preset velocity on specific multi axes.

Exits API at the point that pulse out starts.

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		Axis size
AxisNo	in	Long*		Channel(axis) number (start from 0) array
Vel	in	Double*		Velocity(+direction: more than 0 , – direction: less than 0) array
Accel	in	Double*		Acceleration array (The unit of acceleration is PPS[Pulses/sec^2] when Unit/pulse is set to 1/1)
Decel	in	Double*		Deceleration array (The unit of deceleration is PPS[Pulses/sec^2] when Unit/pulse is set to 1/1)

Return AXT_RT_SUCCESS(0000) : Successful execution of API.

[* See error code Table for more information on status error codes](#)

Description

Multi axis concurrent control is the function that can control multiple axis simultaneously with perfect sync. If velocity profiles have been set equally, many of the axes can be controlled with perfect synchronization of not only start/end points but also acceleration/deceleration intervals. It starts velocity move tasks of many axes simultaneously. Velocity move can accelerate up to the work velocity and maintain the work velocity. It can be stopped until a stop API is called or by external sensors; otherwise, it continues to execute motion in the specified direction. Using this API is very helpful when multiple axes start work simultaneously and synchronized with each other.

- ▶ You can verify whether it is in motion using [AxmStatusReadInMotionAPI](#).

C Example

```
// Velocity drive at velocity 100, acceleration rate 200 and with S-curve

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

```
' Velocity drive at velocity 100, acceleration rate 200 and with S-curve
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

For i = 0 To 1
    AxmMotSetProfileMode i, 4
    AxmMotSetAbsRelMode i, 0
    axes(i) = i
    velocities(i) = 100
    accelerations(i) = 200
    decelerations(i) = 200

Next i

AxmMoveStartMultiVel 2, axes(0), velocities(0), accelerations(0),
                     decelerations(0) 'Setting deceleration stop
```

Delphi Example

```
var
  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
{ Velocity drive at velocity 100, acceleration rate 200 and with S-curve
  profile on the axis 0 and axis 1. }

  for i := 0 to 1 do
  begin
    AxmMotSetProfileMode (i, 4);
    AxmMotSetAbsRelMode (i, 0);
    dVel[i] := 100;
    dAccel[i] := 200;
    dDecel[i] := 200;
    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

API that moves in order to detect the signal that was set on a specific axis.

Format

C

```
DWORD AxmMoveSignalSearch (long IAxisNo, double dVel, double dAccel, long IDetectSignal, long ISignalEdge,
                           long ISignalMethod);
```

Visual Basic

```
Function AxmMoveSignalSearch (ByVal IAxisNo As Long, ByVal dVel As Double, ByVal dAccel As Double, ByVal
                               IDetectSignal As Long, ByVal ISignalEdge As Long, ByVal ISignalMethod As Long) As Long
```

Delphi

```
function AxmMoveSignalSearch (IAxisNo : LongInt; dVel : Double; dAccel : Double; IDetectSignal : LongInt;
                             ISignalEdge : LongInt; ISignalMethod : LongInt) : DWord; stdcall;
```

Input

Name	in/out	Format	Init Value	Explanation
AxisNo	in	long	-	Channel(axis) number (start from 0) array
Vel	in	double		Velocity(+direction: more than 0 , – direction: less than 0) array
Accel	in	double		Acceleration (The unit of acceleration is PPS[Pulses/sec^2] when Unit/pulse is set to 1/1)
DetectSignal	in	long		Sets detection signal
SignalEdge	in	long		Sets Edge direction of detection signal
SignalMethod	In	long		Stop mode

IDetectSignal

#define	Value	Explanation
PosEndLimit	00h	+Elm(End limit) +direction limit sensor signal
NegEndLimit	01h	-Elm(End limit) – direction limit sensor signal
PosSloLimit	02h	+Slm(Deceleration limit) signal – Non-used
NegSloLimit	03h	-Slm(Deceleration limit) signal – Non-used
HomeSensor	04h	IN0(ORG) home sensor signal
EncodZPhase	05h	IN1(Z) Encoder Z phase signal
Unilnput02	06h	IN2(universal) universal input signal #2
Unilnput03	07h	IN3(universal) universal input signal #3

ISignalEdge

#define	Value	Explanation
SIGNAL_DOWN_EDGE	00h	Down Edge
SIGNAL_UP_EDGE	01h	Up Edge

ISignalMethod

#define	Value	Explanation
EMERGENCY_STOP	00h	Emergency stop
SLOWDOWN_STOP	01h	Slowdown stop

Return AXT_RT_SUCCESS(0000) : Successful execution of API.

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

Description

It is the API that moves in order to search the signal that was set on a specific axis.

It can detect limit sensor signal, home sensor signal, Z phase signal of servo driver or Edges of universal input signals.

It is mainly used to implement home search move APIs.

Caution: In case of using QI, if you search IDetectSignal0| PosEndLimit , NegEndLimit(0,1), it will detect the signal's level active state.

- ▶ You can verify whether it is in motion using [AxmStatusReadInMotionAPI](#).
 - ▶ When using [AxmSignalSetInposAPI](#), if Inpos input signal is set to Enable, it won't return until INP input becomes ON although command pulse out has been completed as it is regarded that motion has not been completed. Please refer to [AxmSignalSetInposAPI](#) for details. **
- Caution: If SignalMethod is used as EMERGENCY_STOP(0), the acceleration/deceleration are ignored and it accelerates at the specified velocity and emergency stops.

C Example

```
// If Z phase signal on axis 0 is encountered, it detects Falling Edge as
// it deceleration stops.
AxmMoveSignalSearch (0, 100, 200, 5, 0, 0);
```

VB Example

```
' If Z phase signal on axis 0 is encountered, it detects Falling Edge as
// it deceleration stops.
AxmMoveSignalSearch 0, 100, 200, 5, 0, 0
```

Delphi Example

```
begin
{ If Z phase signal on axis 0 is encountered, it detects Falling Edge as
// it deceleration stops.}
AxmMoveSignalSearch (0, 100, 200, 5, 0, 0);
End;
```

See Also

[AxmMoveStartPos](#), [AxmMovePos](#), [AxmMoveVel](#), [AxmMoveStartMultiVel](#), [AxmMoveSignalCapture](#),
[AxmMoveGetCapturePos](#), [AxmMoveStartMultiPos](#), [AxmMoveMultiPos](#), [AxmMoveStop](#),
[AxmMoveEStop](#), [AxmMoveSStop](#)

AxmMoveSignalCapture

Purpose

API that detects signals that were set on a specific axis and moves to store the positions.

Format

C

```
DWORD AxmMoveSignalCapture (long IAxisNo, double dVel, double dAccel, long IDetectSignal, long ISignalEdge,
                           long ITTarget, long ISignalMethod);
```

Visual Basic

```
Function AxmMoveSignalCapture (ByVal IAxisNo As Long, ByVal dVel As Double, ByVal dAccel As Double, ByVal
                               IDetectSignal As Long, ByVal ISignalEdge As Long, ByVal ITTarget As Long, ByVal ISignalMethod As Long) As Long
```

Delphi

```
function AxmMoveSignalCapture (IAxisNo : LongInt; dVel : Double; dAccel : Double; IDetectSignal : LongInt;
                               ISignalEdge : LongInt; ITTarget : LongInt; ISignalMethod : LongInt) : DWord; stdcall;
```

Input/ Output

Name	in/out	Format	Init Value	Explanation
AxisNo	in	long	-	Channel(axis) number (start from 0) array
Vel	in	double		Velocity(+direction: more than 0 , – direction: less than 0) array
Accel	in	double		Acceleration (The unit of acceleration is PPS[Pulses/sec^2] when Unit/pulse is set to 1/1)
DetectSignal	in	long		Sets detection signal
SignalEdge	in	long		Sets Edge direction of detection signal
SignalMethod	In	long		Stop mode
AxisNo	in	long	-	Channel(axis) number (start from 0) array

IDetectSignal

#define	Value	Explanation
PosEndLimit	00h	+Elm(End limit) +direction limit sensor signal
NegEndLimit	01h	-Elm(End limit) – direction limit sensor signal
PosSloLimit	02h	+Slm(Deceleration limit) signal – Non-used
NegSloLimit	03h	-Slm(Deceleration limit) signal – Non-used
HomeSensor	04h	IN0(ORG) home sensor signal
EncodZPhase	05h	IN1(Z) Encoder Z phase signal
Unilnput02	06h	IN2(universal) universal input signal #2
Unilnput03	07h	IN3(universal) universal input signal #3

ISignalEdge

#define	Value	Explanation
SIGNAL_DOWN_EDGE	00h	Down Edge
SIGNAL_UP_EDGE	01h	Up Edge

ITarget

#define	Value	Explanation
COMMAND	00h	Target position (Stores Cmd position)
ACTUAL	01h	Actual position (ACT (1) : Stores Encoder position)

SignalMethod

#define	Value	Explanation
EMERGENCY_STOP	00h	Emergency stop
SLOWDOWN_STOP	01h	Slowdown stop

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

Description

It is the API that detects the signal that was set on a specific axis and moves to store the position. It can detect limit sensor signal, home sensor signal, Z phase signal of servo driver or Edges of universal input signals.

In signal search, Cmd position or Encoder position can be stored.

The stored position can be verified using '[AxmMoveGetCapturePos](#)' API.

Caution: In case of using QI, if you search IDetectSignal01 PosEndLimit , NegEndLimit(0,1), it will detect the signal's level active state.

- ▶ You can verify whether it is in motion using [AxmStatusReadInMotion](#)API.
- ▶ When using [AxmSignalSetInpos](#)API, if Inpos input signal is set to Enable, it won't return until INP input becomes ON although command pulse out has been completed as it is regarded that motion has not been completed. Please refer to [AxmSignalSetInpos](#)API for details. **

Caution: If SignalMethod is used as EMERGENCY_STOP(0), the acceleration/deceleration is ignored and it accelerates at the specified velocity and emergency stops.

C Example

```
// When Z phase signal Falling Edge on axis 0 is detected, it stores
// Encoder position as it deceleration stops.
AxmMoveSignalCapture (0, 100, 200, 5, 0, ACTUAL, 0);
```

VB Example

```
' When Z phase signal Falling Edge on axis 0 is detected, it stores
// Encoder position as it deceleration stops.
```

```
AxmMoveSignalCapture 0, 100, 200, 5, 0, ACTUAL, 0
```

Delphi Example

```
begin
{ When Z phase signal Falling Edge on axis 0 is detected, it stores
  Encoder position as it deceleration stops. }
AxmMoveSignalCapture (0, 100, 200, 5, 0, ACTUAL, 0);
End;
```

See Also

[AxmMoveStartPos](#), [AxmMovePos](#), [AxmMoveVel](#), [AxmMoveStartMultiVel](#), [AxmMoveSignalSearch](#),
[AxmMoveGetCapturePos](#), [AxmMoveStartMultiPos](#), [AxmMoveMultiPos](#), [AxmMoveStop](#),
[AxmMoveEStop](#), [AxmMoveSStop](#)

AxmMoveGetCapturePos

Purpose

API that verifies the position values stored in ‘AxmMoveSignalCapture’ API.

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; dpCapPosition : PDouble) : DWord; stdcall;
```

Input / Output

Name	in/out	Format	Init Value	Explanation
AxisNo	in	long		Channel (axis) number(starting from 0)
dpCapPos	out	double		Position value in signal search

Return AXT_RT_SUCCESS(0000) : Successful execution of API.

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

Description

It is the API that verifies the position values stored in ‘[AxmMoveSignalCapture](#)’ API.

- ▶ You can verify whether it is in motion using [AxmStatusReadInMotion](#) API.
- ▶ When using [AxmSignalSetInpos](#) API, if Inpos input signal is set to Enable, it won’t return until INP input becomes ON although command pulse out has been completed as it is regarded that motion has not been completed. Please refer to [AxmSignalSetInpos](#) API for details. **

C Example

```
// Verify position during Z phase detection.
double dPos;
AxmMoveGetCapturePos (0, &dPos);
```

VB Example

```
' Verify position during Z phase detection.
Dim dPos As Double
AxmMoveGetCapturePos 0, dPos
```

Delphi Example

```
{ Verify position during Z phase detection. }
var
dPos : Double;
begin
```

```
AxmMoveGetCapturePos ( 0 , @dPos );
end;
```

See Also

[AxmMoveStartPos](#), [AxmMovePos](#), [AxmMoveVel](#), [AxmMoveStartMultiVel](#), [AxmMoveSignalSearch](#),
[AxmMoveSignalCapture](#), [AxmMoveStartMultiPos](#), [AxmMoveMultiPos](#), [AxmMoveStop](#),
[AxmMoveEStop](#), [AxmMoveSStop](#)

AxmMoveStartMultiPos

Purpose

Starts move from the current position by a specific distance about multiple axes. If this API is used, multiple axes start working simultaneously. This API is used when multiple axes need to start working with synchronization. Exits API at the point that pulse out starts.

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		Axis size
AxisNo	in	long*		Channel(axis) number (start from 0) array
Pos	in	double*		Moving distance array
Vel	in	double*		Velocity array
Accel	in	double*		Acceleration (The unit of acceleration is PPS[Pulses/sec^2] when Unit/pulse is set to 1/1) array
Decel	in	double *		Deceleration (The unit of deceleration is PPS[Pulses/sec^2] when Unit/pulse is set to 1/1) array

Return AXT_RT_SUCCESS(0000) : Successful execution of API.

[* See error code Table for more information on status error codes](#)

Description

It executes move by a specific distance (absolute/relative) from the current position of one axis. First choose absolute/relative using [AxmMotSetAbsRelMode](#) API and choose a profile using [AxmMotSetProfileMode](#) API. You can verify whether it is in motion using [AxmStatusReadInMotion](#) API.

In absolute position mode, the setting position value becomes the absolute position value to move to, while in relative position mode, it becomes the moving distance. For example, if the position value has been set to '100', it moves to the position '100' and stop in absolute position mode, whereas in relative position mode, it moves from the current position by '100'.

Multi axis concurrent control is the function that can control multiple axis simultaneously with perfect

sync. If velocity profiles have been set equally, many of the axes can be controlled with perfect synchronization of not only start/end points but also acceleration/deceleration intervals.

- ▶ You can verify whether it is in motion using [AxmStatusReadInMotion](#) API.
- ▶ When using [AxmSignalSetInpos](#) API, if Inpos input signal is set to Enable, it won't return until INP input becomes ON although command pulse out has been completed as it is regarded that motion has not been completed. Please refer to [AxmSignalSetInpos](#) API for details. **

C Example

```
// Trapezoidal move by 2000 at velocity 100 and acceleration rate 200 on
// the absolute coordinates. It exits API if the move starts.
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

```
' Trapezoidal move by 2000 at velocity 100 and acceleration rate 200 on
' the absolute coordinates. It exits API if the move starts.
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

```
{ Trapezoidal move by 2000 at velocity 100 and acceleration rate 200 on
    the absolute coordinates. It exits API if the move starts. }
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

Starts move from the current position by a specific distance about multiple axes. If this API is used, multiple axes start working simultaneously. This API is used when multiple axes need to start working with synchronization. Exits API at the point that pulse out is exited

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		Axis size
AxisNo	in	long*	-	Channel(axis) number (start from 0) array
Pos	in	double*		Moving distance array
Vel	in	double*		Velocity array
Accel	in	double*		Acceleration (The unit of acceleration is PPS[Pulses/sec^2] when Unit/pulse is set to 1/1) array
Decel	in	double *		Deceleration (The unit of deceleration is PPS[Pulses/sec^2] when Unit/pulse is set to 1/1) array

Return AXT_RT_SUCCESS(0000) : Successful execution of API.

[* See error code Table for more information on status error codes](#)

Description

It executes move by a specific distance (absolute/relative) from the current position on multiple axes.

First choose absolute/relative using [AxmMotSetAbsRelMode](#) API and choose a profile using [AxmMotSetProfileMode](#) API.

It moves at preset velocity and acceleration up to a preset position of specific axes. It exits API at the point that pulse out is exited. Since it will occupy CPU process until from the time API starts to the moment the move is exited, special caution may be required in its usage. You cannot verify whether it is in motion using [AxmStatusReadInMotion](#) API.

In absolute position mode, the setting position value becomes the absolute position value to move to,

while in relative position mode, it becomes the moving distance.

For example, in absolute position mode, if the position value has been set to ‘100’, it moves to the position ‘100’ and stop. In relative position mode, it moves from the current position by ‘100’.

- ▶ You can verify whether it is in motion using [AxmStatusReadInMotionAPI](#).
- ▶ When using [AxmSignalSetInposAPI](#), if Inpos input signal is set to Enable, it won’t return until INP input becomes ON although command pulse out has been completed as it is regarded that motion has not been completed. Please refer to [AxmSignalSetInposAPI](#) for details. **

C Example

```
// Trapezoidal move by 2000 at velocity 100, acceleration rate 200 on the
// absolute coordinates. It exits API if the move starts.

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;
}
AxmMoveMultiPos (2, lAxis, dPos, dVel, dAccel, dAccel);
```

VB Example

```
' Trapezoidal move by 2000 at velocity 100, acceleration rate 200 on the
' absolute coordinates. It exits API if the move starts.

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; 'Set stop mode slowdown stop
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

AxmMoveMultiPos 2 , axis(0), Distance(0), velocities(0), accelerations(0),
                accelerations(0)
```

Delphi Example

```
{ Trapezoidal move by 2000 at velocity 100, acceleration rate 200 on the
    absolute coordinates. It exits API if the move starts. }
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; { Set stop mode slowdown stop}
  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;
  AxmMoveMultiPos (2, @lAxis, @dPos, @dVel, @ dAccel, @dAccel);
end;
```

See Also

[AxmMoveStartPos](#), [AxmMovePos](#), [AxmMoveVel](#), [AxmMoveStartMultiVel](#), [AxmMoveSignalSearch](#),
[AxmMoveSignalCapture](#), [AxmMoveGetCapturePos](#), [AxmMoveStartMultiPos](#), [AxmMoveStop](#),
[AxmMoveEStop](#), [AxmMoveSStop](#)

AxmMoveStop

Purpose

API that enables user to stop the current motion in move at arbitrary deceleration rate.

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		Channel (axis) number(starting from 0)
Decel	in	double		Deceleration rate when stopped

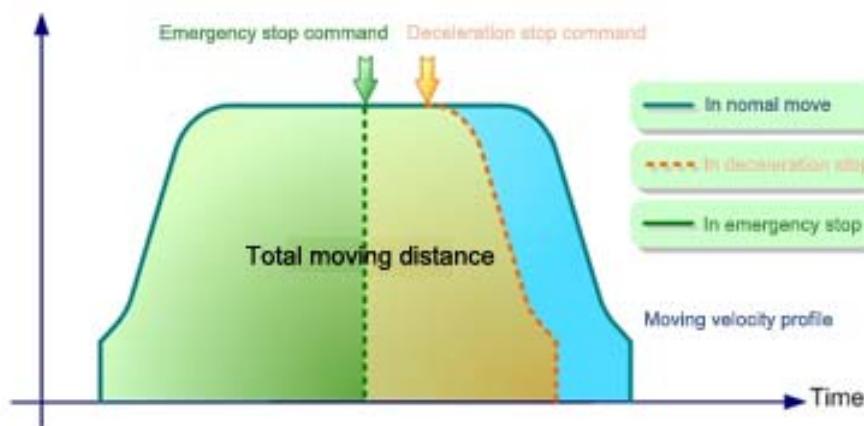
Return AXT_RT_SUCCESS(0000) : Successful execution of API.

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

Description

It is the API that stops the current motion in move at arbitrary deceleration rate.

Velocity



C Example

```
// Emergency stop motion move of axis 0
AxmMoveStop(0, 0);
```

VB Example

```
' Emergency stop motion move of axis 0
AxmMoveStop 0, 0
```



Delphi Example

```
begin
{ Emergency stop motion move of axis 0 }
AxmMoveStop(0, 0);
End;
```

See Also

[AxmMoveStartPos](#), [AxmMovePos](#), [AxmMoveVel](#), [AxmMoveStartMultiVel](#), [AxmMoveSignalSearch](#),
[AxmMoveSignalCapture](#), [AxmMoveGetCapturePos](#), [AxmMoveStartMultiPos](#),
[AxmMoveMultiPos](#), [AxmMoveEStop](#), [AxmMoveSStop](#)

AxmMoveEStop

Purpose

API that emergency stops the current motion in move.

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

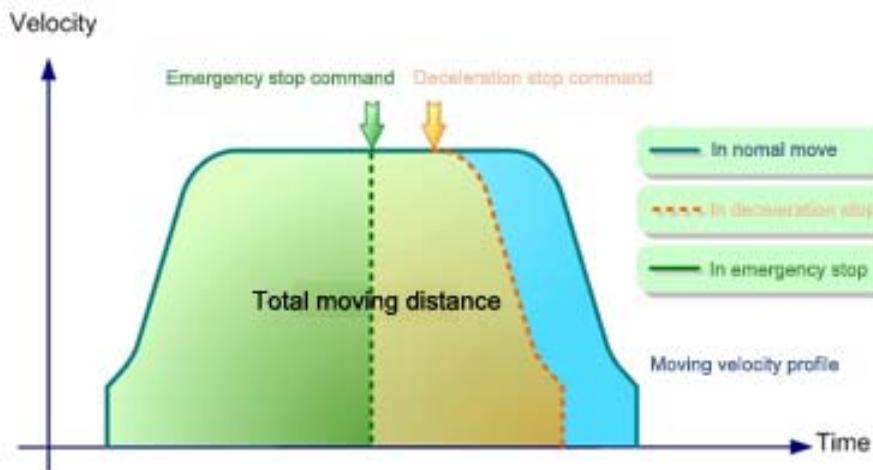
Name	in/out	Format	Init Value	Explanation
AxisNo	in	long	-	Channel (axis) number(starting from 0)

Return AXT_RT_SUCCESS(0000) : Successful execution of API.

[* See error code Table for more information on status error codes](#)

Description

It is the API that emergency stops the current motion in move without deceleration.



C Example

```
// Emergency stop motion move of axis 0
AxmMoveEStop(0);
```

VB Example

```
' Emergency stop motion move of axis 0  
AxmMoveEStop 0
```

Delphi Example

```
begin  
{ Emergency stop motion move of axis 0 }  
AxmMoveEStop(0);  
End;
```

See Also

[AxmMoveStartPos](#), [AxmMovePos](#), [AxmMoveVel](#), [AxmMoveStartMultiVel](#), [AxmMoveSignalSearch](#),
[AxmMoveSignalCapture](#), [AxmMoveGetCapturePos](#), [AxmMoveStartMultiPos](#),
[AxmMoveMultiPos](#), [AxmMoveStop](#), [AxmMoveSStop](#)

AxmMoveSStop

Purpose

API that deceleration stops the current motion in move.

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		Channel (axis) number(starting from 0)

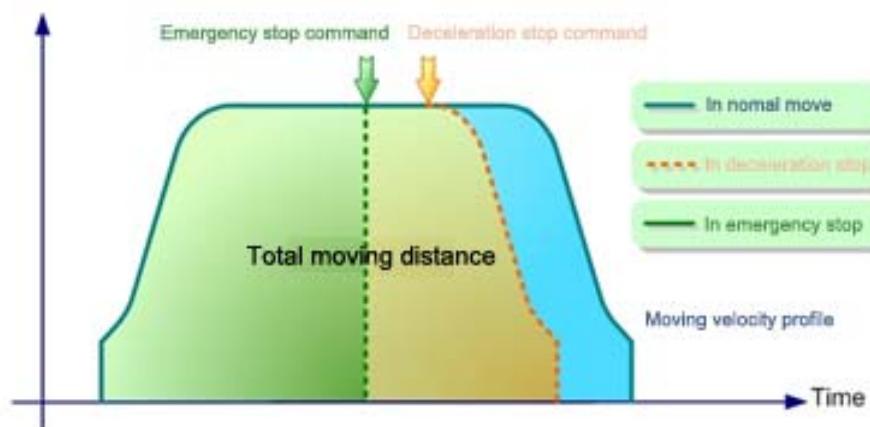
Return AXT_RT_SUCCESS(0000) : Successful execution of API.

[* See error code Table for more information on status error codes](#)

Description

It is the API that decelerates and stops the current motion in move by the amount of the acceleration that user initially input.

Velocity



C Example

```
// Deceleration stop motion move of axis 0
AxmMoveSStop(0);
```

VB Example

```
'Deceleration stop motion move of axis 0  
AxmMoveSStop 0
```

Delphi Example

```
begin  
{ Deceleration stop motion move of axis 0 }  
AxmMoveSStop(0);  
End;
```

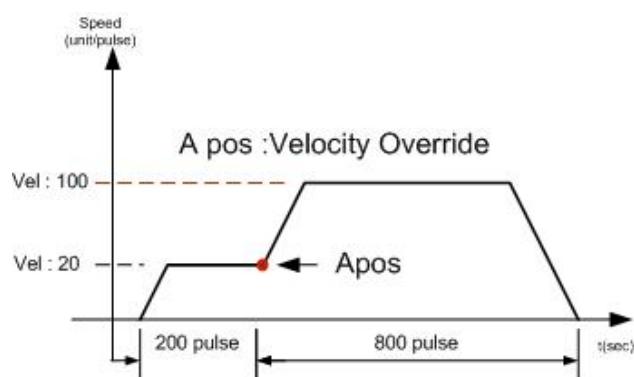
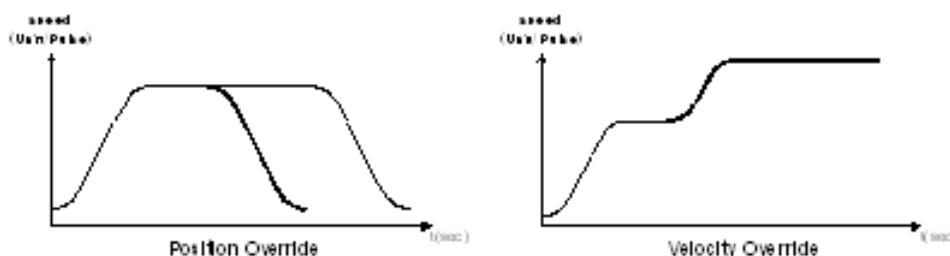
See Also

[AxmMoveStartPos](#), [AxmMovePos](#), [AxmMoveVel](#), [AxmMoveStartMultiVel](#), [AxmMoveSignalSearch](#),
[AxmMoveSignalCapture](#), [AxmMoveGetCapturePos](#), [AxmMoveStartMultiPos](#), [AxmMoveMultiPos](#),
[AxmMoveStop](#), [AxmMoveEStop](#)

Position / Velocity Override Move API

In this chapter, we introduce velocity override move and position override move. Velocity override move means changing working velocity while motion is in progress, and position override drive means changing command distance while motion is in progress

Function	Description.
AxmOverridePos	Adjusts the number of assigned output pulses before the move of a specific axis is exited.
AxmOverrideSetMaxVel	Sets the maximum velocity to override before the velocity of a specific axis overrides.
AxmOverrideVel	Sets the velocity variably during the move of a specific axis.
AxmOverrideVelAtPos	Constantly moves in velocity mode at the preset velocity and acceleration rate for a specific axis. During the move, it executes velocity override at a designated position. Exits API at the point that pulse out starts.



AxmOverridePos

Purpose

API that changes preset moving distance during the motion of a specific axis.

Format

C

```
DWORD AxmOverridePos (long IAxisNo, double dOverridePos);
```

Visual Basic

```
Function AxmOverridePos (ByVal IAxisNo As Long, ByVal dOverridePos As Double) As Long
```

Delphi

```
function AxmOverridePos (IAxisNo : LongInt; dOverridePos : Double) : DWord; stdcall;
```

Input / Output

Name	in/out	Format	Init Value	Explanation
AxisNo	in	long		Channel (axis) number(starting from 0)
OverridePos	In	double		Moving distance to change

Return AXT_RT_SUCCESS(0000) : Successful execution of API.

[* See error code Table for more information on status error codes](#)

Description

Caution: When inputting a position to override, the position must be entered as a position value in relative form because the moving distance in the moving direction is inputted as a position value in relative form.

Position override means modifying the command distance or the command coordinates while in execution with InPosition being Enable. In order to use this API, you need to use a single axis motion API such as [AxmMoveStartPos](#) while in motion.

- ▶ The API can be used while in motion acceleration or in constant move. Even if the command is ordered after the motion's deceleration stops, it will be ignored.
Therefore, if user's override command is ignored, you have to acknowledge that the motion has already been completed.
- ▶ In case that it executes linear, circular, spline, helical interpolation tasks, override cannot be used.

Note1: the position value to override is the distance value relative to current move direction; thus, position should be relative position value.

Note2: in case of SMC-2V03, it should be less than min. move distance; when move distance is over than override distance, there is emergency stop.

Note3: in case of PCI-N804/404, it should be less than initial move distance; when move distance is over than override distance, it switches to move direction opposite to override position after deceleration stop. If override occurs again, ignore it.

C Example

```
AxmMotSetProfileMode (0, 4);      // Asymmetric S Curve
AxmMotSetAbsRelMode (0, 0);
AxmMoveStartPos (0, 1000, 200, 800, 800);

// Override to the position 2000 while in move of axis.
AxmOverridePos (0, 2000);
```

VB Example

```
AxmMotSetProfileMode0, 4 'Asymmetric S Curve
AxmMotSetAbsRelMode 0, 0
AxmMoveStartPos 0, 1000, 200, 800, 800

'Override to the position 2000 while in move of axis.
AxmOverridePos 0, 2000
```

Delphi Example

```
begin
{ 0 Override to the position 2000 while in move of axis. }

AxmMotSetProfileMode (0, 4); { Asymmetric S Curve }
AxmMotSetAbsRelMode (0, 0);
AxmMoveStartPos (0, 1000, 200, 800, 800);

AxmOverridePos (0, 2000);
End;
```

See Also

[AxmOverrideSetMaxVel](#), [AxmOverrideVel](#), [AxmOverrideVelAtPos](#), [AxmOverrideAccelVelDecel](#)

AxmOverrideSetMaxVel

Purpose

Set a maximum velocity to override before overriding the velocity of a specific axis. When it is necessary to override at a higher velocity than the current velocity to move at, it sets an anticipated maximum velocity before the move. If it will override only at a lower velocity than the velocity to move at, this API doesn't have to be used.

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		Channel (axis) number(starting from 0)
OverrideMaxVel	In	double		Maximum velocity to override

Return AXT_RT_SUCCESS(0000) : Successful execution of API.

[* See error code Table for more information on status error codes](#)

Description

It is used when it is necessary to override the velocity while a single axis motion is in progress.

In order to use this API, you need to use a single axis motion API such as [AxmMoveStartPos](#) while not in motion.

Caution : If it executes velocity override five times, a maximum velocity must be set among them.

C Example

```
AxmOverrideSetMaxVel(0, 500); // Set the maximum value of override

AxmMotSetProfileMode(0, 4); // Asymmetric S Curve
AxmMotSetAbsRelMode(0, 0);
AxmMoveStartPos(0, 1000, 200, 800, 800);

// Override the velocity to 500 while in move of axis.
AxmOverrideVel (0, 500);
```

VB Example

```
AxmOverrideSetMaxVel 0, 500 ' Set the maximum value of override

AxmMotSetProfileMode 0, 4 ' Asymmetric S Curve
AxmMotSetAbsRelMode 0, 0
AxmMoveStartPos 0, 1000, 200, 800, 800
```

```
'Override the velocity to 500 while in move of axis.  
AxmOverrideVel 0, 500
```

Delphi Example

```
Begin  
  AxmOverrideSetMaxVel(0, 500); { Set the maximum value of override 정}  
  AxmMotSetProfileMode(0, 4); { Asymmetric S Curve }  
  AxmMotSetAbsRelMode(0, 0);  
  AxmMoveStartPos(0, 1000, 200, 800, 800);  
  
  { Override the velocity to 500 while in move of axis.. }  
  AxmOverrideVel (0, 500);  
End;
```

See Also

[AxmOverrideVel](#), [AxmOverridePos](#), [AxmOverrideVelAtPos](#), [AxmOverrideAccelVelDecel](#)

AxmOverrideVel

Purpose

API that changes moving velocity while in motion of a specific axis.

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		Channel (axis) number(starting from 0)
OverrideVelocity	In	double		Moving velocity to change[Unit/Sec]

Return AXT_RT_SUCCESS(0000) : Successful execution of API.

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

Description

It is used when it is necessary to override the velocity while a single axis motion is in progress.

In order to use this API, you need to use a single axis motion API such as [AxmMoveStartPos](#)while not in motion.

Caution: Before using AxmOverrideVel API, use [AxmOverrideSetMaxVel](#)to set the maximum velocity that is available.

EX> Assuming that velocity override is executed twice,

1. Set the higher velocity from the above two as the maximum velocity using [AxmOverrideSetMaxVel](#).

2. Set variably the velocity of the designated axis in move where [AxmMoveStartPos](#)is appointed to be used, as the first velocity using AxmOverrideVel.

3. Set variably the velocity of the designated axis in move as the second velocity using AxmOverrideVel.

► The API can be used while in motion acceleration or in constant move. Even if the command is ordered after the motion's deceleration stops, it will be ignored.

Therefore, if user's override command is ignored, you have to acknowledge that the motion has already been completed.

► In case that it executes linear, circular, spline, helical interpolation tasks, override cannot be used..

C Example

```
AxmOverrideSetMaxVel (0, 500); // Sets a maximum value of override
AxmMotSetProfileMode (0, 4); // Asymmetric S Curve
AxmMotSetAbsRelMode (0, 0);
AxmMoveStartPos (0, 1000, 200, 800, 800);
```

```
// Override the velocity to 500 while in move of axis.  
AxmOverrideVel (0, 500);
```

VB Example

```
AxmOverrideSetMaxVel 0, 500 ' Sets a maximum value of override  
  
AxmMotSetProfileMode 0, 4 ' Asymmetric S Curve  
AxmMotSetAbsRelMode 0, 0  
AxmMoveStartPos 0, 1000, 200, 800, 800  
  
'Override the velocity to 500 while in move of axis.  
AxmOverrideVel 0, 500
```

Delphi Example

```
Begin  
  
AxmOverrideSetMaxVel (0, 500); { Sets a maximum value of override }  
  
AxmMotSetProfileMode (0, 4); { Asymmetric S Curve}  
AxmMotSetAbsRelMode (0, 0);  
AxmMoveStartPos (0, 1000, 200, 800, 800);  
  
{ Override the velocity to 500 while in move of axis. }  
AxmOverrideVel (0, 500);  
End;
```

See Also

[AxmOverrideSetMaxVel](#), [AxmOverridePos](#), [AxmOverrideVelAtPos](#), [AxmOverrideAccelVelDecel](#)

AxmOverrideAccelVelDecel

Purpose

API that changes moving velocity and acceleration and deceleration while in motion of a specific axis.

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		Channel (axis) number(starting from 0)
OverrideVelocity	In	double		Moving velocity to change [Unit/Sec]
dMaxAccel	In	double		Moving accel to change [Unit/Sec^2]
dMaxDecel	In	double		Moving decel to change [Unit/Sec^2]

Return AXT_RT_SUCCESS(0000) : Successful execution of API.

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

Description

It is used when it is necessary to override the velocity while a single axis motion is in progress.

In order to use this API, you need to use a single axis motion API such as [AxmMoveStartPos](#) not in motion.

Caution: Before using AxmOverrideAccelVelDecel API, use [AxmOverrideSetMaxVel](#) to set the maximum velocity that is available.

EX> Assuming that velocity override is executed twice,

1. Set the higher velocity from the above two as the maximum velocity using [AxmOverrideSetMaxVel](#).
2. Set variably the velocity of the designated axis in move where [AxmMoveStartPos](#) is appointed to be used, as the first velocity using AxmOverrideAccelVelDecel.
3. Set variably the velocity of the designated axis in move as the second velocity using AxmOverrideAccelVelDecel.

► The API can be used while in motion acceleration or in constant move. Even if the command is ordered after the motion's deceleration stops, it will be ignored.

Therefore, if user's override command is ignored, you have to acknowledge that the motion has already been completed.

- ▶ In case that it executes linear, circular, spline, helical interpolation tasks, override cannot be used.

C Example

```
AxmOverrideSetMaxVel (0, 500); // Sets a maximum value of override

AxmMotSetProfileMode (0, 4); // Asymmetric S Curve
AxmMotSetAbsRelMode (0, 0);
AxmMoveStartPos (0, 1000, 200, 800, 800);

// Override the velocity to 500 while in move of axis.
AxmOverrideAccelVelDecel (0, 500, 4000, 4000);
```

VB Example

```
AxmOverrideSetMaxVel0, 500 ' Sets a maximum value of override

AxmMotSetProfileMode0, 4 ' Asymmetric S Curve
AxmMotSetAbsRelMode 0, 0
AxmMoveStartPos0, 1000, 200, 800, 800

'Override the velocity to 500 while in move of axis.
AxmOverrideAccelVelDecel 0, 500, 4000, 4000
```

Delphi Example

```
Begin

AxmOverrideSetMaxVel (0, 500); { Sets a maximum value of override }

AxmMotSetProfileMode (0, 4); { Asymmetric S Curve}
AxmMotSetAbsRelMode (0, 0);
AxmMoveStartPos (0, 1000, 200, 800, 800);

{ Override the velocity to 500 while in move of axis. }
AxmOverrideAccelVelDecel (0, 500, 4000, 4000);
End;
```

See Also

[AxmOverrideSetMaxVel](#) , [AxmOverridePos](#) , [AxmOverrideVelAtPos](#)

AxmOverrideVelAtPos

Purpose

API that changes moving velocity at the position user has specified while in motion of a specific axis.

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 (IAxisNo : 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		Channel (axis) number(starting from 0)
Pos	in	double		Moving distance
Vel	in	double		Moving velocity(+direction: more than 0 , -direction: less than 0)
Accel	in	double		Acceleration (The unit of acceleration is PPS[Pulses/sec^2] when Unit/pulse is set to 1/1)
Decel	in	double		Deceleration (The unit of deceleration is PPS[Pulses/sec^2] when Unit/pulse is set to 1/1)
OverridePos	in	double		Position where velocity will be changed
OverrideVelocity	in	double		Moving velocity to change
Target	In	long		Choose position source where velocity will be changed

Target

#define	Value	Explanation
COMMAND	00h	Command position(Cmd position)
ACTUAL	01h	Actual position (Encoder position)

Return AXT_RT_SUCCESS(0000) : Successful execution of API.

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

Description

While moving to a designated position, it changes the velocity to the preset velocity at the position specified by user. This API is used when it needs to move either at high speed or at low speed.

Note: Before using AxmOverrideVelAtPos API, set the max. velocity that can be set by [AxmOverrideSetMaxVel](#).

- ▶ You can verify whether it is in motion using [AxmStatusReadInMotion](#)API.
- ▶ When using [AxmSignalSetInpos](#)API, if Inpos input signal is set to Enable, it won't return until INP input becomes ON although command pulse out has been completed as it is regarded that motion has not been completed. Please refer to [AxmSignalSetInpos](#)API for details. **

C Example

```
// Move by 50000 at velocity 100 and acceleration rate 200 on the axis 0.  
Velocity 500 override at the position 10000.  
AxmOverrideVelAtPos (0, 50000, 100, 200, 200, 10000, 500, ACTUAL);
```

VB Example

```
'Move by 50000 at velocity 100 and acceleration rate 200 on the axis 0.  
AxmOverrideVelAtPos 0, 50000, 100, 200, 200, 10000, 500, ACTUAL
```

Delphi Example

```
begin  
{ Move by 50000 at velocity 100 and acceleration rate 200 on the axis 0. }  
AxmOverrideVelAtPos (0, 50000, 100, 200, 200, 10000, 500, ACTUAL);  
End;
```

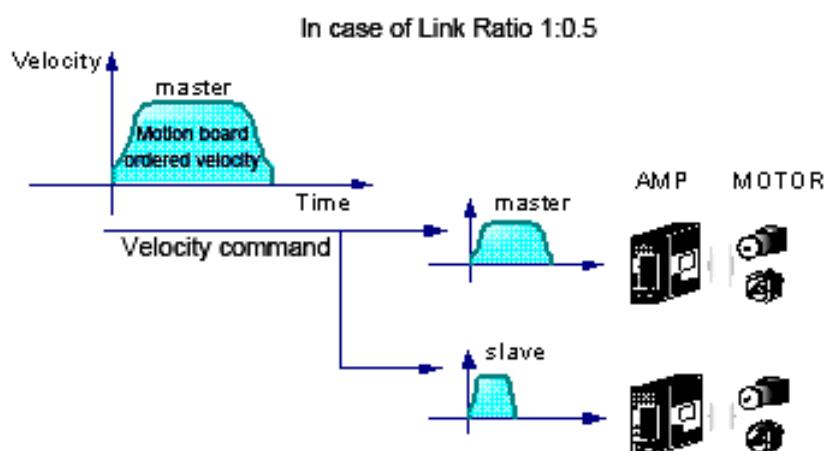
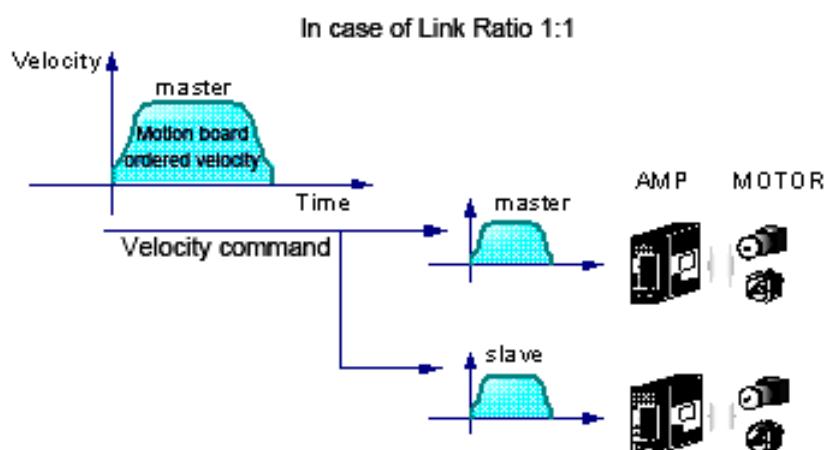
See Also

[AxmOverrideSetMaxVel](#) , [AxmOverridePos](#), [AxmOverrideVel](#), [AxmOverrideAccelVelDecel](#)

MASTER, SLAVE Move API

Function	Description
AxmLinkSetMode	Sets electric gear ratio between Master axis and Slave axis.
AxmLinkGetMode	Returns electric gear ratio between Master axis and Slave axis.
AxmLinkResetMode	Clear setting of electric gear ratio between Master axis and Slave axis.

Sync move, also named as electric gear mode, is the method in which independent two axes are controlled just as one axis. If user sets master axis, slave axis and the move ratio between them, the slave moves together as the master axis moves.



AxmLinkSetMode

Purpose

Set electric gear ratio between Master axis and Slave axis.

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		Master channel(axis) number (start from 0)
SlaveAxisNo	in	long		Slave channel(axis) number (start from 0)
SlaveRatio	in	double		Gear ratio of slave to master axis(1)

Return AXT_RT_SUCCESS(0000) : Successful execution of API.

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

Description

When a transfer command is ordered on a specific axis, this API allows that the start of the transfer motion is synchronized with the motion situation of the master axis. Set master axis and slave axis so that when the master axis moves, the slave axis moves together. Moving ratio is the ratio of a slave axis's move per one rotation of a master axis. (SlaveRatio 0 ~ 100% 0 : 0%, 0.5 : 50%, 1.0 : 100%)

C Example

```
// Since the ratio of master axis to slave axis is set to 0.5,
// when in move, if the master moves by 1, the slave moves by 0.5.
AxmLinkSetMode(0, 1, 0.5);

// Returns electric gear ratio between Master axis and Slave axis.
double dpGearRatio;
long lpSlaveAxis;
AxmLinkGetMode (0, &lpSlaveAxis, &dpGearRatio);
```

VB Example

```
'Since the ratio of master axis to slave axis is set to 0.5,
'when in move, if the master moves by 1, the slave moves by 0.5.
AxmLinkSetMode 0, 1, 0.5

'Returns electric gear ratio between Master axis and Slave axis.
Dim dpGearRatio As Double
Dim lpSlaveAxis As Long
```

```
AxmLinkGetMode 0, lpSlaveAxis, dpGearRatio
```

Delphi Example

```
{ Returns electric gear ratio between Master axis and Slave axis.  
var  
dpGearRatio: Double;  
lpSlaveAxis: LongInt;  
Begin  
  
{ Since the ratio of master axis to slave axis is set to 0.5,  
when in move, if the master moves by 1, the slave moves by 0.5.}  
AxmLinkSetMode(0, 1, 0.5);  
  
AxmLinkGetMode (0, @lpSlaveAxis, @dpGearRatio);  
end;
```

See Also

[AxmLinkGetMode](#), [AxmLinkResetMode](#)

AxmLinkGetMode

Purpose

Return electric gear ratio between Master axis and Slave axis.

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		Master channel(axis) number (start from 0)
SlaveAxisNo	out	long		Slave channel(axis) number
SlaveRatio	out	double*		Gear ratio of slave to master axis(1)

Return AXT_RT_SUCCESS(0000) : Successful execution of API.

* [See error code Table for more information on status error codes](#)

Description

It returns the ratio between the master axis and the slave axis such that when the master axis moves, the slave axis moves together.

Moving ratio is the ratio of a slave axis's move per one rotation of a master axis.

C Example

```
// Since the ratio of master axis to slave axis is set to 0.5,
// when in move, if the master moves by 1, the slave moves by 0.5.
AxmLinkSetMode (0, 1, 0.5);

// Returns electric gear ratio between Master axis and Slave axis.
double dpGearRatio;
long lpSlaveAxis;
AxmLinkGetMode(0, &lpSlaveAxis, &dpGearRatio);
```

VB Example

```
'Since the ratio of master axis to slave axis is set to 0.5,
'when in move, if the master moves by 1, the slave moves by 0.5.
AxmLinkSetMode 0, 1, 0.5

'Returns electric gear ratio between Master axis and Slave axis.
```

```
Dim dpGearRatio As Double  
Dim lpSlaveAxis As Long  
AxmLinkGetMode 0, lpSlaveAxis, dpGearRatio
```

Delphi Example

```
{ Returns electric gear ratio between Master axis and Slave axis.}  
var  
dpGearRatio: Double;  
lpSlaveAxis: LongInt;  
Begin  
  
{ Since the ratio of master axis to slave axis is set to 0.5,  
when in move, if the master moves by 1, the slave moves by 0.5.}  
AxmLinkSetMode (0, 1, 0.5);  
  
AxmLinkGetMode(0, @lpSlaveAxis, @dpGearRatio);  
end;
```

See Also

[AxmLinkSetMode](#), [AxmLinkResetMode](#)

AxmLinkResetMode

Purpose

Clears setting of electric gear ratio between Master axis and Slave axis.

Format

C

```
DWORD AxmLinkResetMode (long IMasterAxisNo);
```

Visual Basic

```
Function AxmLinkResetMode (ByVal IMasterAxisNo As Long) As Long
```

Delphi

```
function AxmLinkResetMode (IMasterAxisNo : LongInt) : DWord; stdcall;
```

Input / Output

Name	in/out	Format	Init Value	Explanation
MasterAxisNo	in	long		Master channel(axis) number (start from 0)

Return AXT_RT_SUCCESS(0000) : Successful execution of API.

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

Description

It clears the ratio between master axis and slave axis such that when the master axis moves, the slave axis moves together.

Moving ratio is the ratio of a slave axis's move per one rotation of a master axis.

C Example

```
// Clear setting of electric gear ratio between Master axis and Slave
axis.
AxmLinkResetMode (0);
```

VB Example

```
'Clear setting of electric gear ratio between Master axis and Slave axis.
AxmLinkResetMode 0
```

Delphi Example

```
{ Clear setting of electric gear ratio between Master axis and Slave
axis.}
AxmLinkResetMode (0);
```

See Also

[AxmLinkSetMode](#), [AxmLinkGetMode](#)

Interpolation Move API (Line, Circle)

In this chapter, we will discuss interpolation motion control. Interpolation means that more than two axes are associated with each other in order to execute the motions such as linear interpolation, circular interpolation and so on.

Concurrent control of multiple axes can be considered a same function as multi axis motion control. However, there is a big difference between them in that interpolation automatically controls the velocity of each axis related with interpolation move in order to follow the path that user requires. For CAMC-IP, it supports 2 axes interpolation and 2 axes circular interpolation. For CAMC-QI, it can execute linear interpolation move from 2 axes to 4 axes, 2 axes circular interpolation move, 3 axes or 4 axes spline, and helical interpolation moving tasks automatically using continuous interpolation. Generally, linear interpolation, circular interpolation and helical interpolation can execute one interpolation, but spline interpolation cannot be executed without using continuous interpolation.

Besides, if universal output, triggers, interrupts supported in each node can be used during continuous interpolation, a better application equipment can be built by using velocity settings, node acceleration/deceleration and automatic acceleration/deceleration. This chapter describes how to use general interpolation for linear interpolation and circular interpolation and how to use continuous interpolation.

Function	Description
AxmLineMove	<p>Multi axis linear interpolation moves by designating a start point and an end point. Exits API after move has started.</p> <p>Used in conjunction with AxmContiBeginNode, AxmContiEndNode, it specifies a start point and an end point on a specific coordination system and becomes a stored API in the linear interpolation queue. For linear profile continuous interpolation move, it is stored in the internal queue and is started by using AxmContiStart API.</p>
AxmCircleCenterMove	<p>Circular interpolation moves by designating a start point, a middle point and an end point. Exits API after move has started.</p> <p>Used in conjunction with AxmContiBeginNode, AxmContiEndNode, it specifies a start point, a middle point and an end point on a specific coordination system and becomes a stored API in the circular interpolation queue. For profile circular continuous interpolation move, it is stored in internal queue and is started by using AxmContiStart API.</p>
AxmCirclePointMove	<p>Circular interpolation moves by designating a middle point and an end point. Exits API after move has started.</p> <p>Used in conjunction with AxmContiBeginNode, AxmContiEndNode, it specifies a middle point and an end point on a specific coordination system and becomes a stored API in the circular interpolation queue. For profile circular continuous interpolation move, it is stored in internal queue</p>

	and is started by using AxmContiStart API.
AxmCircleRadiusMove	Circular interpolation moves by designating a start point, an end point and a radius. Exits API after move has started. Used in conjunction with AxmContiBeginNode , AxmContiEndNode , it specifies a start point, an end point and a radius on a specific coordination system and becomes a stored API in the circular interpolation queue. For profile circular continuous interpolation move, it is stored in internal queue and is started by using AxmContiStart API.
AxmCircleAngleMove	Circular interpolation moves by designating a start point, a rotational angle and a radius. Exits API after move has started. Used in conjunction with AxmContiBeginNode , AxmContiEndNode , it specifies a start point, a rotational angle and a radius on a specific coordination system and becomes a stored API in the circular interpolation queue. For profile circular continuous interpolation move, it is stored in internal queue and is started by using AxmContiStart API.

The previously mentioned multi axis move was the multi axis move such that two axes simultaneously start and execute actions of each. However, there is an occasion such that two axes have to move according to specific paths of lines or circular arcs through co-related movements. In order to help such moves, linear interpolation, circular interpolation, continuous interpolation moves, etc. for each. Settings for velocity profile and absolute/relative coordinates of these interpolation moves are designed to follow the axis that has the lowest axis number among those axes with interpolation move. Therefore, be careful in the settings.

► Caution: Match all the axes that interpolate Unit/Pulse equally. If the Unit/Pulse are not matched with each other, all the units will be different. They must be identical to interpolation control.

► Caution: When using a motion board with more than 4 axes motion (CAMC-IP)
In boards with more than four axes motion, two axes of Axis 0–1 are categorized as axis group 1, two axes of Axis 2–3 as axis group 2, four axes of Axis 4–5 as axis group 3, Axis 6–7 as axis group 4, and so on. Linear interpolation move and circular interpolation move among the axes in the same group are available to use, but those among the axes in different axis groups are not. It is because CAMC-IP chip is two axes and the chip cannot support these features. You must keep that in mind when setting coordinates.

► Caution: When using a motion board with more than 8 axes motion (CAMC-QI)
In boards with more than eight axes motion, four axes are categorized as axis group 1, four axes of Axis 4–7 as axis group 2, four axes of Axis 8–11 as axis group 3, and so on. Linear interpolation move and circular interpolation move among the axes in the same group are available to use, but those among the axes in different axis groups are not. It is because CAMC-QI chip is four axes and the chip cannot support these features. You must keep that in mind when setting coordinates.

AxmLineMove

Purpose

API that multi axis linear interpolation moves by designating a start point and an end point on a coordination system.

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		Coordination system number (start from 0)
Pos	in	double		Position array
Vel	in	double		Moving velocity (+direction: more than 0 , – direction: less than 0, The unit of moving velocity is PPS[Pulses/sec] when Unit/pulse is set to 1/1)
Accel	in	double		Moving acceleration (The unit of acceleration is PPS[Pulses/sec^2] when Unit/pulse is set to 1/1)
Decel	in	double		Moving deceleration (The unit of acceleration is PPS[Pulses/sec^2] when Unit/pulse is set to 1/1)

Return AXT_RT_SUCCESS(0000) : Successful execution of API.

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

Description

It exits API after move has started. Used in conjunction with [AxmContiBeginNode](#) , [AxmContiEndNode](#) , it specifies a start point and an end point on a specific coordination system and becomes a stored API in the linear interpolation queue. For linear profile continuous interpolation move, it is stored in the internal queue and is started by using [AxmContiStart](#) API: for IP, two axes are available, and for QI, two to four axes are available. Before executing this API, [AxmContiSetAxisMap](#) API does axis mapping on the axes on which interpolation tasks will be executed into map numbers. It is also applied to APIs that are related with general interpolation control, not only those related with continuous interpolation since

parameter transmission is simple on them. Therefore, this API must be first used in order to map the axes that are to be used in prior to using APIs that are related with general interpolation control.

- ▶ By AxmSStop API and AxmEstop API, you can execute motion deceleration stop and motion emergency stop. Arguments must be stopped by inserting master axis of the Coordinate.
- ▶ You can verify whether it is in motion using [AxmStatusReadInMotion](#) API.
- ▶ When using [AxmSignalSetInpos](#) API, if Inpos input signal is set to Enable, it won't return until INP input becomes ON although command pulse out has been completed as it is regarded that motion has not been completed. Please refer to [AxmSignalSetInpos](#) API for details. **
- ▶ Caution: Match all the axes that interpolate Unit/Pulse equally. If the Unit/Pulse are not matched with each other, all the units will be different. They must be identical to interpolation control.
- ▶ Caution: When using a motion board with more than 4 axes motion (CAMC-IP)
In boards with more than four axes motion, two axes of Axis 0–1 are categorized as axis group 1, two axes of Axis 2–3 as axis group 2, four axes of Axis 4–5 as axis group 3, Axis 6–7 as axis group 4, and so on. Linear interpolation move and circular interpolation move among the axes in the same group are available to use, but those among the axes in different axis groups are not. It is because CAMC-IP chip is two axes and the chip cannot support these features. You must keep that in mind when setting coordinates.
- ▶ Caution: When using a motion board with more than 8 axes motion (CAMC-QI)
In boards with more than eight axes motion, four axes are categorized as axis group 1, four axes of Axis 4–7 as axis group 2, four axes of Axis 8–11 as axis group 3, and so on. Linear interpolation move and circular interpolation move among the axes in the same group are available to use, but those among the axes in different axis groups are not. It is because CAMC-QI chip is four axes and the chip cannot support these features. You must keep that in mind when setting coordinates.

Continuous interpolation API list

Function	Description	Remark
AxmContiSetAxisMap	Sets continuous interpolation axis mapping on a specific coordination system.	
AxmContiGetAxisMap	Returns continuous interpolation axis mapping on a specific coordination system.	
AxmContiSetAbsRelMode	Sets continuous interpolation axis absolute/relative mode on a specific coordination system.	
AxmContiGetAbsRelMode	Returns continuous interpolation axis absolute/relative mode on a specific coordination system.	
AxmContiBeginNode	Starts registering the tasks to be executed during continuous interpolation on a specific coordination system. After calling this API, all the motion tasks	

	that are executed until AxmContiEndNode API is called, are not that they execute actual motions but that they are registered as continuous interpolation motions. Only when AxmContiStart is called, the registered motions actually get executed.	
AxmContiEndNode	Exits registration of the tasks to execute continuous interpolation on a specific coordination system.	
AxmContiReadFree	Verifies whether the internal queue for interpolation move on a specific coordination system is empty.	
AxmContiReadIndex	Verifies the number of interpolation moves stored in the internal queue for interpolation move on a specific coordination system.	
AxmContiWriteClear	Deletes all internal queues stored for continuous interpolation move on a specific coordination system.	
AxmContiStart	Starts move of the internal continuous interpolation queue stored in a specific coordination system.	
AxmContiIsMotion	Verifies whether it is in continuous interpolation move on a specific coordination system.	
AxmContiGetNodeNum	Verifies the currently in-move continuous interpolation index number while in continuous interpolation move on a specific coordination system.	
AxmContiGetTotalNodeNum	Verifies the total number of continuous interpolation move indexes that are set on a specific coordination system.	

Usage examples of general move API >

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);
// Relative position move
```

```
AxmContiSetAbsRelMode(lCoordinate, 1);
// Linear interpolation move
AxmLineMove(lCoordinate, dPos, 200, 400, 400);
```

VB Example

```
im 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

AxmContiWriteClear lCoordinate
AxmContiSetAxisMap lCoordinate, 2, axes(0)
    ' Relative position move
AxmContiSetAbsRelMode lCoordinate, 1
    ' Linear interpolation move
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;

    AxmContiWriteClear(lCoordinate);
    AxmContiSetAxisMap (lCoordinate, lPosSize, @lAxis);

    { Relative position move }
    AxmContiSetAbsRelMode(lCoordinate, 1);

    { Linear interpolation move }
    AxmLineMove(lCoordinate, @dPos, 200, 400, 400);

end;
```

Usage examples of continuous interpolation move API >

It is the API that is stored in the internal queue for linear continuous interpolation move. This API is used in order to store interpolation commands in the internal queue when continuous

interpolation move is used. This API is also used to store linear interpolation move while in continuous interpolation move. If '[AxmContiStart](#)' API is executed after storing the interpolation motions to be executed in the internal queue, all the stored interpolation moves can be continuously executed.

C Example

```
long    lAxis[2];
double  dPos[2];
long    lPosSize = 2;
long    lCoordinate = 0;
double  dVelocity = 200, dAccel = 400;

lAxis[0] = 0;
lAxis[1] = 1;

AxmContiWriteClear(lCoordinate);
AxmContiSetAxisMap (lCoordinate, lPosSize, lAxis);
// Register absolute position move
AxmContiSetAbsRelMode(lCoordinate, 0);

AxmContiBeginNode(lCoordinate);
dPos[0] = 100 , dPos[1] = 100;
AxmLineMove(lCoordinate, dPos, dVelocity, dAccel, dAccel);
dPos[0] = 200 , dPos[1] = 200;
AxmLineMove(lCoordinate, dPos, dVelocity, dAccel, dAccel);
dPos[0] = 300 , dPos[1] = 300;
AxmLineMove(lCoordinate, dPos, dVelocity, dAccel, dAccel);
dPos[0] = 400 , dPos[1] = 400;
AxmLineMove(lCoordinate, dPos, dVelocity, dAccel, dAccel);
dPos[0] = 500 , dPos[1] = 500;
AxmLineMove(lCoordinate, dPos, dVelocity, dAccel, dAccel);
dPos[0] = 600 , dPos[1] = 600;
AxmLineMove(lCoordinate, dPos, dVelocity, dAccel, dAccel);
dPos[0] = 700 , dPos[1] = 700;
AxmLineMove(lCoordinate, dPos, dVelocity, dAccel, dAccel);
dPos[0] = 800 , dPos[1] = 800;
AxmLineMove(lCoordinate, dPos, dVelocity, dAccel, dAccel);
dPos[0] = 900 , dPos[1] = 900;
AxmLineMove(lCoordinate, dPos, dVelocity, dAccel, dAccel);
dPos[0] = 1000 , dPos[1] = 1000;
AxmLineMove(lCoordinate, dPos, dVelocity, dAccel, dAccel);
dPos[0] = 1100 , dPos[1] = 1100;
AxmLineMove(lCoordinate, dPos, dVelocity, dAccel, dAccel);
AxmContiEndNode (lCoordinate);

// Start absolute position move
AxmContiSetAbsRelMode(lCoordinate, 0);

// Start continuous interpolation move
AxmContiStart (lCoordinate, 0, 0);
```

VB Example

```
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

AxmContiWriteClear lCoordinate
AxmContiSetAxisMap lCoordinate, lPosSize, axes(0)

' Register absolute position move
AxmContiSetAbsRelMode lCoordinate, 0

AxmContiBeginNode lCoordinate
dPos(0) = 100: dPos(1) = 100
AxmLineMove lCoordinate, dPos(0), dVelocity, dAccel, dAccel
dPos(0) = 200: dPos(1) = 200
AxmLineMove lCoordinate, dPos(0), dVelocity , dAccel, dAccel
dPos(0) = 300: dPos(1) = 300
AxmLineMove lCoordinate, dPos(0), dVelocity , dAccel, dAccel
dPos(0) = 400: dPos(1) = 400
AxmLineMove lCoordinate, dPos(0), dVelocity , dAccel, dAccel
dPos(0) = 500: dPos(1) = 500
AxmLineMove lCoordinate, dPos(0), dVelocity , dAccel, dAccel
dPos(0) = 600: dPos(1) = 600
AxmLineMove lCoordinate, dPos(0), dVelocity , dAccel, dAccel
dPos(0) = 700: dPos(1) = 700
AxmLineMove lCoordinate, dPos(0), dVelocity , dAccel, dAccel
AxmContiEndNode lCoordinate

' Start absolute position move
AxmContiSetAbsRelMode lCoordinate, 0

' Start continuous interpolation move
AxmContiStart lCoordinate, 0, 0

```

Delphi Example

```

var
  i : LongInt;
  dVelocity, dAccel : Double;
  lAxis : array [0..1] of LongInt;
  dPos : array [0..1] of Double;
  lCoordinate : LongInt;
  lPosSize : LongInt;

begin
  dVelocity := 200;
  dAccel := 400;
  lPosSize := 2;
  lCoordinate := 0;

  // Register absolute position move
  AxmContiSetAbsRelMode(lCoordinate, 0);

  for i := 0 to 1 do
  begin
    lAxis[i] := i;
  end;

  AxmContiWriteClear(lCoordinate);
  AxmContiSetAxisMap (lCoordinate, 2, @lAxis);

```

```
AxmContiBeginNode(lCoordinate);
dPos[0] := 100 ; dPos[1] := 100;
AxmLineMove(lCoordinate, @dPos, dVelocity, dAccel, dAccel);
dPos[0] := 200 ; dPos[1] := 200;
AxmLineMove(lCoordinate, @dPos, dVelocity, dAccel, dAccel);
dPos[0] := 300 ; dPos[1] := 300;
AxmLineMove(lCoordinate, @dPos, dVelocity, dAccel, dAccel);
dPos[0] := 400 ; dPos[1] := 400;
AxmLineMove(lCoordinate, @dPos, dVelocity, dAccel, dAccel);
dPos[0] := 500 ; dPos[1] := 500;
AxmLineMove(lCoordinate, @dPos, dVelocity, dAccel, dAccel);
dPos[0] := 600 ; dPos[1] := 600;
AxmLineMove(lCoordinate, @dPos, dVelocity, dAccel, dAccel);
dPos[0] := 700 ; dPos[1] := 700;
AxmLineMove(lCoordinate, @dPos, dVelocity, dAccel, dAccel);
AxmContiEndNode (lCoordinate);

// Start absolute position move
AxmMotSetAbsRelMode(lCoordinate, 0);

{ Start continuous interpolation move }
AxmContiStart (lCoordinate, 0, 0);
end;
```

See Also

[AxmCircleCenterMove](#), [AxmCirclePointMove](#), [AxmCircleRadiusMove](#), [AxmCircleAngleMove](#),
[AxmContiSetAxisMap](#), [AxmContiGetAxisMap](#), [AxmContiSetAbsRelMode](#),
[AxmContiGetAbsRelMode](#), [AxmContiBeginNode](#), [AxmContiEndNode](#), [AxmContiReadFree](#),
[AxmContiReadIndex](#), [AxmContiWriteClear](#), [AxmContiStart](#), [AxmContiIsMotion](#),
[AxmContiGetNodeNum](#), [AxmContiGetTotalNodeNum](#), [AxmContiSetOptionNodeNum](#)

AxmCircleCenterMove

Purpose

Circular interpolation moves by designating a start point, a middle point and an end point on a specific coordination system. Exits API after move has started.

Format

C

```
DWORD AxmCircleCenterMove (long lCoord, long *lAxisNo, double *dCenterPos, double *dEndPos, double dVel,
                           double dAccel, double dDecel, DWORD uCWDdir);
```

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 uCWDdir As
                             Long) As Long
```

Delphi

```
function AxmCircleCenterMove (lCoord : LongInt; lAxisNo : PLongInt; dCenterPos : PDouble; dEndPos : PDouble;
                             dVel : Double; dAccel : Double; dDecel : Double; uCWDdir : DWord) : DWord; stdcall;
```

Input / Output

Name	in/out	Format	Init Value	Explanation
Coord	in	long		Coordination system number (start from 0)
AxisNo	in	long*		Channel(axis) number array (start from 0) to be used for interpolation
CenterPos	In	double*		Home position array on X axis, Y axis
EndPos	in	double*		Exit position array on X axis, Y axis
Vel	in	double		Moving velocity (+direction: more than 0 , - direction: less than 0, The unit of moving velocity is PPS[Pulses/sec] when Unit/pulse is set to 1/1)
Accel	in	double		Moving acceleration (The unit of acceleration is PPS[Pulses/sec^2] when Unit/pulse is set to 1/1)
Decel	in	double		Moving deceleration (The unit of acceleration is PPS[Pulses/sec^2] when Unit/pulse is set to 1/1)
CWDdir	in	DWORD		Circular direction

uCWDdir

#define	Value	Explanation
DIR_CCW	00h	Counterclockwise

DIR_CW	01h	Clockwise
--------	-----	-----------

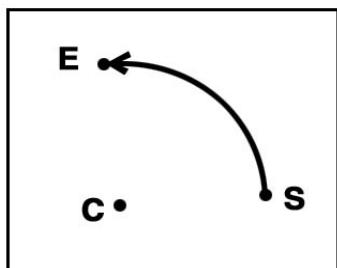
Return AXT_RT_SUCCESS(0000) : Successful execution of API.
 * See error code Table for more information on status error codes

Description

Circular interpolation is applied to two arbitrary axes. We will explain it assuming that the two mapped axes are axis X and axis Y. Axis X is the axis that has a lower axis number and axis Y is the one that has a higher axis number. For example, if axes Z, U are the two mapped axes, then axis Z corresponds to axis X and axis U corresponds with axis Y.

Used in conjunction with [AxmContiBeginNode](#), [AxmContiEndNode](#), it specifies a start point, a middle point and an end point on a specific coordination system and becomes a stored API in the circular interpolation queue. For profile circular continuous interpolation move, it is stored in the internal queue and is started by using [AxmContiStart](#) API. Before executing this API, [AxmContiSetAxisMap](#) API does axis mapping on the axes on which interpolation tasks will be executed into map numbers. It is also applied to APIs that are related with general interpolation control, not only those related with continuous interpolation since parameter transmission is simple on them. Therefore, this API must be first used in order to map the axes that are to be used in prior to using APIs that are related with general interpolation control.

The current position becomes the start point of circular move, and it circular moves based on the preset middle point and the end point.



- ▶ By AxmSStop API and AxmEstop API, you can execute motion deceleration stop and motion emergency stop. Arguments must be stopped by inserting master axis of the Coordinate.
- ▶ You can verify whether it is in motion using [AxmStatusReadInMotion](#) API.
- ▶ When using [AxmSignalSetInpos](#) API, if Inpos input signal is set to Enable, it won't return until INP input becomes ON although command pulse out has been completed as it is regarded that motion has not been completed. Please refer to [AxmSignalSetInpos](#) API for details. **
- ▶ Caution: Match all the axes that interpolate Unit/Pulse equally. If the Unit/Pulse is not matched with each other, all the units will be different. They must be identical to interpolation control.
- ▶ Caution: When using a motion board with more than 4 axes motion (SMC-2V03) In boards with more than four axes motion, two axes of Axis 0-1 are categorized as axis group 1, two axes of Axis 2-3 as axis group 2, four axes of Axis 4-5 as axis group 3, Axis 6-7 as axis group 4, and so on. Linear interpolation move and circular interpolation move among the axes in the same group are available to use, but those among the axes in different axis groups are not. It is because SMC-2V03 is two axes and the board cannot support these features. You must keep that in mind when setting

coordinates.

►Caution: When using a motion board with more than 8 axes motion (PCI-N804/404)

In boards with more than eight axes motion, four axes are categorized as axis group 1, four axes of Axis 4–7 as axis group 2, four axes of Axis 8–11 as axis group 3, and so on. Linear interpolation move and circular interpolation move among the axes in the same group are available to use, but those among the axes in different axis groups are not. It is because PCI-N804/404 chip is four axes and the board cannot support these features. You must keep that in mind when setting coordinates.

► Continuous interpolation API list

Function	Description	Remark
AxmContiSetAxisMap	Sets continuous interpolation axis mapping on a specific coordination system.	
AxmContiGetAxisMap	Returns continuous interpolation axis mapping on a specific coordination system.	
AxmContiSetAbsRelMode	Sets continuous interpolation axis absolute/relative mode on a specific coordination system.	
AxmContiGetAbsRelMode	Returns continuous interpolation axis absolute/relative mode on a specific coordination system.	
AxmContiBeginNode	Starts registering the tasks to be executed during continuous interpolation on a specific coordination system. After calling this API, all the motion tasks that are executed until AxmContiEndNode API is called, are not that they execute actual motions but that they are registered as continuous interpolation motions. Only when AxmContiStart is called, the registered motions actually get executed.	
AxmContiEndNode	Exits registration of the tasks to execute continuous interpolation on a specific coordination system.	
AxmContiReadFree	Verifies whether the internal queue for interpolation move on a specific coordination system is empty.	
AxmContiReadIndex	Verifies the number of interpolation moves stored in the internal queue for interpolation move on a specific coordination system.	
AxmContiWriteClear	Deletes all internal queues stored for continuous interpolation move on a specific coordination system.	
AxmContiStart	Starts move of the internal continuous interpolation	

	queue stored in a specific coordination system.	
AxmContiIsMotion	Verifies whether it is in continuous interpolation move on a specific coordination system.	
AxmContiGetNodeNum	Verifies the currently in-move continuous interpolation index number while in continuous interpolation move on a specific coordination system.	
AxmContiGetTotalNodeNum	Verifies the total number of continuous interpolation move indexes that are set on a specific coordination system.	

Examples of general interpolation >

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;

AxmContiWriteClear(lCoordinate);
AxmContiSetAxisMap (lCoordinate, lPosSize, lAxis);
AxmContiSetAbsRelMode(lCoordinate, 1); // Relative position move

// Circular interpolation move in CW direction
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

AxmContiWriteClear lCoordinate
AxmContiSetAxisMap lCoordinate, 2, axes(0)
AxmContiSetAbsRelMode lCoordinate, 1

' Circular interpolation move in CW direction
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;

  AxmContiWriteClear(lCoordinate);
  AxmContiSetAxisMap (lCoordinate, lPosSize, @lAxis);
  AxmContiSetAbsRelMode(lCoordinate, 1); // relative position move

  AxmCircleCenterMove(lCoordinate, @lAxis, @dCenPos, @dEndPos, 200, 400,
    400, DIR_CW);

End;

```

Examples of continuous interpolation >

C Example

```

long   lAxis[2];
double dPos[2];
long   lPosSize = 2;
long   lCoordinate = 0;
double dCenPos[2];
double dEndPos[2];
double dVelocity = 200;
double dAccel = 400;

for(int i=0; i<lPosSize; i++)
{
    lAxis[i]      = i;
}

AxmContiWriteClear(lCoordinate);
AxmContiSetAxisMap (lCoordinate, lPosSize, lAxis);

```

```

// Relative position move
AxmContiSetAbsRelMode(lCoordinate, 1);

AxmContiBeginNode(lCoordinate);
dPos[0] = 100 , dPos[1] = 100;
AxmLineMove(lCoordinate, dPos, dVelocity, dAccel, dAccel);
dPos[0] = 200 , dPos[1] = 200;
AxmLineMove(lCoordinate, dPos, dVelocity, dAccel, dAccel);
dCenPos[0] = 300 , dCenPos[1] = 300; dEndPos[0] = 400; dEndPos[1] = 400;
AxmCircleCenterMove(lCoordinate, lAxis, dCenPos, dEndPos, dVelocity,
    dAccel, dAccel, DIR_CCW);
AxmLineMove(lCoordinate, dPos, dVelocity, dAccel, dAccel);
dPos[0] = 400 , dPos[1] = 400;
AxmLineMove(lCoordinate, dPos, dVelocity, dAccel, dAccel);
dCenPos[0] = 500 , dCenPos[1] = 500; dEndPos[0] = 600; dEndPos[1] = 600;
AxmCircleCenterMove(lCoordinate, lAxis, dCenPos, dEndPos, dVelocity,
    dAccel, dAccel, DIR_CCW);
AxmLineMove(lCoordinate, dPos, dVelocity, dAccel, dAccel);
dPos[0] = 600 , dPos[1] = 600;
AxmLineMove(lCoordinate, dPos, dVelocity, dAccel, dAccel);
dPos[0] = 700 , dPos[1] = 700;
AxmLineMove(lCoordinate, dPos, dVelocity, dAccel, dAccel);
AxmContiEndNode (lCoordinate);

// Relative position move
AxmContiSetAbsRelMode(lCoordinate, 1);
AxmContiStart (lCoordinate, 0, 0);

```

VB Example

```

Dim dVelocity As Double
Dim dAccel As Double
Dim lPosSize As Long
Dim dPos(0 To 1) As Double
Dim dCenPos(0 To 1) As Double
Dim lCoordinate As Long
Dim axes(0 To 1) As Long

Dim i As Long
lPosSize = 2
lCoordinate = 0

For i = 0 To 1
    axes(i) = i
Next i

dVelocity = 200
dAccel = 400

AxmContiWriteClear lCoordinate
AxmContiSetAxisMap lCoordinate, lPosSize, axes(0)

' Relative position move
AxmContiSetAbsRelMode lCoordinate, 1

AxmContiBeginNode lCoordinate
dPos(0) = 100: dPos(1) = 100
AxmLineMove lCoordinate, dPos(0), dVelocity, dAccel, dAccel
dPos(0) = 200: dPos(1) = 200
AxmLineMove lCoordinate, dPos(0), dVelocity , dAccel, dAccel
dPos(0) = 300: dPos(1) = 300
AxmLineMove lCoordinate, dPos(0), dVelocity, dAccel, dAccel
AxmCircleCenterMove lCoordinate, axes(0), dCenPos(0), dPos(0), dVelocity,
    dAccel, dAccel, DIR_CCW
dPos(0) = 400: dPos(1) = 400
AxmLineMove lCoordinate, dPos(0), dVelocity, dAccel, dAccel

```

```

AxmCircleCenterMove lCoordinate, axes(0), dCenPos(0), dPos(0), dVelocity,
    dAccel, dAccel, DIR_CCW
dPos(0) = 500; dPos(1) = 500
AxmLineMove lCoordinate, dPos(0), dVelocity , dAccel, dAccel
dPos(0) = 600; dPos(1) = 600
AxmLineMove lCoordinate, dPos(0), dVelocity, dAccel, dAccel
dPos(0) = 700; dPos(1) = 700
AxmLineMove lCoordinate, dPos(0), dVelocity , dAccel, dAccel
AxmContiEndNode lCoordinate

AxmContiSetAbsRelMode lCoordinate, 1

AxmContiStart lCoordinate, 0, 0

```

Delphi Example

```

var
  i : LongInt;
  dVelocity, dAccel : Double;
  lAxis : array [0..1] of LongInt;
  dPos : array [0..1] of Double;
  dCenPos : array [0..1] of Double;
  dEndPos : array [0..1] of Double;
  lCoordinate : LongInt;
  lPosSize : LongInt;

begin
  dVelocity := 200;
  dAccel := 400;
  lPosSize := 2;
  lCoordinate := 0;

  for i := 0 to 1 do
  begin
    lAxis[i] := i;
  end;

  // Register as relative coordinates
  AxmContiWriteClear(lCoordinate);
  AxmContiSetAxisMap (lCoordinate, 2, @lAxis);
  AxmContiSetAbsRelMode(lCoordinate, 1);

  AxmContiBeginNode(lCoordinate);
  dPos[0] := 100 ; dPos[1] := 100;
  AxmLineMove(lCoordinate, @dPos, dVelocity, dAccel, dAccel);
  dPos[0] := 200 ; dPos[1] := 200;
  AxmLineMove(lCoordinate, @dPos, dVelocity, dAccel, dAccel);
  dPos[0] := 300 ; dPos[1] := 300;
  dCenPos[0] := 300; dCenPos[1] := 300; dEndPos[0] := 400; dEndPos[1] :=
    400;
  AxmCircleCenterMove(lCoordinate, @lAxis, @dCenPos, @dEndPos, dVelocity,
    dAccel, dAccel, 0);
  dPos[0] := 400 ; dPos[1] := 400;
  AxmLineMove(lCoordinate, @dPos, dVelocity, dAccel, dAccel);
  dPos[0] := 500 ; dPos[1] := 500;
  dCenPos[0] := 500; dCenPos[1] := 500; dEndPos[0] := 600;
  dEndPos[1] := 600;
  AxmCircleCenterMove(lCoordinate, @lAxis, @dCenPos, @dEndPos, dVelocity,
    dAccel, dAccel, 0);
  dPos[0] := 600 ; dPos[1] := 600;
  AxmLineMove(lCoordinate, @dPos, dVelocity, dAccel, dAccel);
  dPos[0] := 700 ; dPos[1] := 700;
  AxmLineMove(lCoordinate, @dPos, dVelocity, dAccel, dAccel);
  AxmContiEndNode (lCoordinate);

```

```
// Register as relative coordinates  
AxmMotSetAbsRelMode(lCoordinate, 1);  
  
AxmContiStart (lCoordinate, 0, 0);  
end;
```

See Also

[AxmLineMove](#), [AxmCirclePointMove](#), [AxmCircleRadiusMove](#), [AxmCircleAngleMove](#), [AxmContiSetAxisMap](#),
[AxmContiGetAxisMap](#), [AxmContiSetAbsRelMode](#), [AxmContiGetAbsRelMode](#), [AxmContiBeginNode](#),
[AxmContiEndNode](#), [AxmContiReadFree](#), [AxmContiReadIndex](#), [AxmContiWriteClear](#), [AxmContiStart](#) ,
[AxmContiIsMotion](#), [AxmContiGetNodeNum](#), [AxmContiGetTotalNodeNum](#), [AxmContiSetOptionNodeNum](#)

AxmCirclePointMove

Purpose

API for circular interpolation by specifying middle point and end point on the specified coordinate system. After the move starts, it exits the API.

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		Coordinate system number(start from 0)
AxisNo	in	long*		Channel(axis) number(start from 0) array used for interpolation
MidPos	In	double*		Array of X middle point, and Y middle point
EndPos	in	double*		Array of X end point and Y end point
Vel	in	double		Move speed(+direction:more than 0, - direction: less than 0, move speed unit is PPS[Pulses/sec] when Unit/pulse is 1/1)
Accel	in	double		Move acceleration(acceleration unit is PPS[Pulses/sec^s] when Unit/pulse is 1/1)
Decel	in	double		Move deceleration(deceleration unit is PPS[Pulses/sec^2] when Unit/pulse is 1/1)
ArcCircle	in	long		Arc(0), Circle(1) selection value

Return AXT_RT_SUCCESS(0000) : Successful execution of API.

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

Description

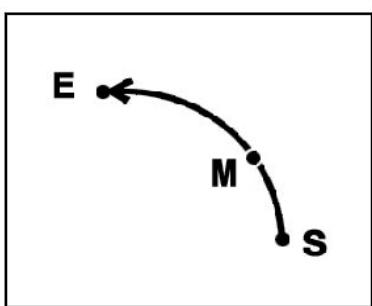
Circular interpolation applies to any two axes. This explanation assumes two mapped axes as X and Y axis. X axis means lower axis number and Y axis means higher axis number in two mapped axes.

For example, when Z and U axis are two mapped axes, Z axis corresponds to X axis and U axis corresponds to Y axis.

Like [AxmContiBeginNode](#) and [AxmContiEndNode](#), when it is being used, it becomes store API in queue which moves by specifying middle and end point in the specified coordinate system. For profile circular continuous interpolation move, it starts [AxmContiStart](#) API by saving it in the internal queue.

Before executing this API, [AxmContiSetAxisMap](#) API does axis mapping which maps axes which is to execute interpolation process to axis number. Not only continuous interpolation, but also functions related to general interpolation control are applied because transmitting parameter is simple. Thus, before using APIs about general interpolation, axes for future use are mapped by this API.

Current position becomes the start point of circular move and it executes circular move going through middle point.



► Caution: After axis mapping using [AxmContiSetAxisMap](#), use from the lowest axis mapping. In case of circular interpolation, it is possible to move only when array it from the lowest axis.

► Caution: If signal of +LIMIT, -LIMIT is on, there's no move.

► By AxmSStop and AxmEstop API, motion deceleration stop and motion emergency stop are possible. Argument is always stopped by transmitting the first axis(the lowest actual axis number) of coordinate.

► [AxmContiIsMotion](#) API can verify whether it is on motion or not.

► When using [AxmSignalSetInpos](#) API, if Inpos is set to Enable, it doesn't return until INP input is set to ON although command pulse output is completed because it considers move is not completed. Refer to [AxmSignalSetInpos](#) API for further information. **

► Caution: Axes which interpolate Unit/Pulse should be equally set. If Unit/Pulse is not equally set, their unit will be different with each other. Always set them equally, and do interpolation control.

► Caution: When using motion board with more than 2 axes (SMC-2V03)

In the board with 8 axes by using SMC-2V03, two axes from Axis 0-1 become axis group 1, two from Axis 2-3 become axis group 2, two from Axis 4-5 become axis group 3, and Axis6-7 become axis group 4. Axes from the same group can process line interpolation move and circular interpolation move, but axes from different groups cannot. Keep in mind when specifying coordinate.

► Caution: When using motion board with more than 8 (PCI-N804/404)

In the board with multi axes by using PCI-N804/404, 4 axes from Axis 0-3 become Axis group 1, 4

from Axis 4–7 become Axis group 2, and another 4 from Axis8–11 become Axis group 3. Axes from the same group can process line interpolation move and circular interpolation move, but axes from different groups cannot. Keep in mind when specifying coordinate.

► List of continuous interpolation API list

API	Description	Note
AxmContiSetAxisMap	Set continuous interpolation axis mapping for specified coordinate system.	
AxmContiGetAxisMap	Return continuous interpolation axis mapping for specified coordinate system.	
AxmContiSetAbsRelMode	Set continuous interpolation axis absolute/relative mode for specified coordinate system.	
AxmContiGetAbsRelMode	Return continuous interpolation axis absolute/relative mode for specified coordinate system.	
AxmContiBeginNode	Start registering tasks which will be processed in continuous interpolation for specified coordinate system. After calling this API, all motion tasks processed before calling AxmContiEndNode API are registered as continuous interpolation motion, but not processing real motion. When AxmContiStart API is called, registered motion is actually started.	
AxmContiEndNode	End registering tasks to process continuous interpolation in specified coordinate system.	
AxmContiReadFree	Identify internal queue if it is empty for interpolation move in a specified coordinate system.	
AxmContiReadIndex	Identify the number of interpolation move in internal queue stored for interpolation move in a specified coordinate system.	
AxmContiWriteClear	Clear all internal queues stored for continuous interpolation move in a specified coordinate system.	
AxmContiStart	Start continuous interpolation move from	

	internal queue stored in a specified coordinate system.	
AxmContiIsMotion	Identify continuous interpolation move if it is running or not in a specified coordinate system.	
AxmContiGetNodeNum	Identify the index number of currently running continuous interpolation among continuous interpolation moves in specified coordinate system.	
AxmContiGetTotalNodeNum	Identify total number of index of continuous interpolation move set in specified coordinate system.	

Normal interpolation move example>

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;

AxmContiWriteClear(lCoordinate);
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

AxmContiWriteClear lCoordinate
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;

    AxmContiWriteClear(lCoordinate);
    AxmContiSetAxisMap (lCoordinate, lPosSize, @lAxis);
    AxmContiSetAbsRelMode lCoordinate, 1

    AxmCircleRadiusMove(lCoordinate, @lAxis, dRadius, @dEndPos, velocity,
        accel, accel, 1, uShortDistance);
end;

```

Example of circular interpolation move>

Start API of axis 2 circular interpolation move.

Current position becomes the start point of circular move and circular move is based on specified radius and end point.

C Example

```

long    lAxis[2];
double  dPos[2];
long    lPosSize = 2;
long    lCoordinate = 0;
DWORD   uShortDistance=0;
double  dEndPos[2];

```

```

double dRadius = 150;
double dVelocity = 200;
double dAccel = 400;

for(int i=0; i<lPosSize; i++)
{
    lAxis[i] = i;
}

AxmContiWriteClear(lCoordinate);
AxmContiSetAxisMap (lCoordinate, lPosSize, lAxis);
AxmContiSetAbsRelMode(lCoordinate, 1);

AxmContiBeginNode(lCoordinate);
dPos[0] = 100 , dPos[1] = 100;
AxmLineMove(lCoordinate, dPos, dVelocity, dAccel, dAccel);
dPos[0] = 200 , dPos[1] = 200;
AxmLineMove(lCoordinate, dPos, dVelocity, dAccel, dAccel);
dEndPos[0] = 300; dEndPos[1] = 0;
AxmCircleRadiusMove(lCoordinate, lAxis, dRadius, dEndPos, dVelocity,
                     dAccel, dAccel, DIR_CCW, uShortDistance);
dPos[0] = 400 , dPos[1] = 400;
AxmLineMove(lCoordinate, dPos, dVelocity, dAccel, dAccel);
dRadius = 250; dEndPos[0] = 500; dEndPos[1] = 0;
AxmCircleRadiusMove(lCoordinate, lAxis, dRadius, dEndPos, dVelocity,
                     dAccel, dAccel, DIR_CCW, uShortDistance);
dPos[0] = 600 , dPos[1] = 600;
AxmLineMove(lCoordinate, dPos, dVelocity, dAccel, dAccel);
dPos[0] = 700 , dPos[1] = 700;
AxmLineMove(lCoordinate, dPos, dVelocity, dAccel, dAccel);
AxmContiEndNode (lCoordinate);

AxmContiSetAbsRelMode(lCoordinate, 1);

AxmContiStart (lCoordinate, 0, 0);

```

VB Example

```

Dim dVelocity As Double
Dim dAccel As Double
Dim lPosSize As Long
Dim dPos(0 To 1) As Double
Dim lCoordinate As Long
Dim axes(0 To 1) As Long
Dim uShortDistance As Long
Dim dRadius As Double

Dim i As Long
lPosSize = 2
lCoordinate = 0
uShortDistance = 0

For i = 0 To 1
    axes(i) = i
Next i

dVelocity = 200
dAccel = 400
dRadius = 150

AxmContiWriteClear lCoordinate
AxmContiSetAxisMap lCoordinate, lPosSize, axes(0)
AxmContiSetAbsRelMode lCoordinate, 1

AxmContiBeginNode lCoordinate
dPos(0) = 100: dPos(1) = 100

```

```

AxmLineMove lCoordinate, dPos(0), dVelocity, dAccel, dAccel
dPos(0) = 200; dPos(1) = 200
AxmLineMove lCoordinate, dPos(0), dVelocity, dAccel, dAccel
dPos(0) = 300; dPos(1) = 300
AxmLineMove lCoordinate, dPos(0), dVelocity, dAccel, dAccel
dPos(0) = 300; dPos(1) = 0
AxmCircleRadiusMove lCoordinate, axes(0), dRadius, dPos(0), dVelocity,
    dAccel, dAccel, DIR_CCW, uShortDistance

dPos(0) = 400; dPos(1) = 400
AxmLineMove lCoordinate, dPos(0), dVelocity, dAccel, dAccel
dPos(0) = 300; dPos(1) = 0
AxmCircleRadiusMove lCoordinate, axes(0), dRadius, dPos(0), dVelocity,
    dAccel, dAccel, DIR_CCW, uShortDistance
dPos(0) = 500; dPos(1) = 500
AxmLineMove lCoordinate, dPos(0), dVelocity, dAccel, dAccel
dPos(0) = 600; dPos(1) = 600
AxmLineMove lCoordinate, dPos(0), dVelocity, dAccel, dAccel
dPos(0) = 700; dPos(1) = 700
AxmLineMove lCoordinate, dPos(0), dVelocity, dAccel, dAccel
AxmContiEndNode lCoordinate

AxmContiSetAbsRelMode lCoordinate, 1

AxmContiStart lCoordinate, 0, 0

```

Delphi Example

```

var
  i : LongInt;
  dVelocity, dAccel : Double;
  lAxis : array [0..1] of LongInt;
  dPos : array [0..1] of Double;
  uShortDistance : DWord;
  lCoordinate : LongInt;
  lPosSize : LongInt;
  dRadius : Double;

begin

  dVelocity := 200;
  dAccel := 400;
  lPosSize := 2;
  lCoordinate := 0;
  uShortDistance := 0;
  dRadius := 150;

  AxmContiSetAbsRelMode(lCoordinate, 1);

  for i := 0 to 1 do
  begin
    lAxis[i] := i;
  end;

  AxmContiWriteClear(lCoordinate);
  AxmContiSetAxisMap (lCoordinate, 2, @lAxis);

  AxmContiBeginNode(lCoordinate);
  dPos[0] := 100 ; dPos[1] := 100;
  AxmLineMove(lCoordinate, @dPos, dVelocity, dAccel, dAccel);
  dPos[0] := 200 ; dPos[1] := 200;
  AxmLineMove(lCoordinate, @dPos, dVelocity, dAccel, dAccel);
  dPos[0] := 300 ; dPos[1] := 0;
  AxmCircleRadiusMove(lCoordinate, @lAxis, dRadius, @dPos, dVelocity,
    dAccel, dAccel, DIR_CCW, uShortDistance);
  dPos[0] := 400 ; dPos[1] := 400;

```

```
AxmLineMove(lCoordinate, @dPos, dVelocity, dAccel, dAccel);
dPos[0] := 300 ; dPos[1] := 0;
AxmCircleRadiusMove(lCoordinate, @lAxis, dRadius, @dPos, dVelocity,
    dAccel, dAccel, DIR_CCW, uShortDistance);
dPos[0] := 600 ; dPos[1] := 600;
AxmLineMove(lCoordinate, @dPos, dVelocity, dAccel, dAccel);
dPos[0] := 700 ; dPos[1] := 700;
AxmLineMove(lCoordinate, @dPos, dVelocity, dAccel, dAccel);

AxmContiEndNode (lCoordinate);

AxmMotSetAbsRelMode(lCoordinate, 1);

AxmContiStart (lCoordinate, 0, 0);
end;
```

See Also

[AxmLineMove](#), [AxmCircleCenterMove](#), [AxmCircleRadiusMove](#), [AxmCircleAngleMove](#),
[AxmContiSetAxisMap](#), [AxmContiGetAxisMap](#), [AxmContiSetAbsRelMode](#),
[AxmContiGetAbsRelMode](#), [AxmContiBeginNode](#), [AxmContiEndNode](#), [AxmContiReadFree](#),
[AxmContiReadIndex](#), [AxmContiWriteClear](#), [AxmContiStart](#), [AxmContiIsMotion](#),
[AxmContiGetNodeNum](#), [AxmContiGetTotalNodeNum](#), [AxmContiSetOptionNodeNum](#)

AxmCircleRadiusMove

Purpose

To process circular interpolation move by specifying the start point, end point, and radius on the specified coordination. It exits the API after move starts.

Format

C

```
DWORD AxmCircleRadiusMove (long lCoord, long *lAxisNo, double dRadius, double *dEndPos, double dVel, double
dAccel, double dDecel, DWORD uCWDdir, 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 uCWDdir 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; uCWDdir : DWord; uShortDistance : DWord) : DWord; stdcall;
```

Input / Output

Name	in/out	Format	Init Value	Explanation
Coord	in	long		Coordinate system number(0~ (max. axis number -1))
AxisNo	in	long*		Channe(axis)l number(0~ (max. axis number -1)) array used for interpolation
Radius	In	double		Circular radius to move
EndPos	in	double		Array of X axis end point and Y axis end point
Vel	in	double		Move speed(+direction:more than 0, -directoin: less than 0, move speed unit is PPS[Pulses/sec] when Unit/pulse is 1/1)
Accel	in	double		Move acceleration(acceleration unit is PPS[Pulses/sec^2] when Unit/pulse is 1/1)
Decel	in	double		Move deceleration(deceleration unit is PPS[Pulses/sec^2] when Unit/pulse is 1/1)
CWDdir	in	DWORD		Circular direction
ShortDistance	in	DWORD		Set value of moving distance of circle to the middle point

CWDir

#define	Value	Explanation
DIR_CCW	00h	Counterclockwise direction
DIR_CW	01h	Clockwise direction

ShortDistance

#define	Value	Explanation
SHORT_DISTANCE	00h	Circular Move in short distance <a in the picture>
LONG_DISTANCE	01h	Circular Move in long distance <b in the picture>

Return AXT_RT_SUCCESS(0000) : Successful execution of API.

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

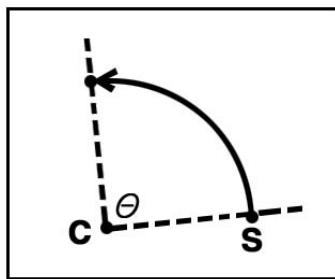
Description

Circular interpolation applies to any two axes. This explanation assumes two mapped axes are X axis and Y axis. X axis means lower axis number and Y axis means higher axis number in two mapped axes. For example, when Z and U axis are the two mapped axes, Z axis corresponds to X axis and U axis corresponds to Y axis.

Like [AxmContiBeginNode](#) and [AxmContiEndNode](#), when it is being used, it becomes store API in queue which moves by specifying middle and end point in the specified coordinate system. For profile circular continuous interpolation move, it starts [AxmContiStart](#) API by storing it in internal queue.

Before executing this API, [AxmContiSetAxisMap](#) API does axis mapping which maps axes to execute interpolation process to axis number. Not only continuous interpolation, but also APIs related to general interpolation control are applied because transmitting parameter is simple. Thus, before using APIs about general interpolation, axes for future use are mapped by this API.

Current position becomes the start point of circular move and it executes circular move based on set radius and end point. If define circular with certain radius connection start point and end point, two center points are appeared; with this, two paths (a, b) can be defined even if it is same rotation direction.



► Caution: After axis mapping using [AxmContiSetAxisMap](#), use from the lowest axis mapping. In case of circular interpolation, it is possible to move only when array it from the lowest axis.

► Caution: If signal of +LIMIT, -LIMIT is on, there's no move.

- ▶ By AxmSStop and AxmEstop API, motion deceleration stop and motion emergency stop are possible. Argument is always stopped by transmitting the first axis(the lowest actual axis number) of coordinate.
- ▶ [AxmContiIsMotion](#) API can verify whether it is on motion or not.
 - ▶ When using [AxmSignalSetInpos](#) API, if Inpos is set to Enable, it doesn't return until INP input is set to ON although command pulse output is completed because it considers move is not completed. Refer to [AxmSignalSetInpos](#) API for further information. **
 - ▶ Caution: Axes which interpolate Unit/Pulse should be equally set. If Unit/Pulse is not equally set, their unit will be different with each other. Always set them equally, and do interpolation control.
 - ▶ Caution: When using motion board with more than 2 axes (SMC-2V03)
In the board with 8 axes by using SMC-2V03, two axes from Axis 0–1 become axis group 1, two from Axis 2–3 become axis group 2, two from Axis 4–5 become axis group 3, and Axis6–7 become axis group 4. Axes from the same group can process line interpolation move and circular interpolation move, but axes from different groups cannot. Keep in mind when specifying coordinate.
 - ▶ Caution: When using motion board with more than 8 (PCI-N804/404)
In the board with multi axes by using PCI-N804/404, 4 axes from Axis 0–3 become Axis group 1, 4 from Axis 4–7 become Axis group 2, and another 4 from Axis8–11 become Axis group 3. Axes from the same group can process line interpolation move and circular interpolation move, but axes from different groups cannot. Keep in mind when specifying coordinate.

▶ List of continuous interpolation API

Function	Description	Note
AxmContiSetAxisMap	Set continuous interpolation axis mapping for specified coordinate system.	
AxmContiGetAxisMap	Return continuous interpolation axis mapping for specified coordinate system.	
AxmContiSetAbsRelMode	Set continuous interpolation axis absolute/relative mode for specified coordinate system..	
AxmContiGetAbsRelMode	Return continuous interpolation axis absolute/relative mode for specified coordinate system.	
AxmContiBeginNode	Start registering tasks which will be processed in continuous interpolation for specified coordinate system. After calling this API, all motion tasks processed before calling AxmContiEndNode API are registered as continuous interpolation motion, but not processing real motion. When AxmContiStart	

	API is called, registered motion is actually started.	
AxmContiEndNode	End registering tasks to process continuous interpolation in specified coordinate system.	
AxmContiReadFree	Identify internal queue if it is empty for interpolation move in specified coordinate system.	
AxmContiReadIndex	Identify the number of interpolation move in internal queue stored for interpolation move in specified coordinate system.	
AxmContiWriteClear	Clear all of the internal queue stored for continuous interpolation move in specified coordinate system.	
AxmContiStart	Start continuous interpolation move from internal queue stored in specified coordinate system.	
AxmContiIsMotion	Identify countinuous interpolation move if it is running or not in specified coordinate system.	
AxmContiGetNodeNum	Identify the index number of currently running continuous interpolation among continuous interpolation moves in specified coordinate system.	
AxmContiGetTotalNodeNum	Identify DO output, interrupt, and trigger ouptput at certain position during continuous interpolation move set in specified coordinate system.	

Example of General Interpolation Move>

C Example

```

long lAxis[2];
double dCenPos[2];
double dAngle=90;
DWORD uCWDDir=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;

```

```
AxmContiWriteClear(lCoordinate);
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

AxmContiWriteClear lCoordinate
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;

  AxmContiWriteClear(lCoordinate);
  AxmContiSetAxisMap (lCoordinate, lPosSize, @lAxis);
  AxmContiSetAbsRelMode(lCoordinate, 1);
  AxmCircleAngleMove(lCoordinate, @lAxis, @dCenPos, dAngle, dVelocity,
    dAccel, dAccel, 1);
```

```
end;
```

Example of Continuous Interpolation Move>

Start API of axis 2 circular interpolation move.

Current position becomes the start point of circular move and circular move is processed based on radius and end point.

C Example

```
long    lAxis[2];
long    lPosSize = 2;
long    lCoordinate = 0;
double  dCenPos[2];
double  dAngle = 180;
double  dPos[2];
double  dVelocity = 200;
double  dAccel = 400;

for(int i=0; i<lPosSize; i++)
{
    lAxis[i] = i;
}

AxmContiWriteClear(lCoordinate);
AxmContiSetAxisMap (lCoordinate, lPosSize, lAxis);
AxmContiSetAbsRelMode(lCoordinate, 1);

AxmContiBeginNode(lCoordinate);
dPos[0] = 100 , dPos[1] = 100;
AxmLineMove(lCoordinate, dPos, dVelocity, dAccel, dAccel);
dPos[0] = 200 , dPos[1] = 200;
AxmLineMove(lCoordinate, dPos, dVelocity, dAccel, dAccel);
dCenPos[0] = 300 , dCenPos[1] = 300;
AxmCircleAngleMove(lCoordinate, lAxis, dCenPos, dAngle, dVelocity, dAccel,
                   dAccel, DIR_CCW);
dPos[0] = 400 , dPos[1] = 400;
AxmLineMove(lCoordinate, dPos, dVelocity, dAccel, dAccel);
dCenPos[0] = 500 , dCenPos[1] = 500;
AxmCircleAngleMove(lCoordinate, lAxis, dCenPos, dAngle, dVelocity, dAccel,
                   dAccel, DIR_CCW);
dPos[0] = 600 , dPos[1] = 600;
AxmLineMove(lCoordinate, dPos, dVelocity, dAccel, dAccel);
dPos[0] = 700 , dPos[1] = 700;
AxmLineMove(lCoordinate, dPos, dVelocity, dAccel, dAccel);
AxmContiEndNode (lCoordinate);

AxmContiSetAbsRelMode(lCoordinate, 1);
AxmContiStart (lCoordinate, 0, 0);
```

VB Example

```
Dim dVelocity As Double
Dim dAccel As Double
Dim lPosSize As Long
Dim lCoordinate As Long
Dim dPos(0 To 1) As Double
Dim dCenPos(0 To 1) As Double
Dim dAngle As Double
Dim axes(0 To 1) As Long

Dim i As Long
```

```

lPosSize = 2
lCoordinate = 0
dAngle = 180

For i = 0 To 1
    axes(i) = i
Next i

dVelocity = 200
dAccel = 400

AxmContiWriteClear lCoordinate
AxmContiSetAxisMap lCoordinate, lPosSize, axes(0)
AxmContiSetAbsRelMode lCoordinate, 1

AxmContiBeginNode lCoordinate
dPos(0) = 100: dPos(1) = 100
AxmLineMove lCoordinate, dPos(0), dVelocity, dAccel, dAccel
dPos(0) = 200: dPos(1) = 200
AxmLineMove lCoordinate, dPos(0), dVelocity, dAccel, dAccel
dPos(0) = 300: dPos(1) = 300
AxmLineMove lCoordinate, dPos(0), dVelocity, dAccel, dAccel
dCenPos(0) = 300 : dCenPos(1) = 300
AxmCircleAngleMove lCoordinate, axes(0), dCenPos(0), dAngle, dVelocity,
    dAccel, dAccel, DIR_CCW
dPos(0) = 400: dPos(1) = 400
AxmLineMove lCoordinate, dPos(0), dVelocity, dAccel, dAccel
dCenPos(0) = 500 : dCenPos(1) = 500
AxmCircleAngleMove lCoordinate, axes(0), dCenPos(0), dAngle, dVelocity,
    dAccel, dAccel, DIR_CCW
dPos(0) = 500: dPos(1) = 500
AxmLineMove lCoordinate, dPos(0), dVelocity, dAccel, dAccel
dPos(0) = 600: dPos(1) = 600
AxmLineMove lCoordinate, dPos(0), dVelocity, dAccel, dAccel
dPos(0) = 700: dPos(1) = 700
AxmLineMove lCoordinate, dPos(0), dVelocity, dAccel, dAccel
AxmContiEndNode lCoordinate

AxmContiSetAbsRelMode lCoordinate, 1
AxmContiStart lCoordinate, 0, 0

```

Delphi Example

```

var
  i : LongInt;
  dVelocity, dAccel : Double;
  lAxis : array [0..1] of LongInt;
  dPos : array [0..1] of Double;
  dCenPos : array [0..1] of Double;
  dAngle : Double;
  lCoordinate : LongInt;
  lPosSize : LongInt;

begin

  dVelocity := 200;
  dAccel := 400;
  lPosSize := 2;
  lCoordinate := 0;
  dAngle := 180;

  for i := 0 to 1 do
  begin
    lAxis[i] := i;
  end;

```

```
AxmContiWriteClear(lCoordinate);
AxmContiSetAxisMap (lCoordinate, 2, @lAxis);
AxmContiSetAbsRelMode(lCoordinate, 1);

AxmContiBeginNode(lCoordinate);
dPos[0] := 100 ; dPos[1] := 100;
AxmLineMove(lCoordinate, @dPos, dVelocity, dAccel, dAccel);
dPos[0] := 200 ; dPos[1] := 200;
AxmLineMove(lCoordinate, @dPos, dVelocity, dAccel, dAccel);
dCenPos [0] := 300; dCenPos[1] := 300;
AxmCircleAngleMove(lCoordinate, @lAxis, @dCenPos, dAngle, dVelocity,
    dAccel, dAccel, DIR_CCW);
dPos[0] := 400 ; dPos[1] := 400;
AxmLineMove(lCoordinate, @dPos, dVelocity, dAccel, dAccel);
dCenPos[0] := 500; dCenPos[1] := 500;
AxmCircleAngleMove(lCoordinate, @lAxis, @dCenPos, dAngle, dVelocity,
    dAccel, dAccel, DIR_CCW);
AxmLineMove(lCoordinate, @dPos, dVelocity, dAccel, dAccel);
dPos[0] := 600 ; dPos[1] := 600;
AxmLineMove(lCoordinate, @dPos, dVelocity, dAccel, dAccel);
dPos[0] := 700 ; dPos[1] := 700;
AxmLineMove(lCoordinate, @dPos, dVelocity, dAccel, dAccel);
dPos[0] := 800 ; dPos[1] := 800;
AxmContiEndNode (lCoordinate);

AxmMotSetAbsRelMode(lCoordinate, 1);

AxmContiStart (lCoordinate, 0, 0);
end;
```

See Also

[AxmLineMove](#), [AxmCircleCenterMove](#), [AxmCirclePointMove](#), [AxmCircleAngleMove](#),
[AxmContiSetAxisMap](#), [AxmContiGetAxisMap](#), [AxmContiSetAbsRelMode](#),
[AxmContiGetAbsRelMode](#), [AxmContiBeginNode](#), [AxmContiEndNode](#), [AxmContiReadFree](#),
[AxmContiReadIndex](#), [AxmContiWriteClear](#), [AxmContiStart](#), [AxmContiIsMotion](#),
[AxmContiGetNodeNum](#), [AxmContiGetTotalNodeNum](#), [AxmContiSetOptionNodeNum](#)

AxmCircleAngleMove

Purpose

To process circular interpolation move by specifying the center point and angle on the specified coordination. It exits the API after move starts.

Format

C

```
DWORD AxmCircleAngleMove(long lCoord, long *lAxisNo, double *dCenterPos, double dAngle, double dVel, double
dAccel, double dDecel, DWORD uCWDdir);
```

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 uCWDdir As Long)
As Long
```

Delphi

```
function AxmCircleAngleMove (lCoord : LongInt; lAxisNo : PLongInt; dCenterPos: PDouble; dAngle : Double; dVel :
Double; dAccel : Double; dDecel : Double; uCWDdir : DWord) : DWord; stdcall;
```

Input / Output

Name	in/out	Format	Init Value	Explanation
Coord	in	long		Coordinate system number(start from 0)
AxisNo	in	long*		Channel number(start from 0) array used for interpolation
Radius	In	double		Circular radius to move
EndPos	in	double		Array of X axis end point and Y axis end point
Vel	in	double		Move speed(+direction:more than 0, – directoin: less than 0, move speed unit is PPS[Pulses/sec] when Unit/pulse is 1/1)
Accel	in	double		Move acceleration(acceleration unit is PPS[Pulses/sec^s] when Unit/pulse is 1/1)
Decel	in	double		Move deceleration(deceleration unit is PPS[Pulses/sec^2] when Unit/pulse is 1/1)
CWDir	in	DWORD		Circular direction

CWDir

#define	Value	Explanation
DIR_CCW	00h	Counterclockwise direction

DIR_CW	01h	Clockwise direction
--------	-----	---------------------

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

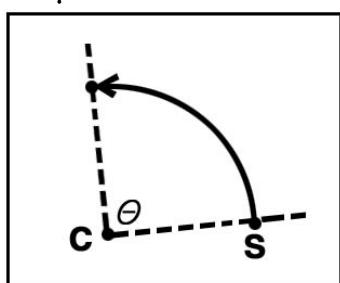
Description

Circular interpolation applies to any two axes. This explanation assumes two mapped axes are X axis and Y axis. X axis means lower axis number and Y axis means higher axis number in two mapped axes. For example, when Z and U axis are the two mapped axes, Z axis corresponds to X axis and U axis corresponds to Y axis.

Like [AxmContiBeginNode](#) and [AxmContiEndNode](#), when it is being used, it becomes store API in queue which moves by specifying middle and end point in the specified coordinate system. For profile circular continuous interpolation move, it starts [AxmContiStart](#) API by storing it in internal queue.

Before executing this API, [AxmContiSetAxisMap](#) API does axis mapping which maps axes to execute interpolation process to axis number. Not only continuous interpolation, but also APIs related to general interpolation control are applied because transmitting parameter is simple. Thus, before using APIs about general interpolation, axes for future use are mapped by this API.

Current position becomes the start point of circular move and it executes circular move by the angle based on end point.



- ▶ Caution: After axis mapping using [AxmContiSetAxisMap](#), use from the lowest axis mapping. In case of circular interpolation, it is possible to move only when array it from the lowest axis.
- ▶ Caution: If signal of +LIMIT, -LIMIT is on, there's no move.
- ▶ By AxmSStop and AxmEstop API, motion deceleration stop and motion emergency stop are possible. Argument is always stopped by transmitting the first axis(the lowest actual axis number) of coordinate.
- ▶ [AxmContiIsMotion](#) API can verify whether it is on motion or not.
- ▶ When using [AxmSignalSetInpos](#) API, if Inpos is set to Enable, it doesn't return until INP input is set to ON although command pulse output is completed because it considers move is not completed. Refer to [AxmSignalSetInpos](#) API for further information. **
- ▶ Caution: Axes which interpolate Unit/Pulse should be equally set. If Unit/Pulse is not equally set, their unit will be different with each other. Always set them equally, and do interpolation control.
- ▶ Caution: When using motion board with more than 2 axes (SMC-2V03)

In the board with 8 axes by using SMC-2V03, two axes from Axis 0–1 become axis group 1, two from Axis 2–3 become axis group 2, two from Axis 4–5 become axis group 3, and Axis 6–7 become axis group 4. Axes from the same group can process line interpolation move and circular interpolation move, but axes from different groups cannot. Keep in mind when specifying coordinate.

► Caution: When using motion board with more than 8 (PCI-N804/404)

In the board with multi axes by using PCI-N804/404, 4 axes from Axis 0–3 become Axis group 1, 4 from Axis 4–7 become Axis group 2, and another 4 from Axis 8–11 become Axis group 3. Axes from the same group can process line interpolation move and circular interpolation move, but axes from different groups cannot. Keep in mind when specifying coordinate.

► List of continuous interpolation API

Function	Description	Note
AxmContiSetAxisMap	Set continuous interpolation axis mapping for specified coordinate system.	
AxmContiGetAxisMap	Return continuous interpolation axis mapping for specified coordinate system.	
AxmContiSetAbsRelMode	Set continuous interpolation axis absolute/relative mode for specified coordinate system..	
AxmContiGetAbsRelMode	Return continuous interpolation axis absolute/relative mode for specified coordinate system.	
AxmContiBeginNode	Start registering tasks which will be processed in continuous interpolation for specified coordinate system. After calling this API, all motion tasks processed before calling AxmContiEndNode API are registered as continuous interpolation motion, but not processing real motion. When AxmContiStart API is called, registered motion is actually started.	
AxmContiEndNode	End registering tasks to process continuous interpolation in specified coordinate system.	
AxmContiReadFree	Identify internal queue if it is empty for interpolation move in specified coordinate system.	
AxmContiReadIndex	Identify the number of interpolation move in	

	internal queue stored for interpolation move in specified coordinate system.	
<u>AxmContiWriteClear</u>	Clear all of the internal queue stored for continuous interpolation move in specified coordinate system.	
<u>AxmContiStart</u>	Start continuous interpolation move from internal queue stored in specified coordinate system.	
<u>AxmContiIsMotion</u>	Identify countinuous interpolation move if it is running or not in specified coordinate system.	
<u>AxmContiGetNodeNum</u>	Identify the index number of currently running continuous interpolation among continuous interpolation moves in specified coordinate system.	
<u>AxmContiGetTotalNodeNum</u>	Identify total number of index of continuous interpolation move set in specified coordinate system.	

Example of General Interpolation Move>

C Example

```
long lAxis[2];
double dCenPos[2];
double dAngle=90;
DWORD uCWDDir=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;

AxmContiWriteClear(lCoordinate);
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

AxmContiWriteClear lCoordinate
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;

  AxmContiWriteClear(lCoordinate);
  AxmContiSetAxisMap (lCoordinate, lPosSize, @lAxis);
  AxmContiSetAbsRelMode(lCoordinate, 1);
  AxmCircleAngleMove(lCoordinate, @lAxis, @dCenPos, dAngle, dVelocity,
    dAccel, dAccel, 1);
end;

```

Example of Continuous Interpolation Move>

Start API of axis 2 circular interpolation move.

Current position becomes the start point of circular move and circular move is processed by the angle based on the end.

C Example

```

long   lAxis[2];
long   lPosSize = 2;
long   lCoordinate = 0;
double dCenPos[2];

```

```

double dAngle = 180;
double dPos[2];
double dVelocity = 200;
double dAccel = 400;

for(int i=0; i<lPosSize; i++){
    lAxis[i] = i;
}

AxmContiWriteClear(lCoordinate);
AxmContiSetAxisMap (lCoordinate, lPosSize, lAxis);
AxmContiSetAbsRelMode(lCoordinate, 1);

AxmContiBeginNode(lCoordinate);
dPos[0] = 100 , dPos[1] = 100;
AxmLineMove(lCoordinate, dPos, dVelocity, dAccel, dAccel);
dPos[0] = 200 , dPos[1] = 200;
AxmLineMove(lCoordinate, dPos, dVelocity, dAccel, dAccel);
dCenPos[0] = 300 , dCenPos[1] = 300;
AxmCircleAngleMove(lCoordinate, lAxis, dCenPos, dAngle, dVelocity, dAccel,
    dAccel, DIR_CCW);
dPos[0] = 400 , dPos[1] = 400;
AxmLineMove(lCoordinate, dPos, dVelocity, dAccel, dAccel);
dCenPos[0] = 500 , dCenPos[1] = 500;
AxmCircleAngleMove(lCoordinate, lAxis, dCenPos, dAngle, dVelocity, dAccel,
    dAccel, DIR_CCW);
dPos[0] = 600 , dPos[1] = 600;
AxmLineMove(lCoordinate, dPos, dVelocity, dAccel, dAccel);
dPos[0] = 700 , dPos[1] = 700;
AxmLineMove(lCoordinate, dPos, dVelocity, dAccel, dAccel);
AxmContiEndNode (lCoordinate);
AxmContiSetAbsRelMode(lCoordinate, 1);
AxmContiStart (lCoordinate, 0, 0);

```

VB Example

```

Dim dVelocity As Double
Dim dAccel As Double
Dim lPosSize As Long
Dim lCoordinate As Long
Dim dPos(0 To 1) As Double
Dim dCenPos(0 To 1) As Double
Dim dAngle As Double
Dim axes(0 To 1) As Long

Dim i As Long
lPosSize = 2
lCoordinate = 0
dAngle = 180

For i = 0 To 1
    axes(i) = i
Next i

dVelocity = 200
dAccel = 400

AxmContiWriteClear lCoordinate
AxmContiSetAxisMap lCoordinate, lPosSize, axes(0)
AxmContiSetAbsRelMode lCoordinate, 1

AxmContiBeginNode lCoordinate
dPos(0) = 100: dPos(1) = 100
AxmLineMove lCoordinate, dPos(0), dVelocity, dAccel, dAccel
dPos(0) = 200: dPos(1) = 200
AxmLineMove lCoordinate, dPos(0), dVelocity, dAccel, dAccel

```

```

dPos(0) = 300: dPos(1) = 300
AxmLineMove lCoordinate, dPos(0), dVelocity, dAccel, dAccel
dCenPos(0) = 300 : dCenPos(1) = 300
AxmCircleAngleMove lCoordinate, axes(0), dCenPos(0), dAngle, dVelocity,
    dAccel, dAccel, DIR_CCW
dPos(0) = 400: dPos(1) = 400
AxmLineMove lCoordinate, dPos(0), dVelocity, dAccel, dAccel
dCenPos(0) = 500 : dCenPos(1) = 500
AxmCircleAngleMove lCoordinate, axes(0), dCenPos(0), dAngle, dVelocity,
    dAccel, dAccel, DIR_CCW
dPos(0) = 500: dPos(1) = 500
AxmLineMove lCoordinate, dPos(0), dVelocity, dAccel, dAccel
dPos(0) = 600: dPos(1) = 600
AxmLineMove lCoordinate, dPos(0), dVelocity, dAccel, dAccel
dPos(0) = 700: dPos(1) = 700
AxmLineMove lCoordinate, dPos(0), dVelocity, dAccel, dAccel
AxmContiEndNode lCoordinate

AxmContiSetAbsRelMode lCoordinate, 1
AxmContiStart lCoordinate, 0, 0

```

Delphi Example

```

var
  i : LongInt;
  dVelocity, dAccel : Double;
  lAxis : array [0..1] of LongInt;
  dPos : array [0..1] of Double;
  dCenPos : array [0..1] of Double;
  dAngle : Double;
  lCoordinate : LongInt;
  lPosSize : LongInt;

begin

  dVelocity := 200;
  dAccel := 400;
  lPosSize := 2;
  lCoordinate := 0;
  dAngle := 180;

  for i := 0 to 1 do
  begin
    lAxis[i] := i;
  end;

  AxmContiWriteClear(lCoordinate);
  AxmContiSetAxisMap (lCoordinate, 2, @lAxis);
  AxmContiSetAbsRelMode(lCoordinate, 1);
  AxmContiBeginNode(lCoordinate);
  dPos[0] := 100 ; dPos[1] := 100;
  AxmLineMove(lCoordinate, @dPos, dVelocity, dAccel, dAccel);
  dPos[0] := 200 ; dPos[1] := 200;
  AxmLineMove(lCoordinate, @dPos, dVelocity, dAccel, dAccel);
  dCenPos [0] := 300; dCenPos[1] := 300;
  AxmCircleAngleMove(lCoordinate, @lAxis, @dCenPos, dAngle, dVelocity,
    dAccel, dAccel, DIR_CCW);
  dPos[0] := 400 ; dPos[1] := 400;
  AxmLineMove(lCoordinate, @dPos, dVelocity, dAccel, dAccel);
  dCenPos[0] := 500; dCenPos[1] := 500;
  AxmCircleAngleMove(lCoordinate, @lAxis, @dCenPos, dAngle, dVelocity,
    dAccel, dAccel, DIR_CCW);
  AxmLineMove(lCoordinate, @dPos, dVelocity, dAccel, dAccel);
  dPos[0] := 600 ; dPos[1] := 600;
  AxmLineMove(lCoordinate, @dPos, dVelocity, dAccel, dAccel);
  dPos[0] := 700 ; dPos[1] := 700;

```

```
AxmLineMove(lCoordinate, @dPos, dVelocity, dAccel, dAccel);
dPos[0] := 800 ; dPos[1] := 800;
AxmContiEndNode (lCoordinate);

AxmMotSetAbsRelMode(lCoordinate, 1);

AxmContiStart (lCoordinate, 0, 0);
end;
```

See Also

[AxmLineMove](#), [AxmCircleCenterMove](#), [AxmCirclePointMove](#), [AxmCircleRadiusMove](#), [AxmContiSetAxisMap](#),
[AxmContiGetAxisMap](#), [AxmContiSetAbsRelMode](#), [AxmContiGetAbsRelMode](#), [AxmContiBeginNode](#),
[AxmContiEndNode](#), [AxmContiReadFree](#), [AxmContiReadIndex](#), [AxmContiWriteClear](#),
[AxmContiStart](#), [AxmContiIsMotion](#), [AxmContiGetNodeNum](#), [AxmContiGetTotalNodeNum](#),
[AxmContiSetOptionNodeNum](#)

Continuous interpolation setting and move API

Continuous interpolation move API processes sequence of interpolation moves previously explained. It initiates and accelerates move, goes thorough specified positions in a steady speed, decelerates again at the end, and stops move.

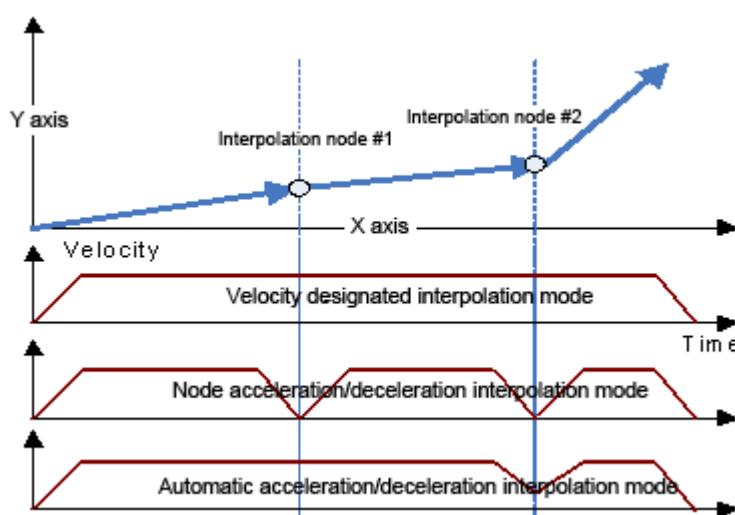
One special thing in continuous interpolation API is that axes to be used in continuous interpolation go through mapping process to new coordinate system, commands of interpolation moves are sequentially saved in memory area called “continuous interpolation queue”, when continuous interpolation initiation command is given, the stored continuous interpolation data comes out and enters into motion control chip, and interpolation moves are continuously processed. In addition, even after interpolation move is started, interpolation data can be entered into the motion control chip.

Function	Description
AxmContiSetAxisMap	Set continuous interpolation axis mapping for specified coordinate system.
AxmContiGetAxisMap	Return continuous interpolation axis mapping for specified coordinate system.
AxmContiSetAbsRelMode	Set continuous interpolation axis absolute/relative mode for specified coordinate system.
AxmContiGetAbsRelMode	Return continuous interpolation axis absolute/relative mode for specified coordinate system.
AxmContiBeginNode	Start registering tasks which will be processed in continuous interpolation for specified coordinate system. After calling this function, all motion tasks processed before calling AxmContiEndNode API are registered as continuous interpolation motion, but not processing real motion. When AxmContiStart API is called, registered motion is actually started.
AxmContiEndNode	End registering tasks to process continuous interpolation in specified coordinate system.
AxmContiReadFree	Identify internal queue is empty for interpolation move in specified coordinate system.
AxmContiReadIndex	Identify the number of interpolation move in internal queue stored for interpolation move in specified coordinate system.
AxmContiWriteClear	Clear all of the internal queue stored for continuous interpolation move in a specified coordinate system.
AxmContiStart	Start continuous interpolation move from internal queue stored in

	a specified coordinate system.
AxmContiIsMotion	Identify continuous interpolation move is running or not in specified coordinate system.
AxmContiGetNodeNum	Identify the index number of currently running continuous interpolation among continuous interpolation moves in specified coordinate system.
AxmContiGetTotalNodeNum	Identify total number of index of continuous interpolation move set in specified coordinate system.

To process continuous interpolation move, axes to be used in continuous interpolation should be mapped. Interpolation move API used in continuous interpolation is combination of axis 2 interpolation move, but when mapping axes, all axes to be used should be mapped in one coordinate system which includes all axes. For example, if you are using three axes 0,1,2 and want to process axis 0,1 interpolation move and then process axis 1,2 interpolation move, you must map axes 0,1,2 into one coordinate system. Once mapping to coordinate system is completed, you should set speed profile to be used in specified coordinate system, and select absolute/relative move.

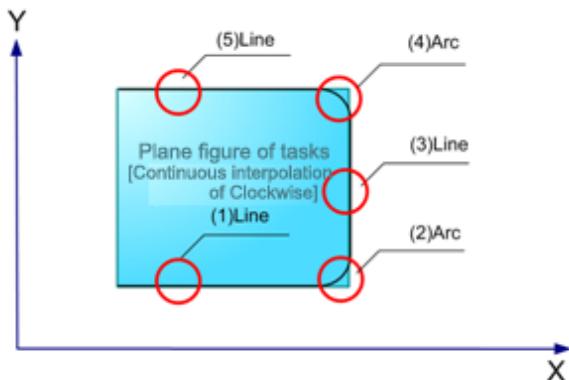
After entering data into interpolation queue, calling [AxmContiStart](#) API will start continuous interpolation move using stored data. Here, you can set speed profile mode during continuous interpolation move. There are speed specification interpolation mode and node ac/deceleration interpolation mode, and automatic ac/deceleration interpolation mode.



Speed specification interpolation mode means general continuous interpolation move in which move started, speed is accelerated, and when it reaches certain speed, it is maintained during continuous interpolation move. Node ac/deceleration interpolation mode means speed is ac/decelerated in every node. Auto ac/deceleration interpolation mode does not have speed change when it goes through continuous path, but it ac/decelerates automatically in emergency curve.

Description of continuous interpolation

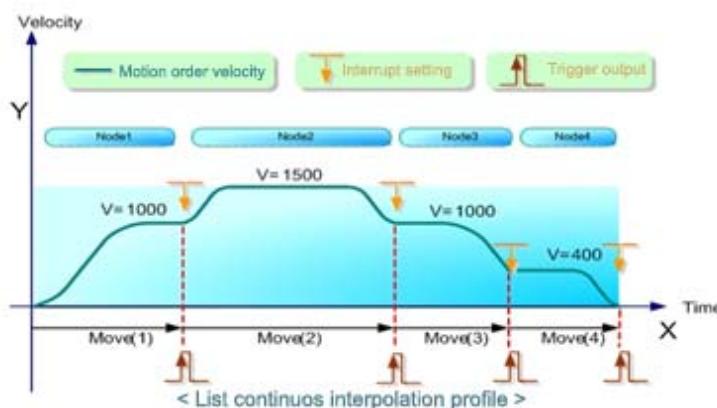
List continuous interpolation motion used here is the function in which several steps of task are registered and processed collectively. Advantage of continuous interpolation motion is continuous execution of tasks without interruption of profile between tasks. Generally, in equipment, by inserting circular arc to the corner where two lines meet (such as dispensing, manufacturing, and painting), continuous control of line → circular arc → line → circular arc → line interpolation is possible.



Generally, you can use steady speed in list continuous interpolation motion, but you can also change profile by applying various speed to move nodes, which is one of the benefits of continuous interpolation move. Processing task continuously without interruption between tasks, you can change speed of tasking for certain point.

Option function

* Notice: In each node, you can specify interrupt. The reason for setting interrupt is that when you want to output DO when one node is completed while motion is being processed, you can specify interrupt in which you can output in the service routine, or specifying other interrupts to be used in motion of other axes. In addition, you can set to doing role of vision trigger when using vision while motioning.



AxmContiSetAxisMap

Purpose

Set axis mapping in which axes to be used to process interpolation tasks to specified coordinate system.

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		Coordinate system number (start from 0)
lSize	In	long		Number of axes used in interpolation (IP: 2, QI : 2 – 4)
RealAxisNo	In	long*		The axis array used in interpolation IP: only 2 axes are possible QI:set 2 – 4 axes Axis number to be set(0,1,2,3) Cf.) setting is possible within one chip, (0,1,2,3) setting possible (4,5,6,7) setting possible

lSize

#define	Value	Explanation
INTERPOLATION_AXIS2	02h	When axis2 is used in interpolation
INTERPOLATION_AXIS3	03h	When axis3 is used in interpolation
INTERPOLATION_AXIS4	04h	When axis4 is used in interpolation

Return AXT_RT_SUCCESS(0000) : Successful execution of API.

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

Description

Not only continuous interpolation, but also APIs related to general interpolation control are applied because transmitting parameter is simple. . Thus, before using APIs related to general interpolation, axes for future use are mapped by this API.

Caution: Always put smaller number first when axis mapping.

Here, the smallest axis becomes a master.

►Caution: After axis mapping using [AxmContiSetAxisMap](#) API, map from the lowest axis. In case of circular interpolation, always put it to axis array from the lowest axis.

►Caution: When using motion board with more than 2 axes (SMC-2V03)

In the board with 8 axes by using SMC-2V03, two axes from Axis 0–1 become axis group 1, two from Axis 2–3 become axis group 2, two from Axis 4–5 become axis group 3, and Axis 6–7 become axis group 4. Axes from the same group can process line interpolation move and circular interpolation move, but axes from different groups cannot. Keep in mind when specifying coordinate.

►Caution: When using motion board with more than 8 (PCI-N804/404)

In the board with multi axes by using PCI-N804/404, 4 axes from Axis 0–3 become Axis group 1, 4 from Axis 4–7 become Axis group 2, and another 4 from Axis 8–11 become Axis group 3. Axes from the same group can process line interpolation move and circular interpolation move, but axes from different groups cannot. Keep in mind when specifying coordinate.

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);
// relative position move
AxmContiSetAbsRelMode(lCoordinate, 1);
// line interpolation move
AxmLineMove(lCoordinate, dPos, 200, 400, 400);

long lAxis1, lAxis2;
AxmContiGetAxisMap(lCoordinate, &lPosSize, lAxis);
// You can identify the number of axes using lPosSize.
lAxis1 = lAxis[0]; lAxis2 = lAxis[1]; // bring mapping axis number.
```

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)
  'relative position move
AxmContiSetAbsRelMode lCoordinate, 1
  'line interpolation move
AxmLineMove lCoordinate, Distance(0), 200, 400, 400

AxmContiGetAxisMap lCoordinate, 2, axes (0)
  // You can identify the number of axes using lPosSize.
lAxis1 = axes (0); lAxis2 = axes (1)  'bring mapping axis number.

```

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;

AxmContiWriteClear(lCoordinate);
AxmContiSetAxisMap (lCoordinate, lPosSize, @lAxis);

{ relative position move }
AxmContiSetAbsRelMode(lCoordinate, 1);

{ line interpolation move }
AxmLineMove(lCoordinate, @dPos, 200, 400, 400);

AxmContiGetAxisMap(lCoordinate, @lPos, lAxis);
{ You can identify the number of axes using lPosSize.}
lAxis1 := lAxis[0]; lAxis2 := lAxis[1]; // bring mapping axis number.

end;

```

See Also

[AxmContiGetAxisMap](#), [AxmContiSetAbsRelMode](#), [AxmContiGetAbsRelMode](#), [AxmContiBeginNode](#),
[AxmContiEndNode](#), [AxmContiReadFree](#), [AxmContiReadIndex](#), [AxmContiWriteClear](#), [AxmContiStart](#),
[AxmContiIsMotion](#), [AxmContiGetNodeNum](#), [AxmContiGetTotalNodeNum](#), [AxmContiSetOptionNodeNum](#)

AxmContiGetAxisMap

Purpose

Return axes to be used in interpolation process with axis mapping to user specified coordinate system.

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		Coordinate system number (start from 0)
lSize	In	long		Number of axes used in interpolation (IP: 2, QI : 2 – 4)
RealAxisNo	out	long*		The axis array used in interpolation IP: only 2 axes are possible QI:set 2 – 4 axes Axis number to be set(0,1,2,3) Cf.) setting is possible within one chip, (0,1,2,3) setting possible (4,5,6,7) setting possible

lSize

#define	Value	Explanation
INTERPOLATION_AXIS2	02h	When axis2 is used in interpolation
INTERPOLATION_AXIS3	03h	When axis3 is used in interpolation
INTERPOLATION_AXIS4	04h	When axis4 is used in interpolation

Return AXT_RT_SUCCESS(0000) : Successful execution of API.

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

Description

Return axes to process interpolation tasks set by '[AxmContiSetAxisMap](#)' with mapping to specified coordinate system.

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);
//relative position move
AxmContiSetAbsRelMode (lCoordinate, 1);
//line interpolation move
AxmLineMove(lCoordinate, dPos, 200, 400, 400);

long lAxis1, lAxis2;
AxmContiGetAxisMap(lCoordinate, & lPosSize, lAxis);
// You can verify the number of axes by lPosSize.
lAxis1 = lAxis[0]; lAxis2 = lAxis[1]; // bring the mapping axis number.

```

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)
'relative position move
AxmContiSetAbsRelMode lCoordinate, 1
' line interpolation move
AxmLineMove lCoordinate, Distance(0), 200, 400, 400

AxmContiGetAxisMap lCoordinate, 2, axes (0)
' You can verify the number of axes by lPos.
lAxis1 = axes (0): lAxis2 = axes (1) 'bring the mapping axis number.

```

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;

AxmContiWriteClear(lCoordinate);
AxmContiSetAxisMap (lCoordinate, lPosSize, @lAxis);
{ relative position move }
AxmContiSetAbsRelMode(lCoordinate, 1);

{ line interpolation move      }
AxmLineMove(lCoordinate, @dPos, 200, 400, 400);

AxmContiGetAxisMap(lCoordinate, @lPos, lAxis);
{ You can verify the number of axes by lPos.}
lAxis1 := lAxis[0]; lAxis2 := lAxis[1]; // bring the mapping axis number.
end;
```

See Also

[AxmContiSetAxisMap](#), [AxmContiGetAxisMap](#), [AxmContiSetAbsRelMode](#), [AxmContiGetAbsRelMode](#),
[AxmContiBeginNode](#), [AxmContiEndNode](#), [AxmContiReadFree](#), [AxmContiReadIndex](#), [AxmContiWriteClear](#),
[AxmContiStart](#), [AxmContiIsMotion](#), [AxmContiGetNodeNum](#), [AxmContiGetTotalNodeNum](#),
[AxmContiSetOptionNodeNum](#)

AxmContiSetAbsRelMode

Purpose

Set moving distance calculation mode of specified coordinate system.

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		Coordinate system number(start from 0)
AbsRelMode	In	DWORD		Moving distance calculation mode

uAbsRelMode

#define	Value	Explanation
POS_ABS_MODE	00h	Position move absolute mode
POS_REL_MODE	01h	Position move relative mode

Return AXT_RT_SUCCESS(0000) : Successful execution of API.

[* See error code Table for more information on status error codes](#)

Description

Set the same moving distance calculation mode to all axes in coordinate system.

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);
//relative position move
AxmContiSetAbsRelMode(lCoordinate, 1);
// line interpolation move
AxmLineMove(lCoordinate, dPos, 200, 400, 400);
```

```
// verify moving distance mode of axis 0.
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)
'relative position move
AxmContiSetAbsRelMode lCoordinate, 1
'line interpolation move
AxmLineMove lCoordinate, Distance(0), 200, 400, 400
'verify moving distance mode of axis 0
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);

  { relative position move }
  AxmContiSetAbsRelMode(lCoordinate, 1);
  { line interpolation move }
  AxmLineMove(lCoordinate, @dPos, 200, 400, 400);
  { verify moving distance mode of axis 0 }
  AxmContiGetAbsRelMode (lCoordinate, @Mode);
end;
```

See Also

[AxmContiSetAxisMap](#), [AxmContiGetAxisMap](#), [AxmContiGetAbsRelMode](#), [AxmContiBeginNode](#),
[AxmContiEndNode](#), [AxmContiReadFree](#), [AxmContiReadIndex](#), [AxmContiWriteClear](#), [AxmContiStart](#),
[AxmContiIsMotion](#), [AxmContiGetNodeNum](#), [AxmContiGetTotalNodeNum](#), [AxmContiSetOptionNodeNum](#)

AxmContiGetAbsRelMode

Purpose

Return moving distance calculation mode of specified coordinate system.

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		Coordinate system number (start from 0)
AbsRelMode	out	DWORD*		Moving distance calculation mode

uAbsRelMode

#define	Value	Explanation
POS_ABS_MODE	00h	Position move absolute mode
POS_REL_MODE	01h	Position move relative mode

Return AXT_RT_SUCCESS(0000) : Successful execution of API.

[* See error code Table for more information on status error codes](#)

Description

Return moving distance calculation mode of specified coordinate system set by
['AxmContiSetAbsRelMode'.](#)

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);
// relative position move
AxmContiSetAbsRelMode(lCoordinate, 1);
// line interpolation move
```

```

AxmLineMove(lCoordinate, dPos, 200, 400, 400);

// verify moving distance mode of axis 0.
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)
    'relative position move
AxmContiSetAbsRelMode lCoordinate, 1
    'line interpolation move
    AxmLineMove lCoordinate, Distance(0), 200, 400, 400

    'verify moving distance mode of axis 0
    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);

    { relative position move }
    AxmContiSetAbsRelMode(lCoordinate, 1);

    { line interpolation move }
    AxmLineMove(lCoordinate, @dPos, 200, 400, 400);

```

```
{ verify moving distance mode of axis 0 }
AxmContiGetAbsRelMode (lCoordinate, @Mode);
end;
```

See Also

[AxmContiSetAxisMap](#), [AxmContiGetAxisMap](#), [AxmContiSetAbsRelMode](#),
[AxmContiBeginNode](#), [AxmContiEndNode](#), [AxmContiReadFree](#), [AxmContiReadIndex](#),
[AxmContiWriteClear](#), [AxmContiStart](#), [AxmContiIsMotion](#), [AxmContiGetNodeNum](#),
[AxmContiGetTotalNodeNum](#), [AxmContiSetOptionNodeNum](#)

AxmContiBeginNode

Purpose

To start registering tasks to be used in continuous interpolation in specified coordinate system.

Format

C

```
DWORD AxmContiBeginNode (long lCoord);
```

Visual Basic

```
Function AxmContiBeginNode (ByVal lCoord As Long) As Long
```

Delphi

```
function AxmContiBeginNode (lCoord: LongInt) : DWord; stdcall;
```

Input / Output

Name	in/out	Format	Init Value	Explanation
Coord	in	long		Coordinate system number (start from 0)

Return AXT_RT_SUCCESS(0000) : Successful execution of API.

[* See error code Table for more information on status error codes](#)

Description

After this API is called, all motion processing is not an actual execution but registration as continuous interpolation motion until [AxmContiEndNode](#) API is called. When [AxmContiStart](#) API is called, registered motion will actually be started.

- ▶ [AxmContiWriteClear](#) Use this API to delete the registered tasks from the memory. After all continuous interpolation tasks are done, use this API for clearing memory.
- ▶ [AxmContiIsMotion](#) Use this API to verify whether continuous interpolation tasks are in use or not.
- ▶ [AxmContiGetTotalNodeNum](#) This API returns total number of registered nodes in the memory.
- ▶ [AxmContiGetNodeNum](#) This API verifies the index number of currently moving continuous interpolation.
- ▶ [AxmContiReadFree](#) This API checks whether memory is empty or not.
- ▶ [AxmContiStart](#) This API starts continuous interpolation task.

C Example

```
long lAxis[2];
double dPos[2];
long lPosSize = 2;
long lCoordinate = 0;
double dVelocity = 200, dAccel = 400;

lAxis[0] = 0;
lAxis[1] = 1;

AxmContiWriteClear(lCoordinate);
AxmContiSetAxisMap (lCoordinate, lPosSize, lAxis);
// Register absolute position move
```

```

AxmContiSetAbsRelMode(lCoordinate, 0);

AxmContiBeginNode (lCoordinate);
dPos[0] = 100 , dPos[1] = 100;
AxmLineMove(lCoordinate, dPos, dVelocity, dAccel, dAccel);
dPos[0] = 200 , dPos[1] = 200;
AxmLineMove(lCoordinate, dPos, dVelocity, dAccel, dAccel);
dPos[0] = 300 , dPos[1] = 300;
AxmLineMove(lCoordinate, dPos, dVelocity, dAccel, dAccel);
dPos[0] = 400 , dPos[1] = 400;
AxmLineMove(lCoordinate, dPos, dVelocity, dAccel, dAccel);
dPos[0] = 500 , dPos[1] = 500;
AxmLineMove(lCoordinate, dPos, dVelocity, dAccel, dAccel);
dPos[0] = 600 , dPos[1] = 600;
AxmLineMove(lCoordinate, dPos, dVelocity, dAccel, dAccel);
dPos[0] = 700 , dPos[1] = 700;
AxmLineMove(lCoordinate, dPos, dVelocity, dAccel, dAccel);
dPos[0] = 800 , dPos[1] = 800;
AxmLineMove(lCoordinate, dPos, dVelocity, dAccel, dAccel);
dPos[0] = 900 , dPos[1] = 900;
AxmLineMove(lCoordinate, dPos, dVelocity, dAccel, dAccel);
dPos[0] = 1000 , dPos[1] = 1000;
AxmLineMove(lCoordinate, dPos, dVelocity, dAccel, dAccel);
dPos[0] = 1100 , dPos[1] = 1100;
AxmLineMove(lCoordinate, dPos, dVelocity, dAccel, dAccel);
AxmContiEndNode (lCoordinate);

//Start absolute position move
AxmContiSetAbsRelMode(lCoordinate, 0);

// Start continuous interpolation move
AxmContiStart (lCoordinate, 0, 0);

```

VB Example

```

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

AxmContiWriteClear lCoordinate
AxmContiSetAxisMap lCoordinate, lPosSize, axes(0)

' Start registering absolute position move
AxmContiSetAbsRelMode lCoordinate, 0
AxmContiBeginNode lCoordinate
dPos(0) = 100: dPos(1) = 100
AxmLineMove lCoordinate, dPos(0), dVelocity, dAccel, dAccel
dPos(0) = 200: dPos(1) = 200
AxmLineMove lCoordinate, dPos(0), dVelocity , dAccel, dAccel
dPos(0) = 300: dPos(1) = 300
AxmLineMove lCoordinate, dPos(0), dVelocity , dAccel, dAccel
dPos(0) = 400: dPos(1) = 400

```

```

AxmLineMove lCoordinate, dPos(0), dVelocity , dAccel, dAccel
dPos(0) = 500: dPos(1) = 500
AxmLineMove lCoordinate, dPos(0), dVelocity , dAccel, dAccel
dPos(0) = 600: dPos(1) = 600
AxmLineMove lCoordinate, dPos(0), dVelocity , dAccel, dAccel
dPos(0) = 700: dPos(1) = 700
AxmLineMove lCoordinate, dPos(0), dVelocity , dAccel, dAccel
AxmContiEndNode lCoordinate

`Start absolute position move
 AxmContiSetAbsRelMode lCoordinate, 0
 `Start continuous interpolation move
AxmContiStart lCoordinate, 0, 0

```

Delphi Example

```

var
  i : LongInt;
  dVelocity, dAccel : Double;
  lAxis : array [0..1] of LongInt;
  dPos : array [0..1] of Double;
  lCoordinate : LongInt;
  lPosSize : LongInt;

begin

  dVelocity := 200;
  dAccel := 400;
  lPosSize := 2;
  lCoordinate := 0;

  // Register absolute position move
  AxmContiSetAbsRelMode(lCoordinate, 0);

  for i := 0 to 1 do
  begin
    lAxis[i] := i;
  end;

  AxmContiWriteClear(lCoordinate);
  AxmContiSetAxisMap(lCoordinate, 2, @lAxis);

  AxmContiBeginNode (lCoordinate);
  dPos[0] := 100 ; dPos[1] := 100;
  AxmLineMove(lCoordinate, @dPos, dVelocity, dAccel, dAccel);
  dPos[0] := 200 ; dPos[1] := 200;
  AxmLineMove(lCoordinate, @dPos, dVelocity, dAccel, dAccel);
  dPos[0] := 300 ; dPos[1] := 300;
  AxmLineMove(lCoordinate, @dPos, dVelocity, dAccel, dAccel);
  dPos[0] := 400 ; dPos[1] := 400;
  AxmLineMove(lCoordinate, @dPos, dVelocity, dAccel, dAccel);
  dPos[0] := 500 ; dPos[1] := 500;
  AxmLineMove(lCoordinate, @dPos, dVelocity, dAccel, dAccel);
  dPos[0] := 600 ; dPos[1] := 600;
  AxmLineMove(lCoordinate, @dPos, dVelocity, dAccel, dAccel);
  dPos[0] := 700 ; dPos[1] := 700;
  AxmLineMove(lCoordinate, @dPos, dVelocity, dAccel, dAccel);
  AxmContiEndNode (lCoordinate);

  // Start absolute position move
  AxmMotSetAbsRelMode(lCoordinate, 0);

  { Start continuous interpolation move }
  AxmContiStart (lCoordinate, 0, 0);
end;

```

See Also

[AxmContiSetAxisMap](#), [AxmContiGetAxisMap](#), [AxmContiSetAbsRelMode](#), [AxmContiGetAbsRelMode](#),
[AxmContiEndNode](#), [AxmContiReadIndex](#), [AxmContiWriteClear](#), [AxmContiStart](#), [AxmContiIsMotion](#),
[AxmContiGetNodeNum](#), [AxmContiGetTotalNodeNum](#), [AxmContiSetOptionNodeNum](#)

AxmContiEndNode

Purpose

To end registering tasks to process continuous interpolation in specified coordinate system.

Format

C

```
DWORD AxmContiEndNode (long lCoord);
```

Visual Basic

```
Function AxmContiEndNode (ByVal lCoord As Long) As Long
```

Delphi

```
function AxmContiEndNode (lCoord: LongInt) : DWord; stdcall;
```

Input / Output

Name	in/out	Format	Init Value	Explanation
Coord	in	long		Coordinate system number (start from 0)

Return AXT_RT_SUCCESS(0000) : Successful execution of API.

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

Description

After this API is called, all motion processing is not actually executed but registered in internal queue until API is called. When [AxmContiStart](#) API is called, registered motion will actually be started,

- ▶ [AxmContiWriteClear](#) Use this API to delete the registered tasks from the memory. After all continuous interpolation tasks are done, use this function for clearing memory.
- ▶ [AxmContiIsMotion](#) Use this API to check whether continuous interpolation tasks are in use or not.
- ▶ [AxmContiGetTotalNodeNum](#) This API returns total number of registered nodes in the memory.
- ▶ [AxmContiGetNodeNum](#) This API verifies the index number of currently moving continuous interpolation.
- ▶ [AxmContiReadFree](#) This API checks whether memory is empty
- ▶ [AxmContiStart](#) This API starts continuous interpolation task.

C Example

```
long    lAxis[2];
double  dPos[2];
long    lPosSize = 2;
long    lCoordinate = 0;
double  dVelocity = 200, dAccel = 400;

lAxis[0] = 0;
lAxis[1] = 1;

AxmContiWriteClear(lCoordinate);
AxmContiSetAxisMap(lCoordinate, lPosSize, lAxis);
// Register absolute position move
AxmContiSetAbsRelMode(lCoordinate, 0);
```

```

AxmContiBeginNode (lCoordinate);
dPos[0] = 100 , dPos[1] = 100;
AxmLineMove(lCoordinate, dPos, dVelocity, dAccel, dAccel);
dPos[0] = 200 , dPos[1] = 200;
AxmLineMove(lCoordinate, dPos, dVelocity, dAccel, dAccel);
dPos[0] = 300 , dPos[1] = 300;
AxmLineMove(lCoordinate, dPos, dVelocity, dAccel, dAccel);
dPos[0] = 400 , dPos[1] = 400;
AxmLineMove(lCoordinate, dPos, dVelocity, dAccel, dAccel);
dPos[0] = 500 , dPos[1] = 500;
AxmLineMove(lCoordinate, dPos, dVelocity, dAccel, dAccel);
dPos[0] = 600 , dPos[1] = 600;
AxmLineMove(lCoordinate, dPos, dVelocity, dAccel, dAccel);
dPos[0] = 700 , dPos[1] = 700;
AxmLineMove(lCoordinate, dPos, dVelocity, dAccel, dAccel);
dPos[0] = 800 , dPos[1] = 800;
AxmLineMove(lCoordinate, dPos, dVelocity, dAccel, dAccel);
dPos[0] = 900 , dPos[1] = 900;
AxmLineMove(lCoordinate, dPos, dVelocity, dAccel, dAccel);
dPos[0] = 1000 , dPos[1] = 1000;
AxmLineMove(lCoordinate, dPos, dVelocity, dAccel, dAccel);
dPos[0] = 1100 , dPos[1] = 1100;
AxmLineMove(lCoordinate, dPos, dVelocity, dAccel, dAccel);
AxmContiEndNode (lCoordinate);

// Start absolute position move
AxmContiSetAbsRelMode(lCoordinate, 0);

// Start continuous interpolation move
AxmContiStart (lCoordinate, 0, 0);

```

VB Example

```

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

AxmContiWriteClear lCoordinate
AxmContiSetAxisMap lCoordinate, lPosSize, axes(0)
'Start registering absolute position move
AxmContiSetAbsRelMode lCoordinate, 0
AxmContiBeginNode lCoordinate
dPos(0) = 100: dPos(1) = 100
AxmLineMove lCoordinate, dPos(0), dVelocity, dAccel, dAccel
dPos(0) = 200: dPos(1) = 200
AxmLineMove lCoordinate, dPos(0), dVelocity , dAccel, dAccel
dPos(0) = 300: dPos(1) = 300
AxmLineMove lCoordinate, dPos(0), dVelocity , dAccel, dAccel
dPos(0) = 400: dPos(1) = 400
AxmLineMove lCoordinate, dPos(0), dVelocity , dAccel, dAccel
dPos(0) = 500: dPos(1) = 500

```

```

AxmLineMove lCoordinate, dPos(0), dVelocity , dAccel, dAccel
dPos(0) = 600: dPos(1) = 600
AxmLineMove lCoordinate, dPos(0), dVelocity , dAccel, dAccel
dPos(0) = 700: dPos(1) = 700
AxmLineMove lCoordinate, dPos(0), dVelocity , dAccel, dAccel
AxmContiEndNode lCoordinate

'Start absolute position move
AxmContiSetAbsRelMode lCoordinate, 0

'Start continuous interpolation move
AxmContiStart lCoordinate, 0, 0

```

Delphi Example

```

var
  i : LongInt;
  dVelocity, dAccel : Double;
  lAxis : array [0..1] of LongInt;
  dPos : array [0..1] of Double;
  lCoordinate : LongInt;
  lPosSize : LongInt;

begin
  dVelocity := 200;
  dAccel := 400;
  lPosSize := 2;
  lCoordinate := 0;

  // Register absolute position move
  AxmContiSetAbsRelMode(lCoordinate, 0);

  for i := 0 to 1 do
  begin
    lAxis[i] := i;
  end;

  AxmContiWriteClear(lCoordinate);
  AxmContiSetAxisMap(lCoordinate, 2, @lAxis);
  AxmContiBeginNode(lCoordinate);
  dPos[0] := 100 ; dPos[1] := 100;
  AxmLineMove(lCoordinate, @dPos, dVelocity, dAccel, dAccel);
  dPos[0] := 200 ; dPos[1] := 200;
  AxmLineMove(lCoordinate, @dPos, dVelocity, dAccel, dAccel);
  dPos[0] := 300 ; dPos[1] := 300;
  AxmLineMove(lCoordinate, @dPos, dVelocity, dAccel, dAccel);
  dPos[0] := 400 ; dPos[1] := 400;
  AxmLineMove(lCoordinate, @dPos, dVelocity, dAccel, dAccel);
  dPos[0] := 500 ; dPos[1] := 500;
  AxmLineMove(lCoordinate, @dPos, dVelocity, dAccel, dAccel);
  dPos[0] := 600 ; dPos[1] := 600;
  AxmLineMove(lCoordinate, @dPos, dVelocity, dAccel, dAccel);
  dPos[0] := 700 ; dPos[1] := 700;
  AxmLineMove(lCoordinate, @dPos, dVelocity, dAccel, dAccel);
  AxmContiEndNode (lCoordinate);

  // Start absolute position move
  AxmMotSetAbsRelMode(lCoordinate, 0);

  { Start continuous interpolation move }
  AxmContiStart (lCoordinate, 0, 0);
end;

```

See Also

[AxmContiSetAxisMap](#), [AxmContiGetAxisMap](#), [AxmContiSetAbsRelMode](#), [AxmContiGetAbsRelMode](#),
[AxmContiBeginNode](#), [AxmContiReadIndex](#), [AxmContiWriteClear](#), [AxmContiStart](#), [AxmContiIsMotion](#),
[AxmContiGetNodeNum](#), [AxmContiGetTotalNodeNum](#), [AxmContiSetOptionNodeNum](#)

AxmContiReadFree

Purpose

Verify internal queue if it is empty for interpolation move.

Format

C

```
DWORD AxmContiReadFree (long lCoord, DWORD *upQueueFree);
```

Visual Basic

```
Function AxmContiReadFree (ByVal lCoord As Long, ByRef upQueueFree As Long) As Long
```

Delphi

```
function AxmContiReadFree (lCoord: LongInt; upQueueFree : PDWord) : DWord; stdcall;
```

Input / Output

Name	in/out	Format	Init Value	Explanation
Coord	in	long		Coordinate system number (start from 0)
QueueFree	out	DWORD*		Status of internal queue

upQueueFree

#define	Value	Explanation
FALSE	00h	Status of internal queue
TRUE	01h	Internal queue is empty

Return AXT_RT_SUCCESS(0000) : Successful execution of API.

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

Description

When using [AxmLineMove](#), [AxmCircleCenterMove](#), [AxmCirclePointMove](#), [AxmCircleRadiusMove](#), [AxmCircleAngleMove](#) API with [AxmContiBeginNode](#) and [AxmContiEndNode](#), it stores tasks to queue and processes continuous interpolation to process interpolation move.
If this internal queue is empty, it says TRUE, or else it says FALSE.

C Example

```
long    lAxis[2];
double  dPos[2];
long    lPosSize = 2;
long    lCoordinate = 0;
double  dVelocity = 200, dAccel = 400;

lAxis[0] = 0;
lAxis[1] = 1;

AxmContiWriteClear(lCoordinate);
AxmContiSetAxisMap (lCoordinate, lPosSize, lAxis);
// Register absolute position move
AxmContiSetAbsRelMode(lCoordinate, 0);
```

```

AxmContiBeginNode(lCoordinate);
dPos[0] = 100 , dPos[1] = 100;
AxmLineMove(lCoordinate, dPos, dVelocity, dAccel, dAccel);
dPos[0] = 200 , dPos[1] = 200;
AxmLineMove(lCoordinate, dPos, dVelocity, dAccel, dAccel);
dPos[0] = 300 , dPos[1] = 300;
AxmLineMove(lCoordinate, dPos, dVelocity, dAccel, dAccel);
dPos[0] = 400 , dPos[1] = 400;
AxmLineMove(lCoordinate, dPos, dVelocity, dAccel, dAccel);
dPos[0] = 500 , dPos[1] = 500;
AxmLineMove(lCoordinate, dPos, dVelocity, dAccel, dAccel);
dPos[0] = 600 , dPos[1] = 600;
AxmLineMove(lCoordinate, dPos, dVelocity, dAccel, dAccel);
dPos[0] = 700 , dPos[1] = 700;
AxmLineMove(lCoordinate, dPos, dVelocity, dAccel, dAccel);
dPos[0] = 800 , dPos[1] = 800;
AxmLineMove(lCoordinate, dPos, dVelocity, dAccel, dAccel);
dPos[0] = 900 , dPos[1] = 900;
AxmLineMove(lCoordinate, dPos, dVelocity, dAccel, dAccel);
dPos[0] = 1000 , dPos[1] = 1000;
AxmLineMove(lCoordinate, dPos, dVelocity, dAccel, dAccel);
dPos[0] = 1100 , dPos[1] = 1100;
AxmLineMove(lCoordinate, dPos, dVelocity, dAccel, dAccel);
AxmContiEndNode (lCoordinate);

//Start absolute position move
AxmContiSetAbsRelMode(lCoordinate, 0);

// Start continuous interpolation move
AxmContiStart (lCoordinate, 0, 0);

// Identify if internal Queue is empty for interpolation move
DWORD ReadInterpolationQueueFree;
AxmContiReadFree (lCoordinate, &ReadInterpolationQueueFree);
if(ReadInterpolationQueueFree) printf("Internal Queue is empty.");
else           printf("Internal Queue is not empty.");

```

VB Example

```

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

AxmContiWriteClear lCoordinate
AxmContiSetAxisMap lCoordinate, lPosSize, axes(0)
' Register absolute position move
AxmContiSetAbsRelMode lCoordinate, 0

AxmContiBeginNode lCoordinate
dPos(0) = 100: dPos(1) = 100
AxmLineMove lCoordinate, dPos(0), dVelocity, dAccel, dAccel

```

```

dPos(0) = 200: dPos(1) = 200
AxmLineMove lCoordinate, dPos(0), dVelocity , dAccel, dAccel
dPos(0) = 300: dPos(1) = 300
AxmLineMove lCoordinate, dPos(0), dVelocity , dAccel, dAccel
dPos(0) = 400: dPos(1) = 400
AxmLineMove lCoordinate, dPos(0), dVelocity , dAccel, dAccel
dPos(0) = 500: dPos(1) = 500
AxmLineMove lCoordinate, dPos(0), dVelocity , dAccel, dAccel
dPos(0) = 600: dPos(1) = 600
AxmLineMove lCoordinate, dPos(0), dVelocity , dAccel, dAccel
dPos(0) = 700: dPos(1) = 700
AxmLineMove lCoordinate, dPos(0), dVelocity , dAccel, dAccel
AxmContiEndNode lCoordinate

' Register absolute position move
  AxmContiSetAbsRelMode lCoordinate, 0

  ' Start continuous interpolation move
  AxmContiStart lCoordinate, 0, 0

  ' Identify if internal Queue is empty for interpolation move
  Dim ReadInterpolationQueueFree As Long
  AxmContiReadFree lCoordinate, ReadInterpolationQueueFree
  If ReadInterpolationQueueFree Then
    MsgBox "Internal Queue is empty.", vbOKCancel
  Else
    MsgBox "Internal Queue is not empty.", 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;
  ReadInterpolationQueueFree : DWORD;
begin

  dVelocity := 200;
  dAccel := 400;
  lPosSize := 2;
  lCoordinate := 0;

  // Register absolute position move
  AxmContiSetAbsRelMode(lCoordinate, 0);

  for i := 0 to 1 do
  begin
    lAxis[i] := i;
  end;

  AxmContiWriteClear(lCoordinate);
  AxmContiSetAxisMap(lCoordinate, 2, @lAxis);
  AxmContiBeginNode(lCoordinate);
  dPos[0] := 100 ; dPos[1] := 100;
  AxmLineMove(lCoordinate, @dPos, dVelocity, dAccel, dAccel);
  dPos[0] := 200 ; dPos[1] := 200;
  AxmLineMove(lCoordinate, @dPos, dVelocity, dAccel, dAccel);
  dPos[0] := 300 ; dPos[1] := 300;
  AxmLineMove(lCoordinate, @dPos, dVelocity, dAccel, dAccel);
  dPos[0] := 400 ; dPos[1] := 400;
  AxmLineMove(lCoordinate, @dPos, dVelocity, dAccel, dAccel);
  dPos[0] := 500 ; dPos[1] := 500;

```

```
AxmLineMove(lCoordinate, @dPos, dVelocity, dAccel, dAccel);
dPos[0] := 600 ; dPos[1] := 600;
AxmLineMove(lCoordinate, @dPos, dVelocity, dAccel, dAccel);
dPos[0] := 700 ; dPos[1] := 700;
AxmLineMove(lCoordinate, @dPos, dVelocity, dAccel, dAccel);
AxmContiEndNode (lCoordinate);

// Start absolute position move
AxmMotSetAbsRelMode(lCoordinate, 0);

{ Start continuous interpolation move }
AxmContiStart (lCoordinate, 0, 0);

{ Identify if internal Queue is empty for interpolation move }
AxmContiReadFree (lCoordinate, @ReadInterpolationQueueFree);
if ReadInterpolationQueueFree = 1 then
    Application.MessageBox('Internal Queue is empty.', 'Ajinextek', MB_OK)
else
    Application.MessageBox('Internal Queue is not empty.', 'Ajinextek',
        MB_OK);

end;
```

See Also

[AxmContiSetAxisMap](#), [AxmContiGetAxisMap](#), [AxmContiSetAbsRelMode](#), [AxmContiGetAbsRelMode](#),
[AxmContiBeginNode](#), [AxmContiEndNode](#), [AxmContiReadIndex](#), [AxmContiWriteClear](#), [AxmContiStart](#),
[AxmContiIsMotion](#), [AxmContiGetNodeNum](#), [AxmContiGetTotalNodeNum](#), [AxmContiSetOptionNodeNum](#)

AxmContiReadIndex

Purpose

Identify the number of interpolation moves stored in internal queue for processing interpolation move.

Format

C

```
DWORD AxmContiReadIndex (long lCoord, long *lpQueueIndex);
```

Visual Basic

```
Function AxmContiReadIndex (ByVal lCoord As Long, ByRef lpQueueIndex As Long) As Long
```

Delphi

```
function AxmContiReadIndex (lCoord: LongInt; lpQueueIndex : PLongInt) : DWord; stdcall;
```

Input / Output

Name	in/out	Format	Init Value	Explanation
Coord	in	long		Coordinate system number (start from 0)
QueueIndex	out	long*		The number of stored interpolation queue

Return AXT_RT_SUCCESS(0000) : Successful execution of API.

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

Description

When using [AxmLineMove](#), [AxmCircleCenterMove](#), [AxmCirclePointMove](#), [AxmCircleRadiusMove](#), [AxmCircleAngleMove](#) API with [AxmContiBeginNode](#) and [AxmContiEndNode](#), it stores tasks to queue and processes continuous interpolation to process interpolation move.

Use this API to identify the number of internal queue. One [AxmLineMove](#) API is regarded as one unit.

C Example

```
long ReadInterpolMotionQueueIndex;
long lAxis[2];
double dPos[2];
long lPosSize = 2;
long lCoordinate = 0;
double dVelocity = 200, dAccel = 400;

lAxis[0] = 0;
lAxis[1] = 1;

AxmContiWriteClear(lCoordinate);
AxmContiSetAxisMap(lCoordinate, lPosSize, lAxis);
// Register absolute position move
AxmContiSetAbsRelMode(lCoordinate, 0);

AxmContiBeginNode(lCoordinate);
dPos[0] = 100 , dPos[1] = 100;
AxmLineMove(lCoordinate, dPos, dVelocity, dAccel, dAccel);
dPos[0] = 200 , dPos[1] = 200;
```

```

AxmLineMove(lCoordinate, dPos, dVelocity, dAccel, dAccel);
dPos[0] = 300 , dPos[1] = 300;
AxmLineMove(lCoordinate, dPos, dVelocity, dAccel, dAccel);
dPos[0] = 400 , dPos[1] = 400;
AxmLineMove(lCoordinate, dPos, dVelocity, dAccel, dAccel);
dPos[0] = 500 , dPos[1] = 500;
AxmLineMove(lCoordinate, dPos, dVelocity, dAccel, dAccel);
dPos[0] = 600 , dPos[1] = 600;
AxmLineMove(lCoordinate, dPos, dVelocity, dAccel, dAccel);
dPos[0] = 700 , dPos[1] = 700;
AxmLineMove(lCoordinate, dPos, dVelocity, dAccel, dAccel);
dPos[0] = 800 , dPos[1] = 800;
AxmLineMove(lCoordinate, dPos, dVelocity, dAccel, dAccel);
dPos[0] = 900 , dPos[1] = 900;
AxmLineMove(lCoordinate, dPos, dVelocity, dAccel, dAccel);
dPos[0] = 1000 , dPos[1] = 1000;
AxmLineMove(lCoordinate, dPos, dVelocity, dAccel, dAccel);
dPos[0] = 1100 , dPos[1] = 1100;
AxmLineMove(lCoordinate, dPos, dVelocity, dAccel, dAccel);
AxmContiEndNode (lCoordinate);

//Start absolute position move
AxmContiSetAbsRelMode(lCoordinate, 0);

// Start continuous interpolation move
AxmContiStart (lCoordinate, 0, 0);

// Identify the number of interpolation move stored in internal Queue for
// interpolation move
if(AxmContiReadIndex(lCoordinate, &ReadInterpolMotionQueueIndex)
   == AXT_RT_SUCCESS){
printf(" %ld of continuous interpolation move inside.. \n",
      ReadInterpolMotionQueueIndex);
}

```

VB Example

```

Dim ReadInterpolMotionQueueIndex As Long
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 strData As String
Dim i As Long

dVelocity = 200
dAccel = 400
lPosSize = 2
lCoordinate = 0

For i = 0 To 1
    axes(i) = i
Next i

AxmContiWriteClear lCoordinate
AxmContiSetAxisMap lCoordinate, lPosSize, axes(0)

' Register absolute position move
AxmContiSetAbsRelMode lCoordinate, 0

AxmContiBeginNode lCoordinate
dPos(0) = 100: dPos(1) = 100
AxmLineMove lCoordinate, dPos(0), dVelocity, dAccel, dAccel
dPos(0) = 200: dPos(1) = 200

```

```

AxmLineMove lCoordinate, dPos(0), dVelocity , dAccel, dAccel
dPos(0) = 300: dPos(1) = 300
AxmLineMove lCoordinate, dPos(0), dVelocity , dAccel, dAccel
dPos(0) = 400: dPos(1) = 400
AxmLineMove lCoordinate, dPos(0), dVelocity , dAccel, dAccel
dPos(0) = 500: dPos(1) = 500
AxmLineMove lCoordinate, dPos(0), dVelocity , dAccel, dAccel
dPos(0) = 600: dPos(1) = 600
AxmLineMove lCoordinate, dPos(0), dVelocity , dAccel, dAccel
dPos(0) = 700: dPos(1) = 700
AxmLineMove lCoordinate, dPos(0), dVelocity , dAccel, dAccel
AxmContiEndNode lCoordinate

`Start absolute position move
AxmContiSetAbsRelMode lCoordinate, 0

` Start continuous interpolation move
AxmContiStart lCoordinate, 0, 0

`Identify the number of interpolation move stored in internal Queue for
interpolation move
If(AxmContiReadIndex (lCoordinate, ReadInterpolMotionQueueIndex) =
    AXT_RT_SUCCESS) Then
strData = "continuous interpolation queues" +
    CStr(ReadInterpolMotionQueueIndex) + "inside."
MsgBox strData
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;
  ReadInterpolationQueueFree : DWORD;
  ReadInterpolMotionQueueIndex,ReadInterpolMotionQueueIndex2 : longInt;
  strData : String;
begin

  dVelocity := 200;
  dAccel := 400;
  lPosSize := 2;
  lCoordinate := 0;

  // Register absolute position move
  AxmContiSetAbsRelMode(lCoordinate, 0);

  for i := 0 to 1 do
  begin
    lAxis[i] := i;
  end;

  AxmContiWriteClear(lCoordinate);
  AxmContiSetAxisMap(lCoordinate, 2, @lAxis);

  AxmContiBeginNode(lCoordinate);
  dPos[0] := 100 ; dPos[1] := 100;
  AxmLineMove(lCoordinate, @dPos, dVelocity, dAccel, dAccel);
  dPos[0] := 200 ; dPos[1] := 200;
  AxmLineMove(lCoordinate, @dPos, dVelocity, dAccel, dAccel);
  dPos[0] := 300 ; dPos[1] := 300;
  AxmLineMove(lCoordinate, @dPos, dVelocity, dAccel, dAccel);
  dPos[0] := 400 ; dPos[1] := 400;

```

```
AxmLineMove(lCoordinate, @dPos, dVelocity, dAccel, dAccel);
dPos[0] := 500 ; dPos[1] := 500;
AxmLineMove(lCoordinate, @dPos, dVelocity, dAccel, dAccel);
dPos[0] := 600 ; dPos[1] := 600;
AxmLineMove(lCoordinate, @dPos, dVelocity, dAccel, dAccel);
dPos[0] := 700 ; dPos[1] := 700;
AxmLineMove(lCoordinate, @dPos, dVelocity, dAccel, dAccel);
AxmContiEndNode (lCoordinate);

// Start absolute position move
AxmMotSetAbsRelMode(lCoordinate, 0);

{ Start continuous interpolation move }
AxmContiStart (lCoordinate, 0, 0);

{ Identify the number of interpolation move stored in internal queue for
interpolation move }
if (AxmContiReadIndex (lCoordinate, @ReadInterpolMotionQueueIndex) =
    AXT_RT_SUCCESS) then
begin
strData := 'continuous interpolation queue ' +
           IntToStr(ReadInterpolMotionQueueIndex) + 'inside.';
Application.MessageBox (PCHAR(strData), 'Ajinextek', MB_OK);
End;
End;
```

See Also

[AxmContiSetAxisMap](#), [AxmContiGetAxisMap](#), [AxmContiSetAbsRelMode](#),
[AxmContiGetAbsRelMode](#), [AxmContiBeginNode](#), [AxmContiEndNode](#), [AxmContiReadFree](#),
[AxmContiWriteClear](#), [AxmContiStart](#), [AxmContiIsMotion](#), [AxmContiGetNodeNum](#),
[AxmContiGetTotalNodeNum](#), [AxmContiSetOptionNodeNum](#)

AxmContiWriteClear

Purpose

To delete all internal queues stored for continuous interpolation move.

Format

C

```
DWORD AxmContiWriteClear (long lCoord);
```

Visual Basic

```
Function AxmContiWriteClear (ByVal lCoord As Long) As Long
```

Delphi

```
function AxmContiWriteClear (lCoord: LongInt) : DWord; stdcall;
```

Input / Output

Name	in/out	Format	Init Value	Explanation
Coord	in	long		Coordinate system number(start from 0)

Return AXT_RT_SUCCESS(0000) : Successful execution of API.

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

Description

When using [AxmLineMove](#), [AxmCircleCenterMove](#), [AxmCirclePointMove](#), [AxmCircleRadiusMove](#), [AxmCircleAngleMove](#) API with [AxmContiBeginNode](#) and [AxmContiEndNode](#), it stores tasks to queue and processes continuous interpolation to process interpolation move. One [AxmLineMove](#) API is regarded as one unit.

Use this API for deleting all the number of queue stored for continuous interpolation move.

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);
//relative position move
AxmContiSetAbsRelMode(lCoordinate, 1);
// line interpolation move
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

AxmContiWriteClear lCoordinate
AxmContiSetAxisMap lCoordinate, 2, axes(0)
    ' relative position move
AxmContiSetAbsRelMode lCoordinate, 1
    ' line interpolation move
    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;

    AxmContiWriteClear(lCoordinate);
    AxmContiSetAxisMap (lCoordinate, lPosSize, @lAxis);

    { relative position move }
    AxmContiSetAbsRelMode(lCoordinate, 1);
    { line interpolation move }
    AxmLineMove(lCoordinate, @dPos, 200, 400, 400);

end;

```

See Also

[AxmContiSetAxisMap](#), [AxmContiGetAxisMap](#), [AxmContiSetAbsRelMode](#), [AxmContiGetAbsRelMode](#),
[AxmContiBeginNode](#), [AxmContiEndNode](#), [AxmContiReadFree](#), [AxmContiReadIndex](#), [AxmContiStart](#),
[AxmContiIsMotion](#), [AxmContiGetNodeNum](#), [AxmContiGetTotalNodeNum](#), [AxmContiSetOptionNodeNum](#)

AxmContiStart

Purpose

To start the move of registered internal continuous interpolation queue

Format

C

```
DWORD AxmContiStart(long lCoord, DWORD dwProfileset, long lAngle);
```

Visual Basic

```
Function AxmContiStart (ByVal lCoord As Long, ByVal dwProfileset As Long, ByVal lAngle As Long) As Long
```

Delphi

```
function AxmContiStart (lCoord : LongInt; dwProfileset : DWord; lAngle: LongInt) : DWord; stdcall;
```

Input / Output

Name	in/out	Format	Init Value	Explanation
Coord	in	long		Coordinate system number (start from 0)
Profileset	in	DWORD		Profile mode
Angle	in	long		Angle when using automatic ac/deceleration mode

Profileset

#define	Value	Explanation
CONTI_NODE_VELOCITY	00h	Speed specified interpolation mode
CONTI_NODE_MANUAL	01h	Node ac/deceleration interpolation mode
CONTI_NODE_AUTO	02h	Automatic ac/deceleration interpolation mode

Return AXT_RT_SUCCESS(0000) : Successful execution of API.

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

Description

When using [AxmLineMove](#), [AxmCircleCenterMove](#), [AxmCirclePointMove](#), [AxmCircleRadiusMove](#), [AxmCircleAngleMove](#) API with [AxmContiBeginNode](#) and [AxmContiEndNode](#), it stores tasks to queue and processes continuous interpolation to process interpolation move. One [AxmLineMove](#) API is regarded as one unit.

Use this API to run internal queue stored for continuous interpolation move.

- ▶ By AxmSStop and AxmEstop API, motion deceleration stop and motion emergency stop are possible. It always stops by transmitting master axis of coordinate.

List continuous interpolation motion used here is the function in which several steps of task are registered and processed collectively. Advantage of continuous interpolation motion is continuous execution of tasks without interruption of profile between tasks. Generally in equipment, by inserting circular arc to the corner where two lines meet (such as dispensing, manufacturing and painting), continuous control of line->circular arc -> line -> circular arc -> line interpolation is possible.

C Example

```

long lAxis[2];
double dPos[2];
long lPosSize = 2;
long lCoordinate = 0;
double dVelocity = 200, dAccel = 400;

lAxis[0] = 0;
lAxis[1] = 1;

AxmContiWriteClear(lCoordinate);
AxmContiSetAxisMap (lCoordinate, lPosSize, lAxis);
//Register absolute position move
AxmContiSetAbsRelMode(lCoordinate, 0);

AxmContiBeginNode(lCoordinate);
dPos[0] = 100 , dPos[1] = 100;
AxmLineMove(lCoordinate, dPos, dVelocity, dAccel, dAccel);
dPos[0] = 200 , dPos[1] = 200;
AxmLineMove(lCoordinate, dPos, dVelocity, dAccel, dAccel);
dPos[0] = 300 , dPos[1] = 300;
AxmLineMove(lCoordinate, dPos, dVelocity, dAccel, dAccel);
dPos[0] = 400 , dPos[1] = 400;
AxmLineMove(lCoordinate, dPos, dVelocity, dAccel, dAccel);
dPos[0] = 500 , dPos[1] = 500;
AxmLineMove(lCoordinate, dPos, dVelocity, dAccel, dAccel);
AxmContiEndNode (lCoordinate);

//Start absolute position move
AxmContiSetAbsRelMode(lCoordinate, 0);
// Start continuous interpolation move
AxmContiStart(lCoordinate, 0, 0);

```

VB Example

```

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

AxmContiWriteClear lCoordinate
AxmContiSetAxisMap lCoordinate, lPosSize, axes(0)

' Register absolute position move
AxmContiSetAbsRelMode lCoordinate, 0
AxmContiBeginNode lCoordinate
dPos(0) = 100: dPos(1) = 100
AxmLineMove lCoordinate, dPos(0), dVelocity, dAccel, dAccel
dPos(0) = 200: dPos(1) = 200
AxmLineMove lCoordinate, dPos(0), dVelocity , dAccel, dAccel
dPos(0) = 300: dPos(1) = 300

```

```

AxmLineMove lCoordinate, dPos(0), dVelocity , dAccel, dAccel
dPos(0) = 400: dPos(1) = 400
AxmLineMove lCoordinate, dPos(0), dVelocity , dAccel, dAccel
dPos(0) = 500: dPos(1) = 500
AxmLineMove lCoordinate, dPos(0), dVelocity , dAccel, dAccel
AxmContiEndNode lCoordinate

` Start absolute position move
AxmContiSetAbsRelMode lCoordinate, 0
` Start continuous interpolation move
AxmContiStart lCoordinate, 0, 0

```

Delphi Example

```

var
  i : LongInt;
  dVelocity, dAccel : Double;
  lAxis : array [0..1] of LongInt;
  dPos : array [0..1] of Double;
  lCoordinate : LongInt;
  lPosSize : LongInt;

begin

  dVelocity := 200;
  dAccel := 400;
  lPosSize := 2;
  lCoordinate := 0;

  //Register absolute position move
  AxmContiSetAbsRelMode(lCoordinate, 0);

  for i := 0 to 1 do
  begin
    lAxis[i] := i;
  end;

  AxmContiWriteClear(lCoordinate);
  AxmContiSetAxisMap (lCoordinate, 2, @lAxis);
  AxmContiBeginNode(lCoordinate);
  dPos[0] := 100 ; dPos[1] := 100;
  AxmLineMove(lCoordinate, @dPos, dVelocity, dAccel, dAccel);
  dPos[0] := 200 ; dPos[1] := 200;
  AxmLineMove(lCoordinate, @dPos, dVelocity, dAccel, dAccel);
  dPos[0] := 300 ; dPos[1] := 300;
  AxmLineMove(lCoordinate, @dPos, dVelocity, dAccel, dAccel);
  dPos[0] := 400 ; dPos[1] := 400;
  AxmLineMove(lCoordinate, @dPos, dVelocity, dAccel, dAccel);
  dPos[0] := 500 ; dPos[1] := 500;
  AxmLineMove(lCoordinate, @dPos, dVelocity, dAccel, dAccel);
  AxmContiEndNode (lCoordinate);

  // Start absolute position move
  AxmMotSetAbsRelMode(lCoordinate, 0);
  { Start continuous interpolation move }
  AxmContiStart(lCoordinate, 0, 0);
end;

```

See Also

[AxmContiSetAxisMap](#), [AxmContiGetAxisMap](#), [AxmContiSetAbsRelMode](#), [AxmContiGetAbsRelMode](#),
[AxmContiBeginNode](#), [AxmContiEndNode](#)., [AxmContiReadFree](#), [AxmContiReadIndex](#), [AxmContiWriteClear](#),
[AxmContiIsMotion](#), [AxmContiGetNodeNum](#), [AxmContiGetTotalNodeNum](#), [AxmContiSetOptionNodeNum](#)

AxmContilsMotion

Purpose

To identify if continuous interpolation is in move or not

Format

C

```
DWORD AxmContilsMotion (long lCoord, DWORD *upInMotion);
```

Visual Basic

```
Function AxmContilsMotion (ByVal lCoord As Long, ByRef upInMotion As Long) As Long
```

Delphi

```
function AxmContilsMotion (lCoord: LongInt; upInMotion : PDWord) : DWord; stdcall;
```

Input / Output

Name	in/out	Format	Init Value	Explanation
Coord	in	long		Coordinate system number (start from 0)
InMotion	out	DWORD*		Identify if continuous interpolation is in move

InMotion

upStatus	Value	Explanation
FALSE	00h	continuous interpolation is not in move
TRUE	01h	continuous interpolation is not in move

Return AXT_RT_SUCCESS(0000) : Successful execution of API.

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

Description

When using [AxmLineMove](#), [AxmCircleCenterMove](#), [AxmCirclePointMove](#), [AxmCircleRadiusMove](#), [AxmCircleAngleMove](#) API with [AxmContiBeginNode](#) and [AxmContiEndNode](#), it stores tasks to queue and processes continuous interpolation to process interpolation move. One [AxmLineMove](#) API is regarded as one unit.

Use this API to identify whether coordinate system is in move when it starts continuous interpolation while internal queue stored for continuous interpolation move is running continuous interpolation.

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;
```

```

AxmContiWriteClear(lCoordinate);
AxmContiSetAxisMap (lCoordinate, lPosSize, lAxis);
//Register absolute position move
AxmContiSetAbsRelMode(lCoordinate, 0);

AxmContiBeginNode(lCoordinate);
dPos[0] = 100 , dPos[1] = 100;
AxmLineMove(lCoordinate, dPos, dVelocity, dAccel, dAccel);
dPos[0] = 200 , dPos[1] = 200;
AxmLineMove(lCoordinate, dPos, dVelocity, dAccel, dAccel);
dPos[0] = 300 , dPos[1] = 300;
AxmLineMove(lCoordinate, dPos, dVelocity, dAccel, dAccel);
dPos[0] = 400 , dPos[1] = 400;
AxmLineMove(lCoordinate, dPos, dVelocity, dAccel, dAccel);
dPos[0] = 500 , dPos[1] = 500;
AxmLineMove(lCoordinate, dPos, dVelocity, dAccel, dAccel);
AxmContiEndNode (lCoordinate);

//Start absolute position move
AxmContiSetAbsRelMode(lCoordinate, 0);
// Start continuous interpolation move
AxmContiStart (lCoordinate, 0, 0);

Sleep(100);
AxmContiIsMotion (lCoordinate, &IsContMotion);
if(IsContMotion ){ // Identify whether in move or not.
MessageBox("List continuous interpolation move is in motion now\n");
}

```

VB Example

```

Public Declare Sub Sleep Lib "kernel32" Alias "Sleep" (ByVal
      dwMilliseconds As Long)
' In basic, header file should be declared to use Sleep API.
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

AxmContiWriteClear lCoordinate
AxmContiSetAxisMap lCoordinate, lPosSize, axes(0)

' Register absolute position move
AxmContiSetAbsRelMode lCoordinate, 0

AxmContiBeginNode lCoordinate
dPos(0) = 100: dPos(1) = 100
AxmLineMove lCoordinate, dPos(0), dVelocity, dAccel, dAccel
dPos(0) = 200: dPos(1) = 200
AxmLineMove lCoordinate, dPos(0), dVelocity , dAccel, dAccel
dPos(0) = 300: dPos(1) = 300
AxmLineMove lCoordinate, dPos(0), dVelocity , dAccel, dAccel

```

```

dPos(0) = 400: dPos(1) = 400
AxmLineMove lCoordinate, dPos(0), dVelocity , dAccel, dAccel
dPos(0) = 500: dPos(1) = 500
AxmLineMove lCoordinate, dPos(0), dVelocity , dAccel, dAccel
AxmContiEndNode lCoordinate

' Start absolute position move
AxmContiSetAbsRelMode lCoordinate, 0
' Start continuous interpolation move
AxmContiStart lCoordinate, 0, 0

Sleep 100
AxmContiIsMotion lCoordinate, IsContMotion
If(IsContMotion = 1) Then
MsgBox "List continuous interpolation move is in motion now.", 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;

  // Register absolute position move
  AxmContiSetAbsRelMode(lCoordinate, 0);

  for i := 0 to 1 do
  begin
    lAxis[i] := i;
  end;

  AxmContiWriteClear(lCoordinate);
  AxmContiSetAxisMap (lCoordinate, 2, @lAxis);

  AxmContiBeginNode(lCoordinate);
  dPos[0] := 100 ; dPos[1] := 100;
  AxmLineMove(lCoordinate, @dPos, dVelocity, dAccel, dAccel);
  dPos[0] := 200 ; dPos[1] := 200;
  AxmLineMove(lCoordinate, @dPos, dVelocity, dAccel, dAccel);
  dPos[0] := 300 ; dPos[1] := 300;
  AxmLineMove(lCoordinate, @dPos, dVelocity, dAccel, dAccel);
  dPos[0] := 400 ; dPos[1] := 400;
  AxmLineMove(lCoordinate, @dPos, dVelocity, dAccel, dAccel);
  dPos[0] := 500 ; dPos[1] := 500;
  AxmLineMove(lCoordinate, @dPos, dVelocity, dAccel, dAccel);
  AxmContiEndNode (lCoordinate);

  // Start absolute position move
  AxmMotSetAbsRelMode(lCoordinate, 0);
  { Start continuous interpolation move }
  AxmContiStart (lCoordinate, 0, 0);
  Sleep(100);
  AxmContiIsMotion (lCoordinate, @IsContMotion);
  if(IsContMotion = 1) then { Identify if interpolation is in motion or

```

```
    not..}
begin
Application.MessageBox('List continuous interpolation move is in motion
now.', 'Ajinextek', MB_OK);

End ;
end;
```

See Also

[AxmContiSetAxisMap](#), [AxmContiGetAxisMap](#), [AxmContiSetAbsRelMode](#), [AxmContiGetAbsRelMode](#),
[AxmContiBeginNode](#), [AxmContiEndNode](#), [AxmContiReadFree](#), [AxmContiReadIndex](#), [AxmContiWriteClear](#),
[AxmContiStart](#), [AxmContiGetNodeNum](#), [AxmContiGetTotalNodeNum](#), [AxmContiSetOptionNodeNum](#)

AxmContiGetNodeNum

Purpose

To identify index number of current running continuous interpolation while continuous interpolation move.

Format

C

```
DWORD AxmContiGetNodeNum (long lCoord, long *lpNodeNum);
```

Visual Basic

```
Function AxmContiGetNodeNum (ByVal lCoord As Long, ByRef lpNodeNum As Long) As Long
```

Delphi

```
function AxmContiGetNodeNum (lCoord: LongInt; lpNodeNum : PLongInt) : Dword; stdcall;
```

Input / Output

Name	in/out	Format	Init Value	Explanation
Coord	in	long		Coordinate system number (start from 0)
NodeNum	out	long*		Index number of continuous interpolation

Return AXT_RT_SUCCESS(0000) : Successful execution of API.

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

Description

When using [AxmLineMove](#), [AxmCircleCenterMove](#), [AxmCirclePointMove](#), [AxmCircleRadiusMove](#), [AxmCircleAngleMove](#) API with [AxmContiBeginNode](#) and [AxmContiEndNode](#), it stores tasks to queue and processes continuous interpolation to process interpolation move. One [AxmLineMove](#) API is regarded as one unit.

During continuous interpolation move of internal queue stored for continuous interpolation move, it identifies index number of currently running continuous interpolation when it starts continuous interpolation move. Note that there are always 7 units in internal hardware queue in order to move.

C Example

```
long    lpNodeNum1;
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;

AxmContiWriteClear(lCoordinate);
AxmContiSetAxisMap (lCoordinate, lPosSize, lAxis);
//Register absolute position move
AxmContiSetAbsRelMode(lCoordinate, 0);

AxmContiBeginNode (lCoordinate);
```

```

dPos[0] = 100 , dPos[1] = 100;
AxmLineMove(lCoordinate, dPos, dVelocity, dAccel, dAccel);
dPos[0] = 200 , dPos[1] = 200;
AxmLineMove(lCoordinate, dPos, dVelocity, dAccel, dAccel);
dPos[0] = 300 , dPos[1] = 300;
AxmLineMove(lCoordinate, dPos, dVelocity, dAccel, dAccel);
dPos[0] = 400 , dPos[1] = 400;
AxmLineMove(lCoordinate, dPos, dVelocity, dAccel, dAccel);
dPos[0] = 500 , dPos[1] = 500;
AxmLineMove(lCoordinate, dPos, dVelocity, dAccel, dAccel);
AxmContiEndNode (lCoordinate);

//Start absolute position move
AxmContiSetAbsRelMode(lCoordinate, 0);
// Start continuous interpolation move
AxmContiStart (lCoordinate, 0, 0);

Sleep(100);
AxmContiIsMotion (lCoordinate, &IsContMotion);
if(IsContMotion ){ // Identify interpolation is in motion or not..
AxmContiGetNodeNum (0, &lpNodeNum1);
printf("\t Currently operating list interpolation move number is %ld\n",
      lpNodeNum1 );
}

```

VB Example

```

Public Declare Sub Sleep Lib "kernel32" Alias "Sleep" (ByVal
    dwMilliseconds As Long)
'In basic, header file should be declared to use Sleep function.
Dim IsContMotion, lpNodeNum1 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

AxmContiWriteClear lCoordinate
AxmContiSetAxisMap lCoordinate, lPosSize, axes(0)

' Register absolute position move
AxmContiSetAbsRelMode lCoordinate, 0

AxmContiBeginNode lCoordinate
dPos(0) = 100: dPos(1) = 100
AxmLineMove lCoordinate, dPos(0), dVelocity, dAccel, dAccel
dPos(0) = 200: dPos(1) = 200
AxmLineMove lCoordinate, dPos(0), dVelocity , dAccel, dAccel
dPos(0) = 300: dPos(1) = 300
AxmLineMove lCoordinate, dPos(0), dVelocity , dAccel, dAccel
dPos(0) = 400: dPos(1) = 400
AxmLineMove lCoordinate, dPos(0), dVelocity , dAccel, dAccel
dPos(0) = 500: dPos(1) = 500
AxmLineMove lCoordinate, dPos(0), dVelocity , dAccel, dAccel

```

```

AxmContiEndNode lCoordinate

` Start absolute position move
AxmContiSetAbsRelMode lCoordinate, 0
` Start continuous interpolation move
AxmContiStart lCoordinate, 0, 0

Sleep 100
AxmContiIsMotion lCoordinate, IsContMotion
If(IsContMotion = 1) Then
AxmContiGetNodeNum 0, lpNodeNum1
strData = "Currently operating list interpolation move number" +
          CStr(lpNodeNum1) + "is.."
MsgBox strData
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;
  lpNodeNum1 : longInt;
  strData : String;

begin

  dVelocity := 200;
  dAccel := 400;
  lPosSize := 2;
  lCoordinate := 0;

  //Register absolute position move
  AxmContiSetAbsRelMode(lCoordinate, 0);

  for i := 0 to 1 do
  begin
    lAxis[i] := i;
  end;

  AxmContiWriteClear(lCoordinate);
  AxmContiSetAxisMap (lCoordinate, 2, @lAxis);

  AxmContiBeginNode(lCoordinate);
  dPos[0] := 100 ; dPos[1] := 100;
  AxmLineMove(lCoordinate, @dPos, dVelocity, dAccel, dAccel);
  dPos[0] := 200 ; dPos[1] := 200;
  AxmLineMove(lCoordinate, @dPos, dVelocity, dAccel, dAccel);
  dPos[0] := 300 ; dPos[1] := 300;
  AxmLineMove(lCoordinate, @dPos, dVelocity, dAccel, dAccel);
  dPos[0] := 400 ; dPos[1] := 400;
  AxmLineMove(lCoordinate, @dPos, dVelocity, dAccel, dAccel);
  dPos[0] := 500 ; dPos[1] := 500;
  AxmLineMove(lCoordinate, @dPos, dVelocity, dAccel, dAccel);
  AxmContiEndNode (lCoordinate);

  // Start absolute position move
  AxmMotSetAbsRelMode(lCoordinate, 0);
  { Start continuous interpolation move }
  AxmContiStart (lCoordinate, 0, 0);
  Sleep(100);
  AxmContiIsMotion (lCoordinate, @IsContMotion);

```

```
if(IsContMotion = 1 ) then { Identify if interpolation is in motion or
    not.}
begin
AxmContiGetNodeNum ( 0 , @lpNodeNum1 );
strData      := 'Currently operating interpolation move number' +
    IntToStr(lpNodeNum1) + 'is...';
Application.MessageBox ( PCHAR(strData), 'Ajinextek', MB_OK );

End ;
end;
```

See Also

[AxmContiSetAxisMap](#), [AxmContiGetAxisMap](#), [AxmContiSetAbsRelMode](#), [AxmContiGetAbsRelMode](#),
[AxmContiBeginNode](#), [AxmContiEndNode](#), [AxmContiReadFree](#), [AxmContiReadIndex](#), [AxmContiWriteClear](#),
[AxmContiStart](#), [AxmContiIsMotion](#), [AxmContiGetTotalNodeNum](#), [AxmContiSetOptionNodeNum](#)

AxmContiGetTotalNodeNum

Purpose

To identify total number of index of specified continuous interpolation move.

Format

C

```
DWORD AxmContiGetTotalNodeNum (long lCoord, long *lpNodeNum);
```

Visual Basic

```
Function AxmContiGetTotalNodeNum (ByVal lCoord As Long, ByRef lpNodeNum As Long) As Long
```

Delphi

```
function AxmContiGetTotalNodeNum (lCoord: LongInt; lpNodeNum : PLongInt) : Dword; stdcall;
```

Input / Output

Name	in/out	Format	Init Value	Explanation
Coord	in	long		Coordinate system number(start from 0)
NodeNum	out	long*		Total number of continuous interpolation index

Return AXT_RT_SUCCESS(0000) : Successful execution of API.

[* See error code Table for more information on status error codes](#)

Description

When using [AxmLineMove](#), [AxmCircleCenterMove](#), [AxmCirclePointMove](#), [AxmCircleRadiusMove](#), [AxmCircleAngleMove](#) API with [AxmContiBeginNode](#) and [AxmContiEndNode](#), it stores tasks to queue and processes continuous interpolation to process interpolation move. One [AxmLineMove](#) API is regarded as one unit.

This API identifies the total number of nodes of continuous interpolation move specified here.

C Example

```
long lpTotalNodeNum1;
long lpNodeNum1;
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;

AxmContiWriteClear(lCoordinate);
AxmContiSetAxisMap (lCoordinate, lPosSize, lAxis);
//Register absolute position move
AxmContiSetAbsRelMode(lCoordinate, 0);

AxmContiBeginNode(lCoordinate);
dPos[0] = 100 , dPos[1] = 100;
```

```

AxmLineMove(lCoordinate, dPos, dVelocity, dAccel, dAccel);
dPos[0] = 200 , dPos[1] = 200;
AxmLineMove(lCoordinate, dPos, dVelocity, dAccel, dAccel);
dPos[0] = 300 , dPos[1] = 300;
AxmLineMove(lCoordinate, dPos, dVelocity, dAccel, dAccel);
dPos[0] = 400 , dPos[1] = 400;
AxmLineMove(lCoordinate, dPos, dVelocity, dAccel, dAccel);
dPos[0] = 500 , dPos[1] = 500;
AxmLineMove(lCoordinate, dPos, dVelocity, dAccel, dAccel);
AxmContiEndNode (lCoordinate);

//Start absolute position move
AxmContiSetAbsRelMode(lCoordinate, 0);
// Start continuous interpolation move
AxmContiStart (lCoordinate, 0, 0);

Sleep(100);

AxmContiIsMotion (lCoordinate, &IsContMotion);
if(IsContMotion ){ // Identify if interpolation is in motion or not..
AxmContiGetTotalNodeNum (0, &lpTotalNodeNum1);
printf("\t Total number of currently operating list interpolation move
      is %ld\n", lpTotalNodeNum1);
}

```

VB Example

```

Public Declare Sub Sleep Lib "kernel32" Alias "Sleep" (ByVal
    dwMilliseconds As Long)
'In basic, header file should be declared to use Sleep function.
Dim IsContMotion, lpTotalNodeNum1 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

AxmContiWriteClear lCoordinate
AxmContiSetAxisMap lCoordinate, lPosSize, axes(0)
' Register absolute position move
AxmContiSetAbsRelMode lCoordinate, 0

AxmContiBeginNode lCoordinate
dPos(0) = 100: dPos(1) = 100
AxmLineMove lCoordinate, dPos(0), dVelocity, dAccel, dAccel
dPos(0) = 200: dPos(1) = 200
AxmLineMove lCoordinate, dPos(0), dVelocity , dAccel, dAccel
dPos(0) = 300: dPos(1) = 300
AxmLineMove lCoordinate, dPos(0), dVelocity , dAccel, dAccel
dPos(0) = 400: dPos(1) = 400
AxmLineMove lCoordinate, dPos(0), dVelocity , dAccel, dAccel
dPos(0) = 500: dPos(1) = 500
AxmLineMove lCoordinate, dPos(0), dVelocity , dAccel, dAccel
AxmContiEndNode lCoordinate

```

```

` Start register absolute position move
AxmContiSetAbsRelMode lCoordinate, 0
` Start continuous interpolation move
AxmContiStart lCoordinate, 0, 0

Sleep 100
AxmContiIsMotion lCoordinate, IsContMotion
If(IsContMotion = 1) Then
AxmContiGetTotalNodeNum 0, lpTotalNodeNum1
strData = "Total number of currently operating list interpolation move" +
          CStr(lpTotalNodeNum1) + " is..."
MsgBox strData
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;
  lpTotalNodeNum1: longInt;
  strData : String;

begin

  dVelocity := 200;
  dAccel := 400;
  lPosSize := 2;
  lCoordinate := 0;

  // Register absolute position move
  AxmContiSetAbsRelMode(lCoordinate, 0);

  for i := 0 to 1 do
  begin
    lAxis[i] := i;
  end;

  AxmContiWriteClear(lCoordinate);
  AxmContiSetAxisMap (lCoordinate, 2, @lAxis);
  AxmContiBeginNode(lCoordinate);
  dPos[0] := 100 ; dPos[1] := 100;
  AxmLineMove(lCoordinate, @dPos, dVelocity, dAccel, dAccel);
  dPos[0] := 200 ; dPos[1] := 200;
  AxmLineMove(lCoordinate, @dPos, dVelocity, dAccel, dAccel);
  dPos[0] := 300 ; dPos[1] := 300;
  AxmLineMove(lCoordinate, @dPos, dVelocity, dAccel, dAccel);
  dPos[0] := 400 ; dPos[1] := 400;
  AxmLineMove(lCoordinate, @dPos, dVelocity, dAccel, dAccel);
  dPos[0] := 500 ; dPos[1] := 500;
  AxmLineMove(lCoordinate, @dPos, dVelocity, dAccel, dAccel);
  AxmContiEndNode (lCoordinate);

  // Start absolute position move
  AxmMotSetAbsRelMode(lCoordinate, 0);
  { Start continuous interpolation move }
  AxmContiStart (lCoordinate, 0, 0);
  Sleep(100);
  AxmContiIsMotion (lCoordinate, @IsContMotion);
  if(IsContMotion = 1) then { Identify if interpolation is in motion or
                           not..}

```

```
begin
AxmContiGetTotalNodeNum (0, @ lpTotalNodeNum1);
strData    := 'Total number of currently operating list interpolation
               move' + IntToStr(lpTotalNodeNum1) + 'is..';
Application.MessageBox (PCHAR(strData), 'Ajinextek', MB_OK);

End ;
end;
```

See Also

[AxmContiSetAxisMap](#), [AxmContiGetAxisMap](#), [AxmContiSetAbsRelMode](#), [AxmContiGetAbsRelMode](#),
[AxmContiBeginNode](#), [AxmContiEndNode](#), [AxmContiReadFree](#), [AxmContiReadIndex](#), [AxmContiWriteClear](#),
[AxmContiStart](#), [AxmContiIsMotion](#), [AxmContiGetNodeNum](#),

Trigger API

Function	Description
<u>AxmTriggerSetTimeLevel</u>	Set trigger signal duration, trigger output level, and trigger output method in specified axis.
<u>AxmTriggerGetTimeLevel</u>	Return trigger signal duration, trigger output level, and trigger output method in specified axis.
<u>AxmTriggerSetAbsPeriod</u>	Set whether trigger function will be used or not, output level, position comparater, trigger signal duration, and trigger output mode in specified axis.
<u>AxmTriggerGetAbsPeriod</u>	Return whether trigger function will be used or not, output level, position comparater, trigger signal duration, and trigger output mode in specified axis.
<u>AxmTriggerSetBlock</u>	Output trigger for regular section from start point to endpoint which user set.
<u>AxmTriggerGetBlock</u>	Read trigger setting of ' <u>AxmTriggerSetBlock</u> ' API.
<u>AxmTriggerOneShot</u>	User outputs one trigger pulse.
<u>AxmTriggerSetTimerOneshot</u>	User outputs one trigger pulse after few seconds.
<u>AxmTriggerOnlyAbs</u>	Output absolute position trigger infinite absolute position.
<u>AxmTriggerSetReset</u>	Reset trigger setting.

Trigger signal use

During motion move, AXM can send trigger signal of 5V level as well as interrupt for certain situation. Trigger signal is from signal pin, and when its setting is ActiveHIGH, it maintains 0V for normal time, 5V for trigger signal, and back to 0V. For ActiveLow it works in opposite way. Of course, it maintains 24V for mechanical part. Before making trigger signal, you may set things like trigger signal level and duration by [AxmTriggerSetTimeLevel](#) API and you may call functions making trigger for certain condition. Trigger occurring APIs include [AxmTriggerSetAbsPeriod](#) / [AxmTriggerOnlyAbs](#) API which make trigger for certain point, [AxmTriggerSetBlock](#) API which makes trigger regularly during certain period, and [AxmTriggerOneShot](#) / [AxmTriggerSetTimerOneshot](#) API which make triggers together at any time.

AxmTriggerSetTimeLevel

Purpose

To set trigger output time, trigger level, trigger output way, and whether using interrupt setting or not when trigger output for specific axis.

Format

C

```
DWORD AxmTriggerSetTimeLevel (long lAxisNo, double dTrigTime, DWORD uTriggerLevel, DWORD uSelect, DWORD ulInterrupt);
```

Visual Basic

```
Function AxmTriggerSetTimeLevel (ByVal lAxisNo As Long, ByVal dTrigTime As Double, ByVal uTriggerLevel As Long, ByVal uSelect As Long ,ByVal ulInterrupt As Long) As Long
```

Delphi

```
function AxmTriggerSetTimeLevel (lAxisNo : LongInt; dTrigTime : Double; uTriggerLevel : Dword; uSelect: Dword :ulInterrupt: Dword) : Dword; stdcall;
```

Input / Output

Name	in/out	Format	Init Value	Explanation
AxisNo	in	long		Channel(axis) number(start from 0)
TrigTime	In	double		Trigger output signal time IP:minimum(1) – maximum(4000) is possible (1 usec) – (4 msec) QI: 1 – 50000 (50msec)
TriggerLevel	In	DWORD		Set whether using trigger output signal or not and Active Level
Select	In	DWORD		Selects trigger standard position input source
Interrupt	in	DWORD		Whether to output interrupt when output trigger

TriggerLevel

#define	Value	Explanation
LOW	00h	B Contact(NORMAL CLOSE) Use Active Low Level Trigger Signal
HIGH	01h	A Contact(NORMAL OPEN) Use Active High Level Trigger signal
UNUSED	02h	Trigger signal not in use
USED	03h	Trigger signal in use(maintain current status)

Select

#define	Value	Explanation
COMMAND	00h	Command position
ACTUAL	01h	Actual position

ulInterrupt

uUse	Value	Explanation
DISABLE	00h	interrupt not it use
ENABLE	01h	interrupt in use

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

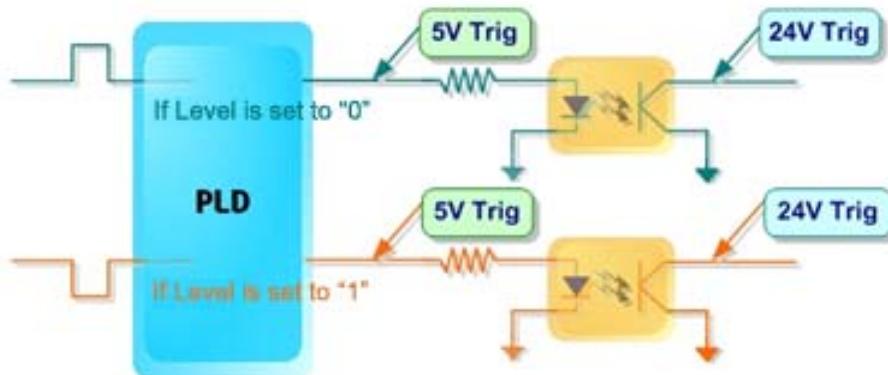
Description

Low(B contact) : normal Close → detected Open

High(A contact): normal Open → detected Close

Falling Edge : maintains high status for normal time, low status for trigger period, and then goes back to high status.

Rising Edge : maintains Low status for normal time, high status for trigger period, and then goes back to low status.



* Note : In the case of 5V trigger output, it maintains 0V for normal time and 5V if trigger occurs when active level is set to 0. In the case of 24V trigger, it becomes N24V for trigger output in Open status. Thus, when using trigger with 24V with terminal strip, you need to put pull-up resistance of 2kΩ to identify trigger output.

dTrigTime : trigger output time

SMC-2V03: 1usec – maximum 4msec (set between 1 – 4000)

PCI-N804/404: 1usec – maximum 50msec (set between 1 – 50000)

C Example

```
// Set trigger output time, level, output method, and output interrupt of
// trigger function of axis 0
```

```

AxmTriggerSetTimeLevel(0, 4000, HIGH, 0, 1);

double dTrigTime;
DWORD dwTriggerLevel;
DWORD dwSelect;
DWORD dwInterrupt;

// return trigger output time, level, output method, and output interrupt
// of trigger function of axis 0.
AxmTriggerGetTimeLevel(0, &dTrigTime, &dwTriggerLevel, &dwSelect, &
dwInterrupt);

```

VB Example

```

' Set trigger output time, level, output method, and output interrupt of
' trigger function of axis 0
AxmTriggerSetTimeLevel 0, 4000, HIGH, 0, 1

Dim dTrigTime As Double
Dim dwTriggerLevel, dwSelect, dwInterrupt As Long
' return trigger output time, level, output method, and output interrupt of
' trigger function of axis 0.
AxmTriggerGetTimeLevel 0, dTrigTime, dwTriggerLevel, dwSelect, dwInterrupt

```

Delphi Example

```

var
  dwTriggerLevel, dwSelect, dwInterrupt : Dword;
  dTrigTime : Double;

Begin
  { Set trigger output time, level, output method, and output interrupt of
    trigger function of axis 0 }
  AxmTriggerSetTimeLevel(0, 4000, HIGH, 0, 1);

  { Return trigger output time, level, output method, and output interrupt
    of trigger function of axis 0.}
  AxmTriggerGetTimeLevel(0, @dTrigTime, @dwTriggerLevel, @dwSelect,
  @dwInterrupt);
End;

```

See Also

[AxmTriggerGetTimeLevel](#), [AxmTriggerSetAbsPeriod](#), [AxmTriggerGetAbsPeriod](#), [AxmTriggerSetBlock](#),
[AxmTriggerGetBlock](#), [AxmTriggerOneShot](#), [AxmTriggerOnlyAbs](#), [AxmTriggerSetReset](#)

AxmTriggerGetTimeLevel

Purpose

Return trigger output time, trigger level, trigger output method, and whether to set output interrupt or not when outputting trigger.

Format

C

```
DWORD AxmTriggerGetTimeLevel (long IAxisNo, double *dpTrigTime, DWORD *upTriggerLevel, DWORD *upSelect,
DWORD *upInterrupt);
```

Visual Basic

```
Function AxmTriggerGetTimeLevel (ByVal IAxisNo As Long, ByVal dpTrigTime As Double, ByRef upTriggerLevel As
Long, ByRef upSelect As Long, ByRef upInterrupt As Long) As Long
```

Delphi

```
function AxmTriggerGetTimeLevel (IAxisNo : LongInt; dpTrigTime : Pdouble; upTriggerLevel : PDWord; upSelect :
PDWord; upInterrupt : PDWord) : Dword; stdcall;
```

Input / Output

Name	in/out	Format	Init Value	Explanation
AxisNo	in	long		Channel(axis) number(start from 0)
TrigTime	out	double*		Trigger output signal time IP:minimum(1) – maximum(4000) is possible (1 usec) – (4 msec)
TriggerLevel	out	DWORD*		Set whether using trigger output signal and Active Level
Select	out	DWORD*		Selects trigger standard position input source
Interrupt	out	DWORD*		Whether to output interrupt or not when outputting trigger

TriggerLevel

#define	Value	Explanation
LOW	00h	B Contact(NORMAL CLOSE) Use Active Low Level Trigger Signal
HIGH	01h	A Contact(NORMAL OPEN) Use Active High Level Trigger signal
UNUSED	02h	Trigger signal not it use
USED	03h	Trigger signal in use (maintain current status)

Select

#define	Value	Explanation
COMMAND	00h	Command position
ACTUAL	01h	Actual position

ulInterrupt

uUse	Value	Explanation
DISABLE	00h	interrupt not in use
ENABLE	01h	interrupt in use

Return AXT_RT_SUCCESS(0000) : Successful execution of API.

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

Description

cf : If you want to bring only necessary data, you can put NULL for unnecessary variable.

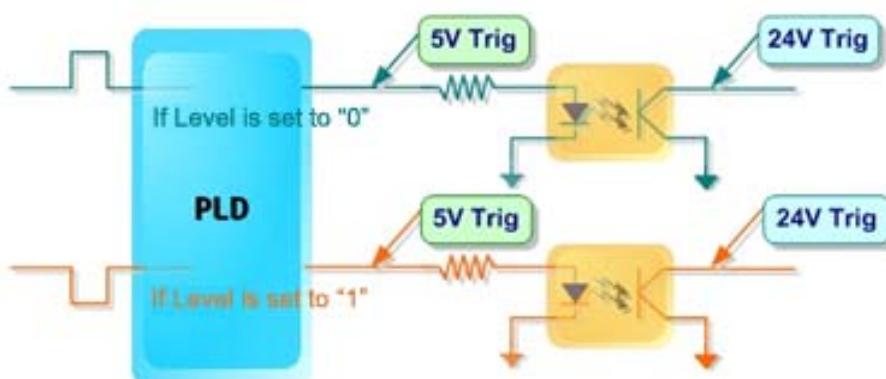
It returns trigger output time, trigger level, trigger output method, and whether to set interrupt or not when outputting trigger specified by '[AxmTriggerSetTimeLevel](#)'.

Low(B contact) : normal Close → detected Open

High(A contact) : normal Open → detected Close

Falling Edge : maintains high status for normal time, low status for trigger period, and then goes back to high status.

Rising Edge : maintains low status for normal time, high status for trigger period, and then goes back to low status.



* Note : In the case of 5V trigger output, it maintains 0V for normal time and 5V when trigger occurs if active level is set to 0. In the case of 24V trigger, it becomes N24V for trigger output in Open status. Thus, when using trigger with 24V with terminal strip, you need to put pull-up resistance of 2kΩ to identify trigger output.

C Example

```
// Set trigger output time, level, output method, and output interrupt of
// trigger function of axis 0
AxmTriggerSetTimeLevel (0, 4000, HIGH, 0, 1);

double dTrigTime;
DWORD dwTriggerLevel;
DWORD dwSelect;
DWORD dwInterrupt;
```

```
// return trigger output time, level, output method, and output interrupt
// of trigger function of axis 0.
AxmTriggerGetTimeLevel(0, &dTrigTime, &dwTriggerLevel, &dwSelect, &
dwInterrupt);
```

VB Example

```
'Set trigger output time, level, output method, and output interrupt of
// trigger function of axis 0
AxmTriggerSetTimeLevel 0, 4000, HIGH, 0, 1

Dim dTrigTime As Double
Dim dwTriggerLevel, dwSelect, dwInterrupt As Long
//return trigger output time, level, output method, and output interrupt of
// trigger function of axis 0.
AxmTriggerGetTimeLevel 0, dTrigTime, dwTriggerLevel, dwSelect, dwInterrupt
```

Delphi Example

```
var
dwTriggerLevel, dwSelect, dwInterrupt : Dword;
dTrigTime : Double;

Begin

{ Set trigger output time, level, output method, and output interrupt of
// trigger function of axis 0 }
AxmTriggerSetTimeLevel (0, 4000, HIGH, 0, 1);

{ Return trigger output time, level, output method, and output interrupt
// of trigger function of axis 0.}
AxmTriggerGetTimeLevel(0, @dTrigTime, @dwTriggerLevel, @dwSelect,
@dwInterrupt);
End;
```

See Also

[AxmTriggerSetTimeLevel](#), [AxmTriggerSetAbsPeriod](#), [AxmTriggerGetAbsPeriod](#), [AxmTriggerSetBlock](#),
[AxmTriggerGetBlock](#), [AxmTriggerOneShot](#), [AxmTriggerSetTimerOneshot](#), [AxmTriggerOnlyAbs](#),
[AxmTriggerSetReset](#)

AxmTriggerSetAbsPeriod

Purpose

Set trigger function for specific axis to period, absolute position mode.

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		Channel(axis) number(start from 0)
Method	In	DWORD		trigger output mode
Pos	In	double		Position value to output trigger signal (When using period: trigger outputs in each position)

Method

#define	Value	Explanation
PERIOD_MODE	00h	Period trigger method using trigger position value.
ABS_POS_MODE	01h	Trigger occurrence in trigger absolute position

Return AXT_RT_SUCCESS(0000) : Successful execution of API.

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

Description

‘[AxmTriggerSetTimeLevel](#)’ trigger function’s trigger output time, level, output method, and whether to set interrupt for output or not are specified. You always have to use with this API. It will be in different status like following according to trigger output level.

Caution: Always set to UNIT/PULSE when setting trigger position.

If the position is set less than UNIT/PULSE, it can not be outputted on that position since min. unit is set to UNIT/PULSE.

Caution: unlike [AxmTriggerSetBlock](#) API, trigger output is based on the change in counter from the beginning point. When you set trigger mode to period mode, trigger specified will be printed.

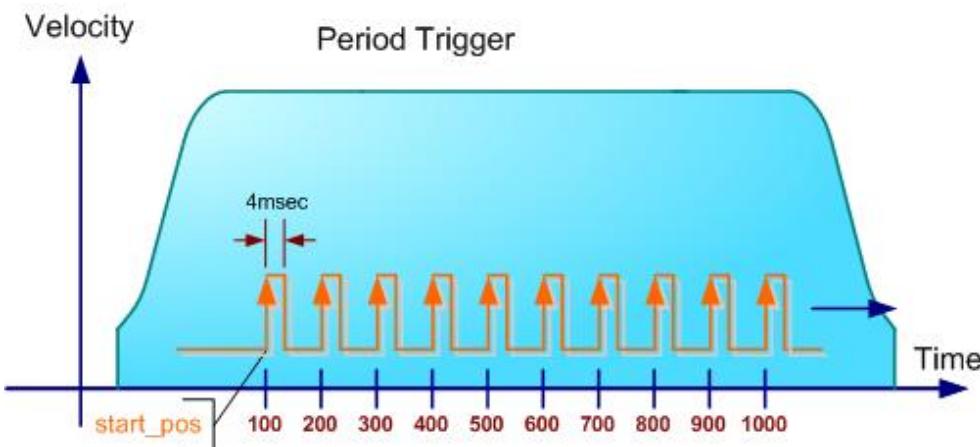
Low(B Contact) : Normal Close → detected Open

High(A Contact) : Normal Open → detected Close

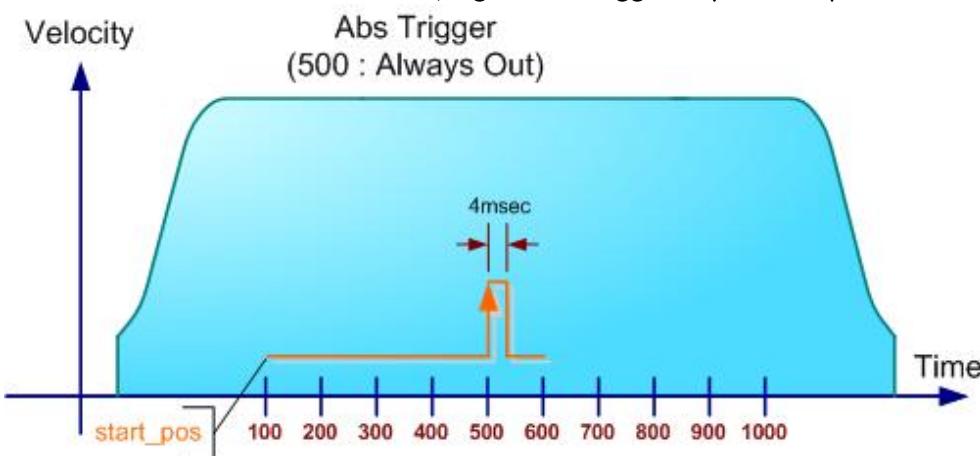
Falling Edge : maintains high status for normal time, low status for trigger period, and then goes back to high status.

Rising Edge : maintains low status for normal time, high status for trigger period, and then goes back to low status..

Period mode : if Position is 100, trigger output will be generated every 100 pulse.



Absolute mode : if Position is 500, it generates trigger output at the position of 500.



C Example

```
// Set trigger output time, level, output method, and output interrupt of
// trigger function of axis 0
AxmTriggerSetTimeLevel (0, 4000, HIGH, 0, 1);
// Set trigger function of axis 0 to period
AxmTriggerSetAbsPeriod (0, PERIOD_MODE, 100);

// Identify whether using trigger function of axis 0 or not
double uPos;
DWORD uMode;

AxmTriggerGetAbsPeriod (0, &uMode, &uPos);
```

VB Example

```
'Set trigger output time, level, output method, and output interrupt of
// trigger function of axis 0
```

```
AxmTriggerSetTimeLevel 0, 4000, HIGH, 0, 1
' Set trigger function of axis 0 to period
AxmTriggerSetAbsPeriod 0, PERIOD_MODE, 100

' Identify whether using trigger function of axis 0 or not
Dim uMode As Long
Dim uPos As Double

AxmTriggerGetAbsPeriod 0, uMode, uPos
```

Delphi Example

```
{ Identify whether using trigger function of axis 0 or not}
var
uMode: DWord;
uPos: Double;

Begin

{ Set trigger output time, level, output method, and output interrupt of
  trigger function of axis 0 }
AxmTriggerSetTimeLevel (0, 4000, HIGH, 0, 1);
{ Set trigger function of axis 0 to period }
AxmTriggerSetAbsPeriod (0, PERIOD_MODE, 100);

AxmTriggerGetAbsPeriod (0, @uMode, @uPos);
end;
```

See Also

[AxmTriggerSetTimeLevel](#), [AxmTriggerGetTimeLevel](#), [AxmTriggerGetAbsPeriod](#), [AxmTriggerSetBlock](#),
[AxmTriggerGetBlock](#), [AxmTriggerOneShot](#), [AxmTriggerSetTimerOneshot](#), [AxmTriggerOnlyAbs](#),
[AxmTriggerSetReset](#)

AxmTriggerGetAbsPeriod

Purpose

Return period, absolute position mode of trigger function for specific axis.

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		Channel(axis) number(start from 0)
Method	Out	DWORD*		trigger output mode
Pos	Out	double*		Position value to output trigger signal (When using period: trigger outputs in each position)

Method

#define	Value	Explanation
PERIOD_MODE	00h	Period trigger method using trigger position value.
ABS_POS_MODE	01h	Trigger occurrence in trigger absolute position

Return AXT_RT_SUCCESS(0000) : Successful execution of API.

[* See error code Table for more information on status error codes](#)

Description

cf : If you only want to bring necessary data, you can put NULL for unnecessary variable.

Return trigger functions in period, absolute position mode to specific axis set by

[‘AxmTriggerSetAbsPeriod’ API.](#)

Caution: In SMC-2V03, when calling AxmTriggerSetBlock API, ABS_POS_MODE is used in internal library, so return value for this API will be 1.

C Example

```
// Set trigger output time, level, output method, and output interrupt of
// trigger function of axis 0
AxmTriggerSetTimeLevel (0, 4000, HIGH, 0, 1);
// Set trigger function of axis 0 to period
```

```

AxmTriggerSetAbsPeriod (0, PERIOD_MODE, 100);

// Identify whether using trigger function of axis 0 or not
double uPos;
DWORD uMode;

AxmTriggerGetAbsPeriod (0, &uMode, &uPos);

```

VB Example

```

' Set trigger output time, level, output method, and output interrupt of
  trigger function of axis 0
AxmTriggerSetTimeLevel 0, 4000, HIGH, 0, 1
' Set trigger function of axis 0 to period
AxmTriggerSetAbsPeriod 0, PERIOD_MODE, 100

' Identify whether using trigger function of axis 0 or not
Dim uMode As Long
Dim uPos As Double

AxmTriggerGetAbsPeriod 0, uMode, uPos

```

Delphi Example

```

{ Identify whether using trigger function of axis 0 or not }
var
uMode: DWord;
uPos: Double;

Begin

{ Set trigger output time, level, output method, and output interrupt of
  trigger function of axis 0 }
AxmTriggerSetTimeLevel (0, 4000, HIGH, 0, 1);
{ Set trigger function of axis 0 to period }
AxmTriggerSetAbsPeriod (0, PERIOD_MODE, 100);

AxmTriggerGetAbsPeriod (0, @uMode, @uPos);
end;

```

See Also

[AxmTriggerSetTimeLevel](#), [AxmTriggerGetTimeLevel](#), [AxmTriggerSetAbsPeriod](#), [AxmTriggerSetBlock](#),
[AxmTriggerGetBlock](#), [AxmTriggerOneShot](#), [AxmTriggerSetTimerOneshot](#), [AxmTriggerOnlyAbs](#),
[AxmTriggerSetReset](#)

AxmTriggerSetBlock

Purpose

Set trigger output for regular intervals between starting point and end point which user specified.

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		Channel(axis) number(start from 0)
StartPos	In	double		Starting position of trigger signal
EndPos	In	double		Ending position of trigger signal
PeriodPos	In	double		Set trigger output period position

Return AXT_RT_SUCCESS(0000) : Successful execution of API.

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

Description

'[AxmTriggerSetTimeLevel](#)' trigger function's trigger output time, level, output method, and whether to set interrupt for output or not are specified. You always have to use with this API. It will be in different status like following according to trigger output level.

Caution: Always set to UNIT/PULSE when setting trigger position.

If position is set less than UNIT/PULSE, it can not be outputted on that position since min. unit is set to UNIT/PULSE.

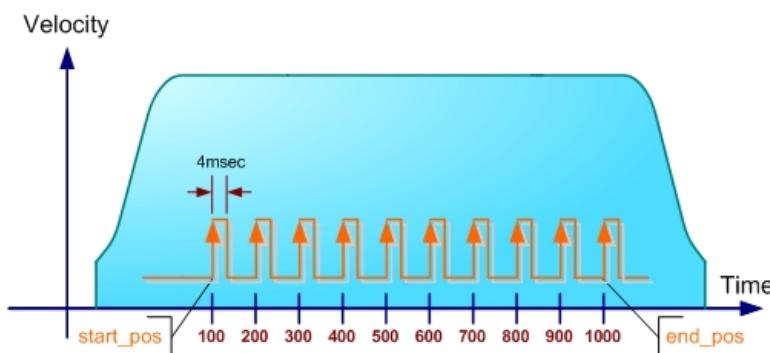
In case of SMC-2V03 module, trigger does not occur if it does not pass the trigger starting point. Also, trigger does not occur if it passes the trigger end point and enters into the range of trigger.

Low(B Contact) : Normal Close -> detected Open

High(A Contact) : Normal Open -> detected Close

Falling Edge : maintains high status for normal time, low status for trigger period, and then goes back to high status.

Rising Edge : maintains low status for normal time, high status for trigger period, and then goes back to low status..



C Example

```
// Set trigger output time, level, output method, and output interrupt of
// trigger function of axis 0
AxmTriggerSetTimeLevel (0, 4000, HIGH, 0, 1);
// trigger output of 4msec for every 100 pulse between 100 and 1050, level
// low setting, interrupt enable.
AxmTriggerSetBlock (0, 100, 1050, 100);

// Read trigger setting of AxmSetBlockTrigger API in axis 0
double dpStartPos, dpEndPos, dpPeriodPos;

AxmTriggerGetBlock (0, &dpStartPos, &dpEndPos, &dpPeriodPos);
```

VB Example

```
'Set trigger output time, level, output method, and output interrupt of
'trigger function of axis 0
AxmTriggerSetTimeLevel 0, 4000, HIGH, 0, 1
'Trigger output of 4msec for every 100 pulse between 100 and 1050, level
'low setting, interrupt enable.
AxmTriggerSetBlock 0, 100, 1050, 100

'Read trigger setting of AxmSetBlockTrigger API in axis 0
Dim dpStartPos, dpEndPos, dpPeriodPos As Double

AxmTriggerGetBlock 0, dpStartPos, dpEndPos, dpPeriodPos
```

Delphi Example

```
{ Read trigger setting of AxmSetBlockTrigger API in axis 0 }
var
dpStartPos, dpEndPos, dpPeriodPos : Double;

Begin
{ Set trigger output time, level, output method, and output interrupt of
  trigger function for axis 0 }
(0, 4000, HIGH, 0, 1);
{ Trigger output of 4msec for every 100 pulse between 100 and 1050, level
  low setting, interrupt enable.}
AxmTriggerSetBlock (0, 100, 1050, 100);

AxmTriggerGetBlock (0, @dpStartPos, @dpEndPos, @dpPeriodPos);
end;
```

See Also

[AxmTriggerSetTimeLevel](#), [AxmTriggerGetTimeLevel](#), [AxmTriggerSetAbsPeriod](#), [AxmTriggerGetAbsPeriod](#),
[AxmTriggerGetBlock](#), [AxmTriggerOneShot](#), [AxmTriggerSetTimerOneshot](#), [AxmTriggerOnlyAbs](#), [AxmTriggerSetReset](#)

AxmTriggerGetBlock

Purpose

Return trigger output setting for regular intervals from start point to end point which user specified.

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		Channel(axis) number(start from 0)
StartPos	out	double*		Starting position of trigger signal
EndPos	out	double*		Ending position of trigger signal
PeriodPos	out	double*		Set trigger output period position

Return AXT_RT_SUCCESS(0000) : Successful execution of API.

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

Description

cf : If you only want to bring necessary data, you can put NULL for unnecessary variable.
[‘AxmTriggerSetBlock’](#) ‘ return specified trigger setting.

C Example

```
// Set trigger output time, level, output method, and output interrupt of
// trigger function of axis 0
AxmTriggerSetTimeLevel (0, 4000, HIGH, 0, 1);
// Trigger output of 4msec for every 100 pulse between 100 and 1050, level
// low setting, interrupt enable.
AxmTriggerSetBlock (0, 100, 1050, 100);

// Read trigger setting of AxmSetBlockTrigger API in axis 0
double dpStartPos, dpEndPos, dpPeriodPos;

AxmTriggerGetBlock (0, &dpStartPos, &dpEndPos, &dpPeriodPos);
```

VB Example

```
'Set trigger output time, level, output method, and output interrupt of
trigger function of axis 0
AxmTriggerSetTimeLevel 0, 4000, HIGH, 0, 1
'Trigger output of 4msec for every 100 pulse between 100 and 1050, level
low setting, interrupt enable.
AxmTriggerSetBlock 0, 100, 1050, 100

'Read trigger setting of AxmSetBlockTrigger API in axis 0
Dim dpStartPos, dpEndPos, dpPeriodPos As Double

AxmTriggerGetBlock 0, dpStartPos, dpEndPos, dpPeriodPos
```

Delphi Example

```
{ Read trigger setting of AxmSetBlockTrigger API in axis 0 }
var
dpStartPos, dpEndPos, dpPeriodPos : Double;

Begin
{ Set trigger output time, level, output method, and output interrupt of
trigger function of axis 0 }
AxmTriggerSetTimeLevel (0, 4000, HIGH, 0, 1);
{ Trigger output of 4msec for every 100 pulse between 100 and 1050, level
low setting, interrupt enable.}
AxmTriggerSetBlock (0, 100, 1050, 100);

AxmTriggerGetBlock (0, @dpStartPos, @dpEndPos, @dpPeriodPos);
end;
```

See Also

[AxmTriggerSetTimeLevel](#), [AxmTriggerGetTimeLevel](#), [AxmTriggerSetAbsPeriod](#), [AxmTriggerGetAbsPeriod](#),
[AxmTriggerSetBlock](#), [AxmTriggerOneShot](#), [AxmTriggerSetTimerOneshot](#), [AxmTriggerOnlyAbs](#),
[AxmTriggerSetReset](#)

AxmTriggerOneShot

Purpose

Output one trigger pulse at the position specified by user

Format

C

```
DWORD AxmTriggerOneShot (long IAxisNo);
```

Visual Basic

```
Function AxmTriggerOneShot (ByVal IAxisNo As Long) As Long
```

Delphi

```
function AxmTriggerOneShot (IAxisNo : LongInt) : Dword; stdcall;
```

Input / Output

Name	in/out	Format	Init Value	Explanation
AxisNo	in	long		Channel(axis) number(start from 0)

Return AXT_RT_SUCCESS(0000) : Successful execution of API.

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

Description

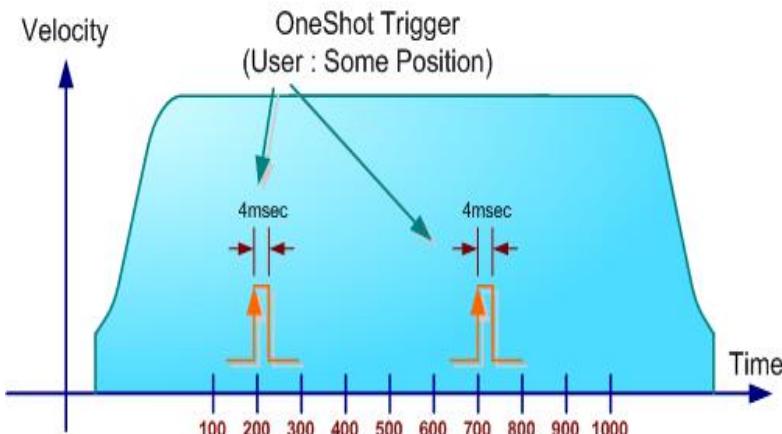
'[AxmTriggerSetTimeLevel](#)' trigger function's trigger output time, level, output method, and whether to set interrupt for output are specified. You always have to use it with this API. It will be in different status like following according to trigger output level.

Low(B Contact) : Normal Close → detected Open

High(A Contact) : Normal Open → detected Close

Falling Edge : maintains high status for normal time, low status for trigger period, and then goes back to high status.

Rising Edge : maintains low status for normal time, high status for trigger period, and then goes back to low status..



C Example

```
// Set trigger output time, level, output method, and output interrupt of  
// trigger function of axis 0  
AxmTriggerSetTimeLevel (0, 4000, HIGH, 0, 1);  
// Compulsorily output one trigger at axis 0.  
AxmTriggerOneShot (0);
```

VB Example

```
'Set trigger output time, level, output method, and output interrupt of  
// trigger function of axis 0  
AxmTriggerSetTimeLevel 0, 4000, HIGH, 0, 1  
'Compulsorily output one trigger at axis 0.  
AxmTriggerOneShot 0
```

Delphi Example

```
Begin  
{ Set trigger output time, level, output method, and output interrupt of  
// trigger function of axis 0 }  
AxmTriggerSetTimeLevel (0, 4000, HIGH, 0, 1);  
{ Compulsorily output one trigger at axis 0.}  
AxmTriggerOneShot (0);  
End;
```

See Also

[AxmTriggerSetTimeLevel](#), [AxmTriggerGetTimeLevel](#), [AxmTriggerSetAbsPeriod](#), [AxmTriggerGetAbsPeriod](#),
[AxmTriggerSetBlock](#), [AxmTriggerGetBlock](#), [AxmTriggerSetTimerOneshot](#), [AxmTriggerOnlyAbs](#),
[AxmTriggerSetReset](#)

AxmTriggerSetTimerOneshot

Purpose

Output trigger pulse at specific time specified by user after calling API.

Format

C

```
DWORD AxmTriggerSetTimerOneshot (long IAxisNo, long ImSec);
```

Visual Basic

```
Function AxmTriggerSetTimerOneshot (ByVal IAxisNo As Long, ByVal ImSec As Long) As Long
```

Delphi

```
function AxmTriggerSetTimerOneshot (IAxisNo : LongInt; ImSec : LongInt) : Dword; stdcall;
```

Input / Output

Name	in/out	Format	Init Value	Explanation
AxisNo	in	Long		Channel(axis) number(start from 0)
mSec	in	Long		Time setting unit : sec

Return AXT_RT_SUCCESS(0000) : Successful execution of API.

[* See error code Table for more information on status error codes](#)

Description

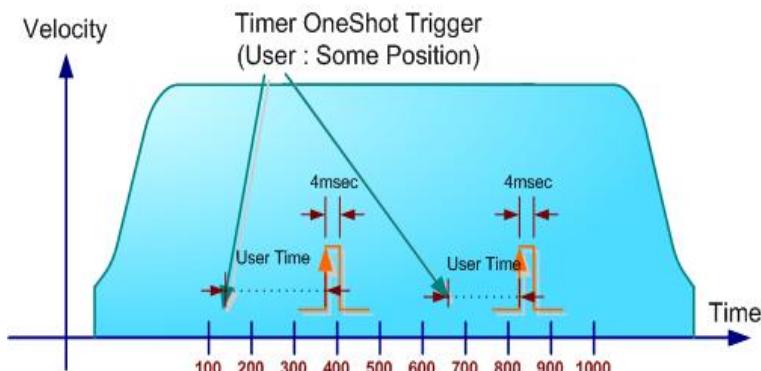
'[AxmTriggerSetTimeLevel](#)' trigger function's trigger output time, level, output method, and whether to set interrupt for output or not are specified. You always have to use with this API. It will be in different status like following according to trigger output level.

Low(B Contact) : Normal Close → detected Open

High(A Contact) : Normal Open → detected Close

Falling Edge : maintains high status for normal time, low status for trigger period, and then goes back to high status.

Rising Edge : maintains low status for normal time, high status for trigger period, and then goes back to low status..



C Example

```
// Set trigger output time, level, output method, and output interrupt of  
// trigger function of axis 0  
  
AxmTriggerSetTimeLevel (0, 4000, HIGH, 0, 1);  
// Compulsorily output one trigger after 1 sec at axis 0.  
AxmTriggerSetTimerOneshot (0, 1000);
```

VB Example

```
'Set trigger output time, level, output method, and output interrupt of  
// trigger function of axis 0  
AxmTriggerSetTimeLevel 0, 4000, HIGH, 0, 1  
'Compulsorily output one trigger after 1 sec at axis 0.  
AxmTriggerSetTimerOneshot 0, 1000
```

Delphi Example

```
Begin  
{ Set trigger output time, level, output method, and output interrupt of  
// trigger function of axis 0 }  
AxmTriggerSetTimeLevel (0, 4000, HIGH, 0, 1);  
{ Compulsorily output one trigger after 1 sec at axis 0.}  
AxmTriggerSetTimerOneshot (0, 1000);  
End;
```

See Also

[AxmTriggerSetTimeLevel](#), [AxmTriggerGetTimeLevel](#), [AxmTriggerSetAbsPeriod](#), [AxmTriggerGetAbsPeriod](#),
[AxmTriggerSetBlock](#), [AxmTriggerGetBlock](#), [AxmTriggerOneShot](#), [AxmTriggerOnlyAbs](#), [AxmTriggerSetReset](#)

AxmTriggerOnlyAbs

Purpose

Output trigger pulse at absolute position specified by user regardless of the number of position.

Format

C

```
DWORD AxmTriggerOnlyAbs (long IAxisNo, long ITrigNum, double* dpTrigPos);
```

Visual Basic

```
Function AxmTriggerOnlyAbs(ByVal IAxisNo As Long, ByVal ITrigNum As Long, ByRef dpTrigPos As Double) As Long
```

Delphi

```
function AxmTriggerOnlyAbs (IAxisNo : LongInt; ITrigNum : LongInt; dpTrigPos : Pdouble) : Dword; stdcall;
```

Input / Output

Name	in/out	Format	Init Value	Explanation
AxisNo	in	long		Channel(axis) number(start from 0)
TrigNum	in	long		Number of trigger output position
TrigPos	in	double*		Trigger position array

Return AXT_RT_SUCCESS(0000) : Successful execution of API.

[* See error code Table for more information on status error codes](#)

Description

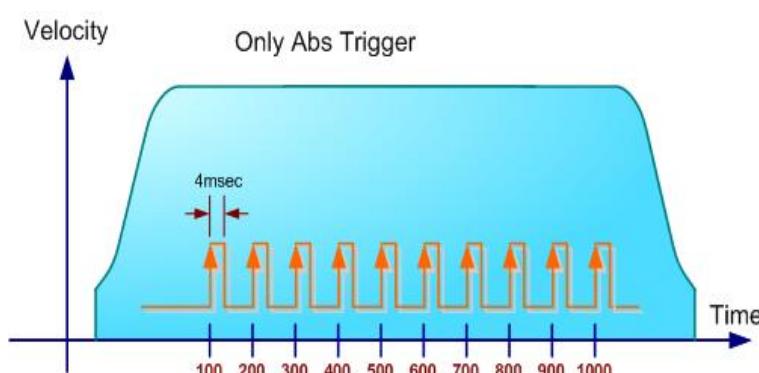
'[AxmTriggerSetTimeLevel](#)' trigger function's trigger output time, level, output method, and whether to set interrupt for output or not are specified. You always have to use with this API. It will be in different status like following according to trigger output level.

Low(B Contact) : Normal Close → detected Open

High(A Contact) : Normal Open → detected Close

Falling Edge : maintains high status for normal time, low status for trigger period, and then goes back to high status.

Rising Edge : maintains low status for normal time, high status for trigger period, and then goes back to low status..



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;

// Set trigger output time, level, output method, and output interrupt of
// trigger function of axis 0
AxmTriggerSetTimeLevel (0, 4000, HIGH, 0, 1);
// Set absolute position of user at axis 0
AxmTriggerOnlyAbs (0, 10, trig_pos); // trig_time: lusec ~ 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

' Set trigger output time, level, output method, and output interrupt of
' trigger function of axis 0

AxmTriggerSetTimeLevel 0, 4000, HIGH, 0, 1
' Set absolute position of user at axis 0.
AxmTriggerOnlyAbs 0, 10, trig_pos(0)      ' trig_time: lusec ~ 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;
  { Set trigger output time, level, output method, and output interrupt of
    trigger function of axis 0 }
  AxmTriggerSetTimeLevel (0, 4000, HIGH, 0, 1);
  { Set absolute position of user at axis 0.}
  AxmTriggerOnlyAbs (0, 10, @trig_pos);   { trig_time: lusec ~ 4000 usec}
End;

```

See Also

[AxmTriggerSetTimeLevel](#), [AxmTriggerGetTimeLevel](#), [AxmTriggerSetAbsPeriod](#), [AxmTriggerGetAbsPeriod](#),
[AxmTriggerSetBlock](#), [AxmTriggerGetBlock](#), [AxmTriggerOneShot](#), [AxmTriggerSetTimerOneshot](#),
[AxmTriggerSetReset](#)

AxmTriggerSetReset

Purpose

Release trigger output specified by user.

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		Channel(axis) number(start from 0)

Return AXT_RT_SUCCESS(0000) : Successful execution of API.

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

Description

Release trigger setting of specified axis set by '[AxmTriggerSetTimeLevel](#)'.

C Example

```
// Release trigger setting of axis 0.
AxmTriggerSetReset(0);
```

VB Example

```
// Release trigger setting of axis 0.
AxmTriggerSetReset 0
```

Delphi Example

```
Begin
// Release trigger setting of axis 0.
AxmTriggerSetReset(0);
End;
```

See Also

[AxmTriggerSetTimeLevel](#), [AxmTriggerGetTimeLevel](#), [AxmTriggerSetAbsPeriod](#),
[AxmTriggerGetAbsPeriod](#), [AxmTriggerSetBlock](#), [AxmTriggerGetBlock](#), [AxmTriggerOneShot](#),
[AxmTriggerSetTimerOneshot](#), [AxmTriggerOnlyAbs](#)

Gantry move function

Gantry means system in which two motor axes are linked by mechanism and controlled like one axis. Generally in gantry system using 2 linear Servo motor, since if there is certain mechanical deviance between two gantries alarm may occur because of overload in Servo motor, method which completely synchronizes Command to both axes is used.

Function	Description
<u>AxmGantrySetEnable</u>	Motion module supports Gantry move system control in which two axes are mechanically linked. When setting master axis to gantry control using this function, slave axis will be synchronized with master axis. After gantry setting is done, all commands like move or stop command to Slave axis will be ignored.
<u>AxmGantryGetEnable</u>	By using this function, return parameters specified when master axis was set to gantry control
<u>AxmGantrySetDisable</u>	When executing home search after gantry control, slave axis conducts home search after master axis conducts home search. This function is used to identify error between master's home and slave's home after home search.

Two axes support gantry move function to control gantry move system which is mechanically linked like the picture below. When master axis(even channel(axis) number(starting from zero)) is set to gantry control using this function, corresponding slave axis will be synchronized with master axis in move. After gantry setting is done, all commands like move or stop command to slave axis will be ignored.



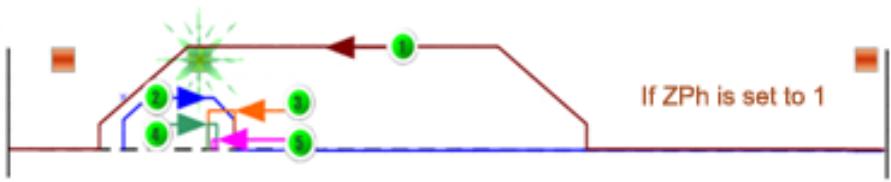
* Note: In the case of gantry system, home sensor of master axis and slave axis will have certain error after they are assembled in the structure where each robot is put paralleled and mechanically linked. When doing home search, error of both home sensors will be compensated in the last step.

It will be done by put assembly error value between sensors on dSIOffset.

The most important thing in gantry control is that master and slave axis should be considered when doing home search. General step will be first finding master axis's home sensor and then finding slave axis's home sensor. The first thing to do is combining two by [AxmGantrySetEnable](#), and then start home move of gantry. Then, you can use it like general home search function.

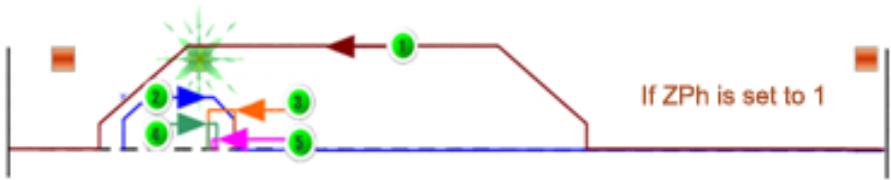
Home search related set variable	Description of variable
HmSig	Set signal to be used as home sensor ex) home sensor(4), +Limit sensor(0), etc.
HmLev	Set active level of home sensor ex) 0: A Contact, 1: B Contact
HmDir	Set initial searching direction when detecting home sensor. ex) 1: (+)direction, 0: (-)direction
Zphas	Whether to detect encoder z phase after detecting home sensor
VelFirst	Initial high speed detecting speed for home search(when initial home sensor was not detected)
VelSecond	Speed for escaping to opposite direction after detecting 1 st sensor in home search.
VelThird	Speed of re-detecting after detecting 1 st sensor in home search
VelLast	Set the final detecting speed in home search[determine detail of home search]
AccFirst	Initial high speed detecting acceleration in home search(Acceleration unit is PPS[Pulses/sec given that Unit/pulse is 1/1])
AccSecond	Acceleration for escaping to opposite direction after detecting 1 st sensor in home search(Acceleration unit is PPS[Pulses/sec given that Unit/pulse is 1/1])
HClrTim	Waiting time before clearing Cmd, Act(Encoder) position after completion of home search
HOffset	Offset value for moving when re-setting mechanical home after completion of home search.
NSWLimit	Set (-) Software Limit which is applied after completion of home search.
PSWLimit	Set (+) Software Limit which is applied after completion of home search. (If NSWLimit is equal to or bigger than PSWLimit value S/W limit will not be set)

1. Find home sensor of Master axis first.



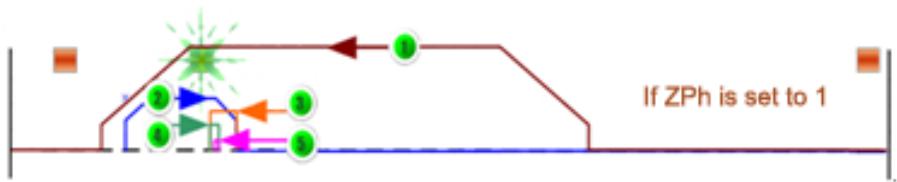
Increase accuracy and speed of home search by detecting Down Edge on Z phase in Step 4 and re-detecting Up Edge on Z phase in Step 5.

2. Clear position after finding home sensor of Slave axis.



Increase accuracy and speed of home search by detecting Down Edge on Z phase in Step 4 and re-detecting Up Edge on Z phase in Step 5.

Understanding speed setting of home search for each step (use same speed for Master, Slave)



1.

Using VelFirst and AccFirst

High Speed Detection of home sensor in HmDir direction and Slow down stop by HAccF deceleration

2.

Using VelSecond, AccSecond

Detection of Down Edge of home sensor in opposite direction of HmDir and Slow down stop by HAccF deceleration

3.

Using VelThird

Detecting Up Edge of home sensor in HmDir direction, and emergency stop.

4.

Using VelThird

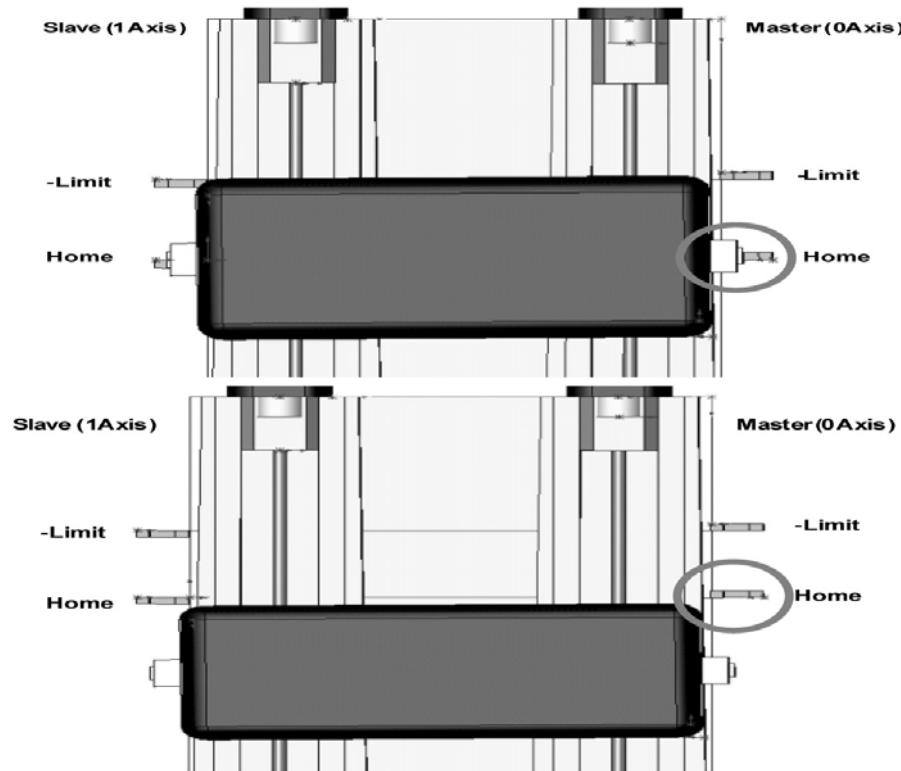
Detecting Down Edge on Z phase in opposite direction of HmDir and emergency stop

5.

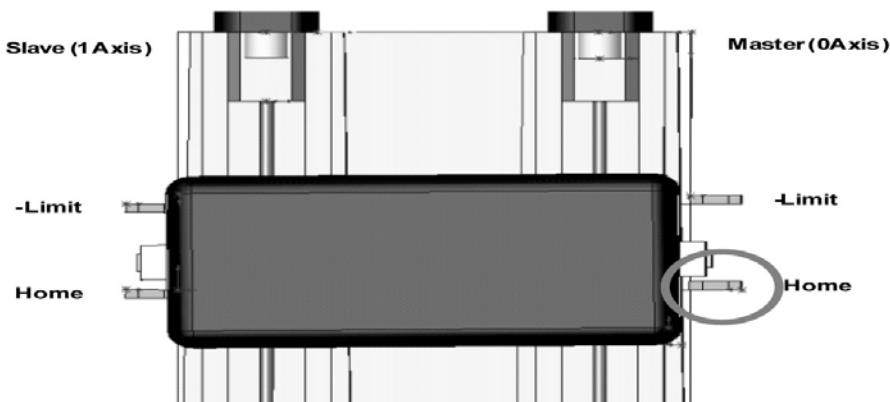
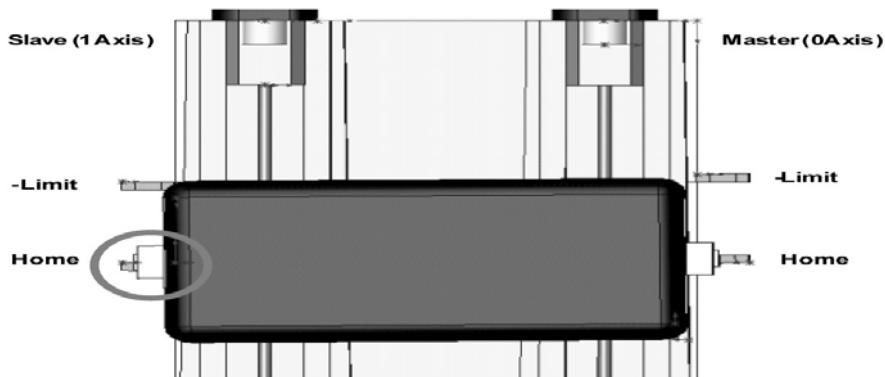
Using VelLast

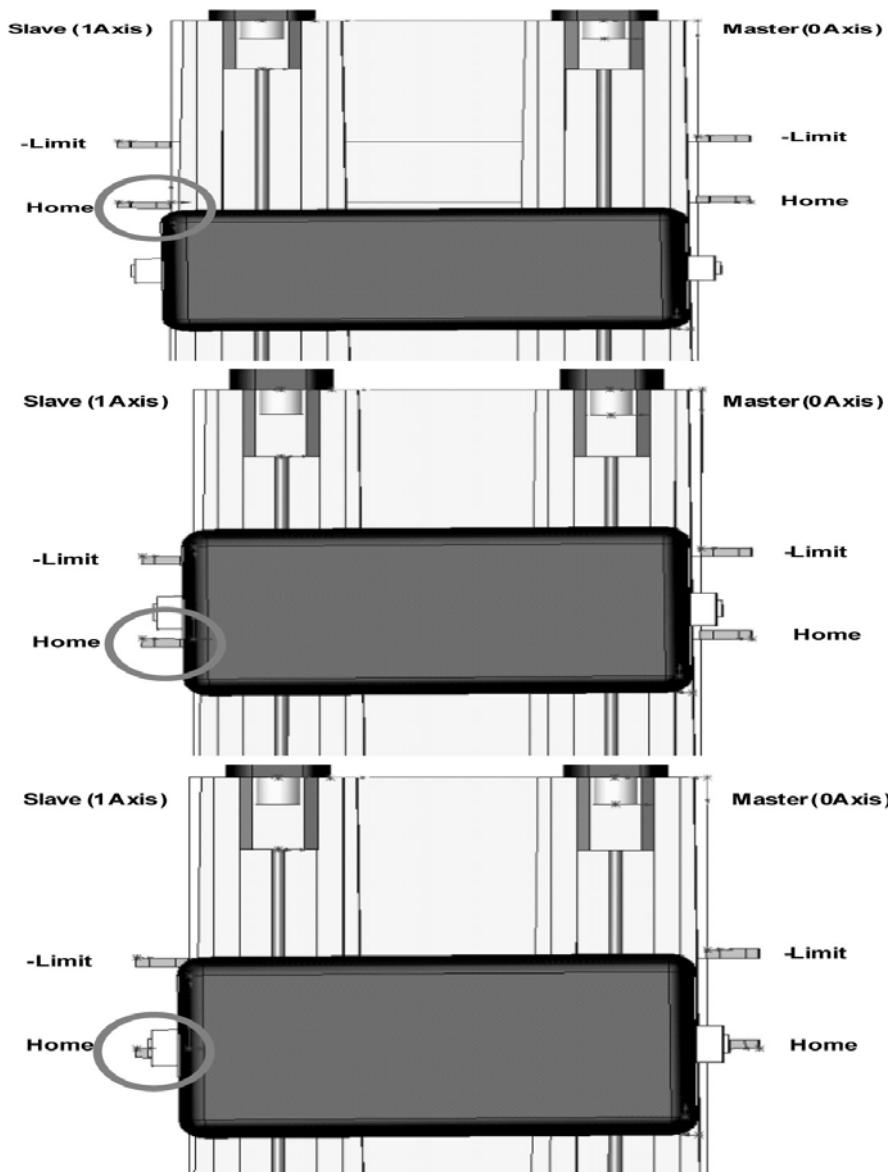
Detecting Up Edge on Z phase in HmDir direction and emergency stop

1. First, find home sensor of Master axis. Set only master axis to position of 0.

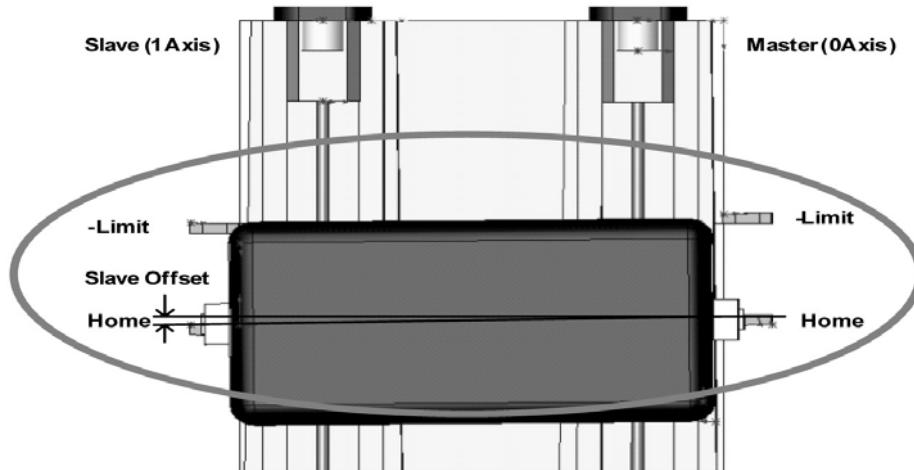


2. Find home sensor of Slave axis. Set only slave axis to position of 0. Then, we can get Offset value of slave against master.

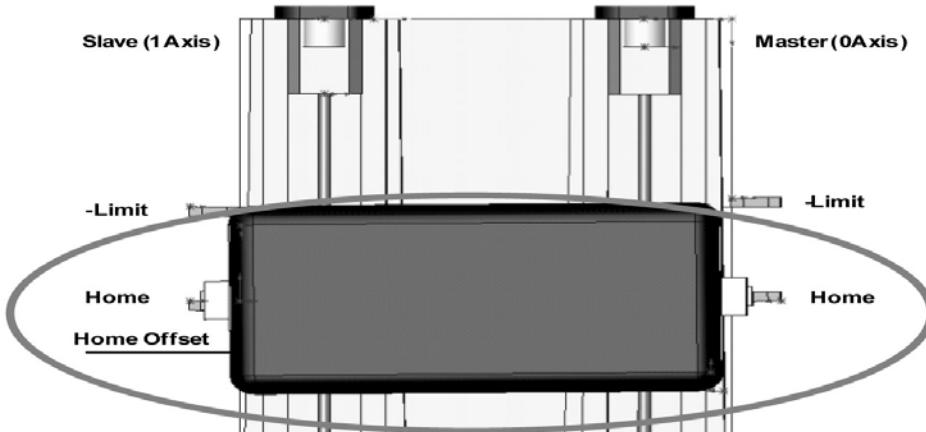




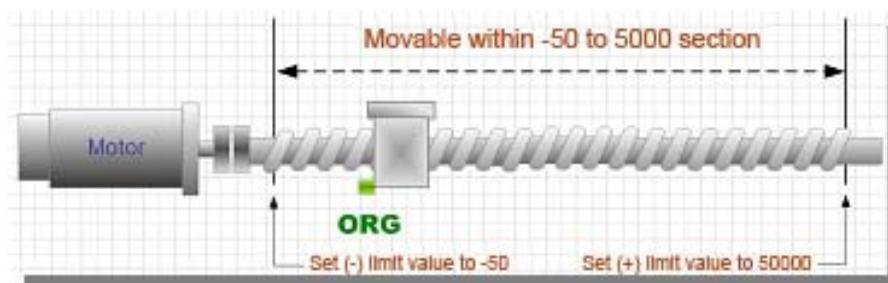
3. Release link between Master and Slave. Move Slave axis by Slave offset. Then send master axis to 0. Link them again.



4. After moving by home offset, make position 0. If there's no home offset, set position to 0 without moving.



5. Software Limit setting and Gantry move



* Caution : Soft limit can be set after home search is successfully processed. Soft limit can be used when limit sensor is not mechanically equiped or limit is to be changed according to situation. One thing to be careful is that PLimit(+) value should be bigger than NLimit(−) for successful setting of Soft Limit. If NSwLimit is same or more than PSwlimit, Soft Limit will not be set.

AxmGantrySetEnable

Purpose

If you set master axis to gantry control by using this API, corresponding Slave axis is synchronized with Master axis. It activates Gantry control API, and all move or stop commands to Slave axis afterwards will be ignored.

Format

C

```
DWORD AxmGantrySetEnable(long lMasterAxisNo, long lSlaveAxisNo, DWORD uSIHomeUse, double dSIOffset,
double dSIOffsetRange);
```

Visual Basic

```
Function AxmGantrySetEnable (ByVal lMasterAxisNo As Long, ByVal lSlaveAxisNo As Long, ByVal uSIHomeUse As
Long, ByVal dSIOffset As Double, ByVal dSIOffsetRange As Double) As Long
```

Delphi

```
function AxmGantrySetEnable (lMasterAxisNo : LongInt; lSlaveAxisNo : LongInt; uSIHomeUse : Dword; dSIOffset :
Double; dSIOffsetRange : Double) : Dword; stdcall;
```

Input/Output

Name	in/out	Format	Init Value	Explanation
MasterAxisNo	in	long		Master channel(axis) number(starting from 0)
SlaveAxisNo	in	long		Slave channel(axis) number(starting from 0)
SIHomeUse	In	DWORD		Whether Slave axis will do home search like Master or not.
SIOffset	In	double		Mechanical error between home sensor of master axis and home sensor of slave axis.
SIOffsetRange	In	double		Specify maximum error value between home sensor of master axis and home sensor of slave axis in home search.

uSIHomeUse

Value	Explanation
00h	Select home search only for Master axis
01h	Identify selecting home search for Master axis and Slave axis together.
02h	Error price confirmation of Master axes and Slave celebration sensors

Return AXT_RT_SUCCESS(0000) : Successful execution of API.

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

Description

If Master axis is set to gantry control by using this API, specified Slave axis will move in synchronization with Master axis. All commands like Move command and stop command after gantry control will be ignored.

Caution for using PCI-N804/404: If gantry setting is ENABLE, slave axis should be identified as InMotion which means it will say True for AxmStatusReadMotion API. If InMotion says False, it means Gantry Enable was not done. In this case, you should check whether alarm or limit occurred.

If home search occurred after gantry control is set, Master axis will do home search, and then Slave axis. This API is used to identify the error between home of master axis and home of slave axis after home search.

uSIHomeUse : whether to use home of slave axis (0 – 2)

(0 : find home of master axis without using home of slave axis.)

(1 : find master axis and slave axis. Conduct correction by applying slave dSIOffset value.)
(2 : find master axis and slave axis. Do not conduct correction by applying slave dSIOffset value.)

Method to identify Offset value of Slave axis.

A. Do servo on for both master and slave.

B. find home by using AxmHomeSetStart API after setting uSIHomeUse = 2 in [AxmGantrySetEnable](#) API.

C. You can see Offset value of master and slave axis by reading Command value of master axis after finding home.

D. Read this Offset value, and put it into dSIOffset parameter of [AxmGantrySetEnable](#) API.

E. When putting dSIOffset value, you have to put opposite sign – dSIOffset because it is the value of slave axis against master axis.

F.dSIOffsetRange means Range of Slave Offset, and it is used to occur error when it gets out of limit by specifying limit of the range.

G. After putting Offset to [AxmGantrySetEnable](#) API, set uSIHomeUse = 1 in [AxmGantrySetEnable](#) API, and then find home by AxmHomeSetStart API.

NOTE : In gantry system, it mechanically links two robots paralleled, and it yields certain value of error in home sensor of Master axis and Slave axis. In the initial equipment setup, after home search is completed, read this error by this API, and set it to dSIOffset. (used only when finding proper Offset value initially.)

► When setting master and slave axis by using [AxmGantrySetEnable](#) API, slave axis should be set to ‘in motion.’ Status value in AxmStatusReadInMotion API is always set to 1. If it can not be set as 1, you can regard it as Enable fail.

►Caution: When using motion board with more than 2 axes (SMC-2V03)

In the board with more than 8 axes by using SMC-2V03, two axes from Axis 0–1 becomes axis group 1, Axis 2–3 becomes axis group 2, Axis 4–5 becomes axis group 3, and Axis 6–7 becomes axis group 4. Axes from the same group can process gantry move, but axes from the different group can’t. The reason is that SMC-2V03 has 2 axes, and it doesn’t support this. Keep in mind when specifying slave and master axis.

►Caution: When using motion board with more than 8 (PCI-N804/404)

In the board with more than 8 axes by using PCI-N804/404, 4 axes from Axis 0~3 becomes axis group 1, 4 axes from Axis 4~7 becomes axis group 2, and 4 axes from Axis 8~11 becomes axis group 3. Axes from the same group can process gantry move, but axes from the different group cannot. The reason is that PCI-N804/404 chip has 4 axes, and it does not support this. Keep in mind when specifying slave and master axis.

C Example

```
if(AxmGantrySetEnable (0, 1, TRUE, 0, 0.0) == AXT_RT_SUCCESS )
    printf("Axis 0, axis 1 are set to Gantry Control.\n");
else
    printf("Axis 0, axis 1 are failed in setting Gantry Control.\n");

long lHmDir = 1;
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 "Axis 0, axis 1 are set to Gantry Control.", vbOKCancel
Else
    MsgBox "Axis 0, axis 1 are failed in setting Gantry Control",
    vbOKCancel
End If

Dim dwHOffset As Double
Dim dwHClrTim As Double
Dim lHmDir, lHmsig, dwZphas As Long

dwHClrTim = 2000#
dwHOffset = 0#

lHmDir = -1
lHmsig = 4
dwZphas = 0
'home search in (-) direction, using Home Sensor, Home Sensor Type A
    contact, not using z phase detecting, not moving offset

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 := -1;
  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('Axis 0, axis 1 are set to Gantry Control.',
        'Ajinextek', MB_OK)
else
    Application.MessageBox('Axis 0, axis 1 are failed in setting Gantry
        Control.', '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

By using this API, return parameter which was specified when setting master axis to gantry control.

Format

C

```
DWORD AxmGantryGetEnable (long lMasterAxisNo, DWORD *upSIHomeUse, double *dpSIOffset, double
*dpSIORange, DWORD *upGatryOn);
```

Visual Basic

```
Function AxmGantryGetEnable (ByVal lMasterAxisNo As Long, ByRef upSIHomeUse As Long, ByRef dpSIOffset As
Double, ByRef dpSIORange As Double, ByRef upGatryOn As Long) As Long
```

Delphi

```
function AxmGantryGetEnable (lMasterAxisNo : LongInt; upSIHomeUse : PDWord; dpSIOffset : Pdouble;
dpSIORange : Pdouble; upGatryOn : PDWord) : Dword; stdcall;
```

Input/Output

Name	in/out	Format	Init Value	Explanation
MasterAxisNo	in	long		Master channel(axis) number(starting from 0)
SIHomeUse	out	DWORD*		Whether Slave axis will do home search like Master or not.
SIOffset	out	double*		Mechanical error between home sensor of master axis and home sensor of slave axis.
SIOffsetRange	out	double*		Specify maximum error value between home sensor of master axis and home sensor of slave axis in home search.
GatryOn	out	DWORD*		Gantry ON/OFF parameter

uSIHomeUse

#define	Value	Explanation
DISABLE	00h	Select home search only for Master axis
ENABLE	01h	Identify selecting home search for Master axis and Slave axis together.

uGatryOn

#define	Value	Explanation
DISABLE	00h	Gantry OFF

ENABLE	01h	Gantry ON
--------	-----	-----------

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

Description

cf : You can put NULL value for unnecessary value to get only the necessary data.

To check the parameters related to Gantry configured by the AmxGantrySetEnable function.
Refer to the description of AsmGantrySetEnable function for the detailed explanation of parameters.

C Example

```
double    dOffset;
DWORD    upHomeResult0;
double    dpS1Offset,dS1ORange;
DWORD    bS1HomeUse, bGtryOn;
if(AxmGantrySetEnable (0, 1, TRUE, 0, 0.0) == AXT_RT_SUCCESS )
    printf("Axis 0, axis 1 are set to Gantry Control.\n");
else
    printf("Axis 0, axis 1 are failed in setting Gantry Control.\n");

long    lHmDir = 1;
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, &bS1HomeUse, &dpS1Offset, &dS1ORange, &bGtryOn);
```

VB Example

```
Dim    dOffset , dS1ORange, dpS1Offset As Double
Dim upHomeResult0, uID As Long
Dim bS1HomeUse, bGtryOn As Long
Dim strData As String
If (AxmGantrySetEnable (0, 1, 1, 0#, 5#) = AXT_RT_SUCCESS) Then
    MsgBox "Axis 0, axis 1 are set to Gantry Control", vbOKCancel
Else
    MsgBox "Axis 0, axis 1 are failed in setting Gantry Control.", vbOKCancel
End If

Dim dwHOffset As Double
Dim dwHClrTim As Double
Dim lHmDir, lHmsig, dwZphas As Long

dwHClrTim = 2000#
dwHOffset = 0#

lHmDir = -1
lHmsig = 4
dwZphas = 0
'home search in (-) direction, using home sensor, Home Sensor Type A
'contact, not using z phase detecting, not moving offset

AxmHomeSetMethod 0, lHmDir, lHmsig, dwZphas, dwHClrTim, dwHOffset
AxmHomeSetVel 0, 10000, 5000, 500, 50, 40000, 10000
```

```
AxmHomeSetStart 0
AxmGantryGetEnable 0, bSlHomeUse, dpSlOffset, dSlORange, bGtryOn
```

Delphi Example

```
var
  dwZphas, lHmsig : DWORD;
  dwHOffset, dwHClrTim : Double;
  lHmDir : LongInt;
  velocity : Double;
  dOffset: Double;
  upHomeResult0, bMaSlChange, bSlHomeUse , bGtryOn :DWORD;
  dSlORange, dpSlOffset : Double;

begin
  lHmDir := -1;
  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('Axis 0, axis 1 are set to Gantry Control.',
      'Ajinextek', MB_OK)
  else
    Application.MessageBox('Axis 0, axis 1 are failed in setting to Gantry
      Control.', '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, @bGtryOn);

End;
```

See Also

[AxmGantrySetEnable](#), [AxmGantrySetDisable](#), [AxmHomeSetMethod](#), [AxmHomeGetMethod](#),
[AxmHomeSetVel](#), [AxmHomeGetVel](#), [AxmHomeSetStart](#), [AxmHomeSetResult](#),
[AxmHomeGetResult](#), [AxmHomeGetRate](#)

AxmGantrySetDisable

Purpose

Release Gantry move system in which two axes are mechanically linked.

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		Master channel(axis) number(starting from 0)
SlaveAxisNo	in	long		Slave channel(axis) number(starting from 0)

Return AXT_RT_SUCCESS(0000) : Successful execution of API.

[* See error code Table for more information on status error codes](#)

Description

Release gantry move system control linked by specification of '[AxmGantrySetEnable](#)'

C Example

```
if(AxmGantrySetEnable (0, 1, TRUE, 0, 0.0) == AXT_RT_SUCCESS )
    printf("Axis 0, axis 1 are set to Gantry Control.\n");
else
    printf("Axis 0, axis 1 are failed in setting Gantry Control.\n");

// Release Gantry move system..
AxmGantrySetDisable (0, 1);
```

VB Example

```
If (AxmGantrySetEnable (0, 1, 1, 0#, 5#) = AXT_RT_SUCCESS) Then
    MsgBox "Axis 0, axis 1 are set to Gantry Control", vbOKCancel
Else
    MsgBox "00, axis 1 are failed in setting Gantry Control", vbOKCancel
End If

AxmGantrySetDisable 0, 1
```

Delphi Example

```
begin
  if (AxmGantrySetEnable (0,1,1, 0.0, 5.0) = AXT_RT_SUCCESS) then
```

```
Application.MessageBox('Axis 0, axis 1 are set to Gantry Control.',  
    'Ajinextek', MB_OK)  
else  
    Application.MessageBox('Axis 0, axis 1 are failed in setting Gantry  
    Control.', 'Ajinextek', MB_OK);  
  
AxmGantrySetDisable (0, 1);  
  
End;
```

See Also

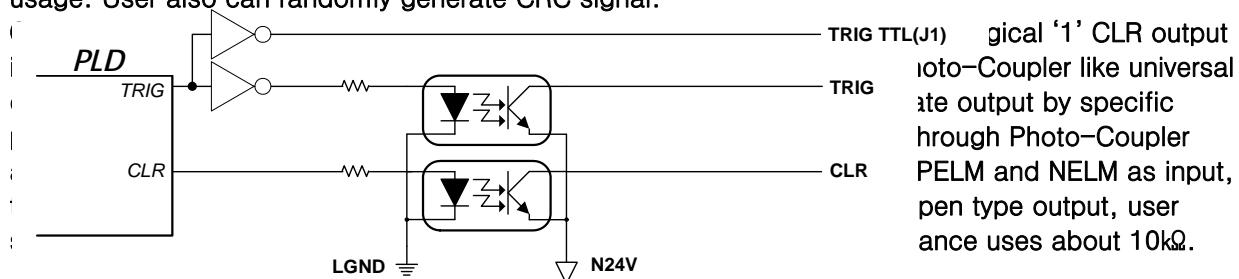
[AxmGantrySetEnable](#), [AxmGantryGetEnable](#), [AxmHomeSetMethod](#), [AxmHomeGetMethod](#),
[AxmHomeSetVel](#), [AxmHomeGetVel](#), [AxmHomeSetStart](#), [AxmHomeSetResult](#),
[AxmHomeGetResult](#), [AxmHomeGetRate](#)

CRC(Remaining pulse clear) API

Function	Description
AxmCrcSetMaskLevel	Set whether to use CRC signal and output level for specified axis.
AxmCrcGetMaskLevel	Return whether to use CRC signal and output level for specified axis.
AxmCrcSetOutput	Forcedly generate CRC signal for specific axis.
AxmCrcGetOutput	Return whether it forcedly generated CRC signal for specific axis.
AxmCrcSetEndLimit	Set whether to use CRC signal and output level for the limit of specified axis.
AxmCrcGetEndLimit	Return whether to use CRC signal and output level for the limit of specified axis.

CRC signal setting

In case of specific Servo pack, when move is completed or identifies limit sensor, there are some situations in which remaining pulse should be cleared by receiving CRC(Current Remaining Clear) signal from outside. For this, it provides signal pin to link directly to Servo Pack. In addition, we provide function to determine the presence of CRC signal usage, output level, and limit signal usage. User also can randomly generate CRC signal.



AxmCrcSetMaskLevel

Purpose

Set whether to use CRC signal for specified axis.

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		channel(axis) number(starting from 0)
Level	In	DWORD		Whether to use CRC signal or not
Method	In	DWORD	0	Remaining pulse clearing output signal pulse width (only for PCI-N804/404, not for SMC-2V03)

Level

#define	Value	Explanation
LOW	00h	B contact(NORMAL CLOSE) Use Active Low Level CRC signal
HIGH	01h	A contact(NORMAL OPEN) Use Active High Level CRC signal
UNUSED	02h	CRC signal not in use(only for SMC-2V03, not for PCI-N804/404)
USED	03h	Maintain current status of CRC signal(only for SMC-2V03, not for PCI-N804/404)

Method

Value	Explanation
00h	Not in use
01h	Not in use.
02h	500 uSec
03h	1 mSec

04h	10 mSec
05h	50 mSec
06h	100 mSec
07h	Logic Level

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

Description

Cf) To make CRC Enable in PCI-N804/404, use [AxmCrcSetEndLimitAPI](#).

Set whether to use CRC signal or signal active level for user's specified axis.

Low(B contact) : For Normal Close -> If detected Open
 High(A contact): For Normal Open -> If detected Close

C Example

```
// set whether to use CRC signal on axis 0
AxmCrcSetMaskLevel (0, LOW, 0);

// identify whether to use CRC signal on axis 0
DWORD uLevel, uMethod;
AxmCrcSetMaskLevel (0, &uLevel, &uMethod);
```

VB Example

```
'set whether to use CRC signal on axis 0
AxmCrcSetMaskLevel 0, LOW, 0

'identify whether to use CRC signal on axis 0
Dim uLevel As Long
Dim uMethod As Long
AxmCrcSetMaskLevel 0, uLevel, uMethod
```

Delphi Example

```
{ identify whether to use CRC signal on axis 0
}
var
uLevel: DWord;
uMethod: DWord;

Begin
{ set whether to use CRC signal on axis 0}
AxmCrcSetMaskLevel (0, LOW, 0);

AxmCrcSetMaskLevel (0, @uLevel, @uMethod);
end;
```

See Also

[AxmCrcGetMaskLevel](#), [AxmCrcSetOutput](#), [AxmCrcGetOutput](#), [AxmCrcSetEndLimit](#), [AxmCrcGetEndLimit](#)

AxmCrcGetMaskLevel

Purpose

Return whether to use CRC signal or not for specific axis.

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		Channel (axis) number(starting from 0)
Level	Out	DWORD*		Whether to use CRC signal or not
Method	Out	DWORD*	0	Remaining pulse clearing output signal pulse width (only for PCI-N804/404, not for SMC-2V03)

Level

#define	Value	Explanation
LOW	00h	B CONTACT (NORMAL CLOSE) Use Active Low Level CRC signal
HIGH	01h	A CONTACT (NORMAL OPEN) Use Active High Level CRC signal
UNUSED	02h	CRC signal not in use (only for SMC-2V03, not for PCI-N804/404)
USED	03h	Maintain current status of CRC signal (only for SMC-2V03, not for PCI-N804/404)

Method

Value	Explanation
00h	Not in use
01h	Not in use
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 for more information on status error codes](#)

Description

Return whether to use CRC signal or signal active level setting for specified axis set by
[‘AxmCrcSetMaskLevel’](#)

Low(B contact) : For Normal Close -> If detected Open
 High(A contact): For Normal Open -> If detected Close

C Example

```
// set whether to use CRC signal on axis 0
AxmCrcSetMaskLevel (0, LOW, 0);

// identify whether to use CRC signal on axis 0
DWORD uLevel, uMethod;
AxmCrcGetMaskLevel (0, &uLevel, &uMethod);
```

VB Example

```
' set whether to use CRC signal on axis 0
AxmCrcSetMaskLevel 0, LOW, 0

' identify whether to use CRC signal on axis 0
Dim uLevel As Long
Dim uMethod As Long
AxmCrcGetMaskLevel 0, uLevel, uMethod
```

Delphi Example

```
{ identify whether to use CRC signal on axis 0 }
var
uLevel: DWord;
uMethod: DWord;

Begin
{ set whether to use CRC signal on axis 0 }
AxmCrcSetMaskLevel (0, LOW, 0);

AxmCrcGetMaskLevel (0, @uLevel, @uMethod);
end;
```

See Also

[AxmCrcSetMaskLevel](#), [AxmCrcSetOutput](#), [AxmCrcGetOutput](#), [AxmCrcSetEndLimit](#), [AxmCrcGetEndLimit](#)

AxmCrcSetOutput

Purpose

Generate forced output of CRC signal for specific axis.

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		Channel (axis) number(starting from 0)
OnOff	In	DWORD		Whether to generate CRC signal by Program

OnOff

#define	Value	Explanation
FALSE	00h	CRC signal Off
TRUE	01h	CRC signal On

Return AXT_RT_SUCCESS(0000) : Successful execution of API.

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

Description

Cf) To make CRC Enable in PCI-N804/404, use [AxmCrcSetEndLimitAPI](#).

Generate CRC signal on the specific axis when the user wants.

Low(B contact) : For Normal Close -> If detected Open

High(A contact): For Normal Open -> If detected Close

C Example

```
// Generate CRC signal on axis 0 by program
AxmCrcSetOutput (0, ENABLE);
```

VB Example

```
' Generate CRC signal on axis 0 by program
AxmCrcSetOutput 0, ENABLE
```

Delphi Example

```
begin
{ Generate CRC signal on axis 0 by program }
AxmCrcSetOutput (0, ENABLE);
End;
```

See Also

[AxmCrcSetMaskLevel](#), [AxmCrcGetMaskLevel](#), [AxmCrcSetEndLimit](#), [AxmCrcGetEndLimit](#)

AxmCrcGetOutput

Purpose

Return whether to generate forced output of CRC signal for specific axis.

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		Channel (axis) number(starting from 0)
OnOff	out	DWORD*		Whether to generate CRC signal by Program

OnOff

#define	Value	Explanation
FALSE	00h	CRC Signal Off
TRUE	01h	CRC Signal On

Return AXT_RT_SUCCESS(0000) : Successful execution of API.

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

Description

Return whether to use CRC signal or signal active level for user's specified axis set by '[AxmCrcSetOutput](#)'.

Low(B contact) : For Normal Close -> Open if detected

High(A contact): For Normal Open -> Close if detected

C Example

```
// Generate CRC signal on axis 0 by program
AxmCrcSetOutput (0, ENABLE);

// identify whether CRC signal is generated on axis 0 by program
DWORD uUse;
AxmCrcGetOutput (0, &uUse);
```

VB Example

```
' Generate CRC signal on axis 0 by program
AxmCrcSetOutput 0, ENABLE
```

```
' identify whether CRC signal is generated on axis 0 by program
Dim uUse As Long
AxmCrcGetOutput 0, uUse
```

Delphi Example

```
var
uUse: DWord;

begin
{ Generate CRC signal on axis 0 by program }
AxmCrcSetOutput (0, ENABLE);
AxmCrcGetOutput (0, @uUse);
end;
```

See Also

[AxmCrcSetMaskLevel](#), [AxmCrcGetMaskLevel](#), [AxmCrcSetOutput](#), [AxmCrcSetEndLimit](#),
[AxmCrcGetEndLimit](#)

AxmCrcSetEndLimit

Purpose

Set whether to output CRC signal when detected by limit sensor of specific axis.

Format

C

```
DWORD AxmCrcSetEndLimit (long lAxisNo, DWORD uPositiveLevel, DWORD uNegativeLevel);
```

Visual Basic

```
Function AxmCrcSetEndLimit (ByVal lAxisNo As Long, ByVal uPositiveLevel As Long, ByVal uNegativeLevel As Long)
As Long
```

Delphi

```
function AxmCrcSetEndLimit (lAxisNo : LongInt; uPositiveLevel : DWord; uNegativeLevel : DWord ) : DWord; stdcall;
```

Input / Output

Name	in/out	Format	Init Value	Explanation
AxisNo	in	long		channel(axis) number(starting from 0)
PositiveLevel	In	DWORD		Whether to use CRC signal for limit of + direction (only for SMC-2V03, not for PCI-N804/404)
NegativeLevel	In	DWORD		Whether to use CRC signal for limit of - direction(only for SMC-2V03, not for PCI-N804/404)

PositiveLevel

#define	Value	Explanation
LOW	00h	B contact(NORMAL CLOSE) Use Active Low Level CRC signal
HIGH	01h	A contact(NORMAL OPEN) Use Active High Level CRC signal
UNUSED	02h	CRC signal for limit input of + direction not in use(only for SMC-2V03, not for PCI-N804/404)+
USED	03h	Maintain current status of CRC signal(only for SMC-2V03, not for PCI-N804/404)

NegativeLevel

#define	Value	Explanation
LOW	00h	B contact(NORMAL CLOSE)

		Use Active Low Level CRC signal
HIGH	01h	A contact(NORMAL OPEN) Use Active High Level CRC signal
UNUSED	02h	CRC signal for limit input of – direction not in use(only for SMC-2V03, not for PCI-N804/404)
USED	03h	Maintain current status of CRC signal(only for SMC-2V03, not for PCI-N804/404)

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

Description

Caution : SMC-2V03 generates remaining pulse clear signal on stop only by limit signal, but PCI-N804/404 enables server remaining pulse clear signal on stop by limit, alarm, emergency stop, and sync stop signals. Signal will be outputted on abnormal exit.

When using CRC, if you use Servo motor, you have to be careful because if you push Emergency Stop(ESTOP) actual position will differ than command position by following error of Servo motor. In this case, you have to forcedly clear position or find home and clear command and actual position. If motor unexpectedly meets End Limit sensor during moving, it is possible to do wrong action because of remaining pulse in Servo derive. Therefore, it is to prevent from wrong action by setting it to automatically generate CRC output when it meets end limit sensor.

Low(B contact) : For normal Close → Open if detected
 High(A contact): For normal Open → Close if detected

C Example

```
// set whether to use CRC signal for limit on axis 0
AxmCrcSetEndLimit(0, LOW, LOW);

// set whether to use CRC signal for limit on axis 0
DWORD uPLevel, uNLevel;
AxmCrcGetEndLimit (0, &uPLevel, &uNLevel);
```

VB Example

```
' set whether to use CRC signal for limit on axis 0
AxmCrcSetEndLimit 0, LOW, LOW

' set whether to use CRC signal for limit on axis 0
Dim uPLevel , uNLevel As Long
AxmCrcGetEndLimit 0, uPLevel, uNLevel
```

Delphi Example

```
var
uPLevel, uNLevel: DWord;

begin
{ set whether to use CRC signal for limit on axis 0 }
AxmCrcSetEndLimit(0, LOW, LOW);
AxmCrcGetEndLimit (0, @uPLevel, @uNLevel);
end;
```



See Also

[AxmCrcSetMaskLevel](#), [AxmCrcGetMaskLevel](#), [AxmCrcSetOutput](#), [AxmCrcGetOutput](#), [AxmCrcGetEndLimit](#)

AxmCrcGetEndLimit

Purpose

Return whether to output CRC signal when detected by limit sensor on specific axis.

Format

C

```
DWORD AxmCrcGetEndLimit (long IAxisNo, DWORD *upPositiveLevel, DWORD *upNegativeLevel);
```

Visual Basic

```
Function AxmCrcGetEndLimit (ByVal IAxisNo As Long, ByRef upPositiveLevel As Long, ByRef upNegativeLevel As Long) As Long
```

Delphi

```
function AxmCrcGetEndLimit (IAxisNo : LongInt; upPositiveLevel : PDWord; upNegativeLevel : PDWord) : DWord;
stdcall;
```

Input / Output

Name	in/out	Format	Init Value	Explanation
AxisNo	in	long		Channel (axis) number(starting from 0)
PositiveLevel	out	DWORD*		Whether to use CRC signal for limit of + direction(only for SMC-2V03, not for PCI-N804/404)
NegativeLevel	out	DWORD*		Whether to use CRC signal for limit of - direction(only for SMC-2V03, not for PCI-N804/404)

PositiveLevel

#define	Value	Explanation
LOW	00h	B contact(NORMAL CLOSE) Use Active Low Level CRC signal
HIGH	01h	A contact(NORMAL OPEN) Use Active High Level CRC signal
UNUSED	02h	CRC signal for limit input of + direction not in use(only for SMC-2V03, not for PCI-N804/404)+
USED	03h	Maintain current status of CRC signal

NegativeLevel

#define	Value	Explanation
LOW	00h	B contact(NORMAL CLOSE) Use Active Low Level CRC signal
HIGH	01h	A contact(NORMAL OPEN)

		Use Active High Level CRC signal
UNUSED	02h	CRC signal for limit input of – direction not in use(only for SMC-2V03, not for PCI-N804/404)
USED	03h	Maintain current status of CRC signal

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

Description

Return whether to use CRC signal for limit of user's specified axis set by '[AxmCrcSetEndLimit](#)'.

Low(B contact) : For normal Close → Open if detected
 High(A contact): For normal Open → Close if detected

C Example

```
// set whether to use CRC signal for limit on axis 0
AxmCrcSetEndLimit(0, LOW, LOW);

// set whether to use CRC signal for limit on axis 0
DWORD uPLevel, uNLevel;
AxmCrcGetEndLimit (0, &uPLevel, &uNLevel);
```

VB Example

```
' set whether to use CRC signal for limit on axis 0
AxmCrcSetEndLimit 0, LOW, LOW

' set whether to use CRC signal for limit on axis 0
Dim uPLevel , uNLevel As Long
AxmCrcGetEndLimit 0, uPLevel, uNLevel
```

Delphi Example

```
var
  uPLevel, uNLevel: DWord;

begin
  { set whether to use CRC signal for limit on axis 0 }
  AxmCrcSetEndLimit(0, LOW, LOW);
  AxmCrcGetEndLimit (0, @uPLevel, @uNLevel);
end;
```

See Also

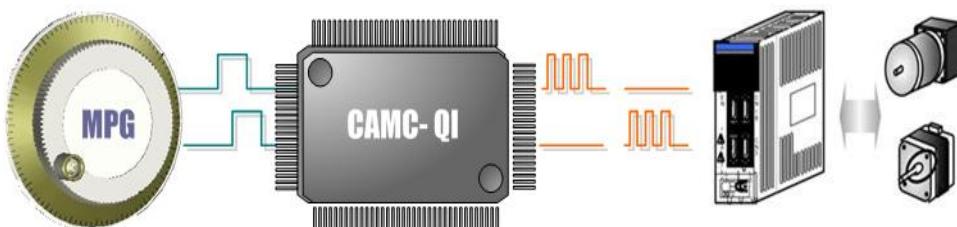
[AxmCrcSetMaskLevel](#), [AxmCrcGetMaskLevel](#), [AxmCrcSetOutput](#), [AxmCrcGetOutput](#), [AxmCrcSetEndLimit](#)

MPG move API

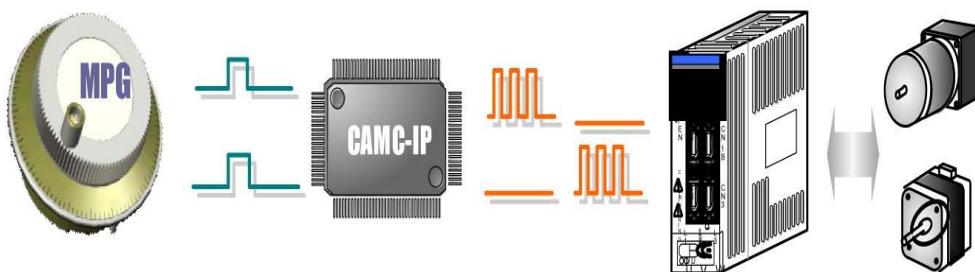
Function	Description.
AxmMPGSetEnable	Set MPG input method, drive move mode, moving distance for specific axis
AxmMPGGetEnable	Return MPG input method, drive move mode, moving distance for specific axis
AxmMPGSetRatio	Set pulse ratio of how much to move for a pulse in MPG drive move mode for specific axis.
AxmMPGGetRatio	Return pulse ratio of how much to move for a pulse in MPG drive move mode for specific axis.
AxmMPGReset	Release MPG drive setting for specific axis.

Directional signal is determined by EXPP, EXMP signal, and the number of pulse by input spec setting register. The speed of output pulse is outputted by Object speed.

PCI-N804/404



SMC-2V03

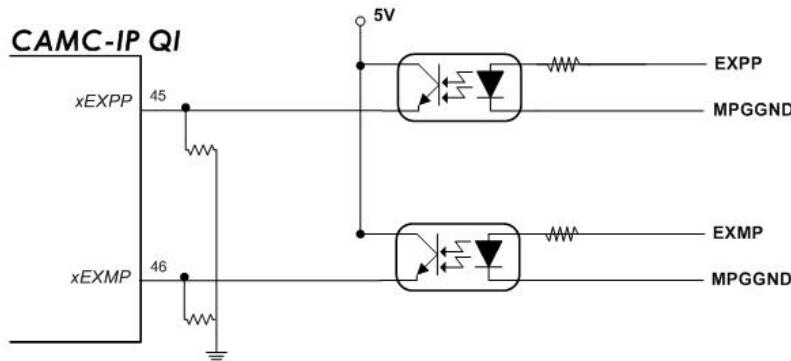


Input pulse type(PCI-N804/404)

input method [2:0]	input method [3] = '0'	input method [3] = '1'
"000" (One pulse input) (Two phase 1 time)	<p>COUNT UP COUNT DOWN</p> <p>[CNTH] N N+1 N N-1</p>	<p>COUNT DOWN COUNT UP</p> <p>[CNTH] N-1 N-2 N-1 N</p>
"100" (Two pulse) -	<p>COUNT UP COUNT DOWN</p> <p>[CNTH] N N+1 N N-1</p>	<p>COUNT DOWN COUNT UP</p> <p>[CNTH] N-1 N-2 N-1 N</p>
"010" (Two phase 2 time) -	<p>COUNT UP COUNT DOWN</p> <p>[CNTH] N N+2 N N-2</p>	<p>COUNT DOWN COUNT UP</p> <p>[CNTH] N-2 N-1 N-1 N</p>
"011" (Two phase 4 times) -	<p>COUNT UP COUNT DOWN</p> <p>[CNTH] N N+4 N N-4</p>	<p>COUNT DOWN COUNT UP</p> <p>[CNTH] N-4 N-3 N-2 N</p>

● interface with the outside MPG

EXPP, EXMP signals are used as input signal of outside rotary encoder of SMC-2V03, QI. They are connected to SMC-2V03, QI chip through maximum 1.2MHz high speed photo coupler.



PCI-N804/404

MPG input signal is not inputted to each axis independently, but shared by both axes. MPG signal for axis 0 is shared by axis 2, MPG signal for axis 1 is shared by axis 3, MPG signal for axis 4 is shared by axis 6, and MPG signal for axis 5 is shared by axis 7. In other words, MPG signal input pin of external connector J2/J4 linked with axis 2/3/6/7 is not used.

[UCFG3][2~0] ^{a)}	[UCFG3][3] = '0' ^{b)}	[UCFG3][3] = '1' ^{c)}
"000" ^{d)} (One pulse mode) ^{e)}	<p>CW CCW</p> <p>PULS</p> <p>DIR</p> <p>Level 'Low' Level 'High'</p>	<p>CW CCW</p> <p>PULS</p> <p>DIR</p> <p>Level 'High' Level 'Low'</p>
"001" ^{d)} (One pulse mode) ^{e)}	<p>CW CCW</p> <p>PULS</p> <p>DIR</p> <p>Level 'Low' Level 'High'</p>	<p>CW CCW</p> <p>PULS</p> <p>DIR</p> <p>Level 'High' Level 'Low'</p>
"010" ^{d)} (Two pulse mode) ^{e)}	<p>CW CCW</p> <p>PULS</p> <p>DIR</p> <p>Level 'Low' Active High pulse</p>	<p>CW CCW</p> <p>PULS</p> <p>DIR</p> <p>Level 'High' Active Low pulse</p>
"011" ^{d)} (Two pulse mode) ^{e)}	<p>CW CCW</p> <p>PULS</p> <p>DIR</p> <p>Active High pulse Level 'Low'</p>	<p>CW CCW</p> <p>PULS</p> <p>DIR</p> <p>Active Low pulse Level 'High'</p>
"100" ^{d)} (Two phase mode) ^{e)}	<p>CW CCW</p> <p>PULS</p> <p>DIR</p> <p>PULS phase lead than DIR</p>	<p>CW CCW</p> <p>PULS</p> <p>DIR</p> <p>DIR phase lead than PULSE PULS phase lead than DIR</p>

AxmMPGSetEnable

Purpose

Set MPG input method, drive move mode, and moving distance for specific axis.

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		Channel (axis) number(starting from 0)
InputMethod	In	long		Input signal method (PCI-N804/404 is MPG_DIFF_TWO_PHASE_1X not Support)
DriveMode	In	long		MPG motion mode
MPGPos	In	double		Moving distance for MPG_PRESET_MODE mode
dVel	In	double		MPG velocity
dAccel	In	double		MPG accel(IP ONLY)

InputMethod(SMC-2V03)

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

DriveMode

(SMC-2V03)

Value	Explanation

00h	MPG slave mode
01h	MPG PRESET mode
02h	MPG continuous mode

InputMethod(PCI-N804/404)

#define	Value	Explanation
MPG_DIFF_ONE_PHASE	00h	MPG input type One Phase
MPG_DIFF_TWO_PHASE_1X	01h	MPG input type One Phase (PCI-N804/404 is not support)
MPG_DIFF_TWO_PHASE_2X	02h	MPG input type TwoPhase2
MPG_DIFF_TWO_PHASE_4X	03h	MPG input type TwoPhase4

DriveMode

(PCI-N804/404)

Value	Explanation
00h	MPG continuous mode

Return AXT_RT_SUCCESS(0000) : Successful execution of API.

[* See error code Table for more information on status error codes](#)**Description**

Set distance, move mode, and move direction which will be used in MPG drive pulse move mode for specific axis.

Here, what dMPGdenominator = 4095, MPGnumerator=0 means is that for 200 pulses per one MPG cycle, it outputs 1pulse which is 1:1. In the case of dMPGdenominator = 4095, MPGnumerator=1 , it outputs 2 pulses which is 1:2.

Formula for internal chip output is MPG PULSE = ((Numerator) * (Denominator) / 4096).

Caution: The results of API execution after [AxmMPGSetEnable](#) execution are different according to product.

PCI-N804/404: motion move status in the normal status before [AxmMPGReset](#) execution

SMC-2V03: motion move status only when pulse is outputted by input of MPG input signals

C Example

```
// Set Mpg signal on axis 0.
AxmMPGSetEnable (0, MPG_DIFF_ONE_PHASE, 1, 10, 50, 200);
AxmMPGSetRatio(0, 1, 4096 );

// return Mpg signal setting on axis 0.
long lpInputMethod, lpDriveMode;
double dpVel, dpAccel, dpDecel, dpMPGPos;
double dMPGdenominator, dMPGnumerator, dVel;
AxmMPGGetEnable (0, &lpInputMethod, & lpDriveMode, &dpMPGPos, &dVel, &
dAccel);
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

' Set Mpg signal on axis 0.
AxmMPGSetEnable 0, MPG_DIFF_ONE_PHASE, 1, 10, 50
AxmMPGSetRatio 0, 1, 4096

' return Mpg signal setting on axis 0.
AxmMPGGetEnable 0, lInputMethod, lDriveMode, dMPGPosition, dVel, dAccel
AxmMPGGetRatio 0, dMPGnumerator, dMPGdenominator

```

Delphi Example

```

{ return Mpg signal setting on axis 0. }
var
dVelocity : Double;
dAccel : Double;
dDecel : Double;
dMPGdenominator : Double;
dMPGnumerator : Double;
lpInputMethod,lpDriveMode : longInt;
dMPGPosition, dVel : Double;

begin

// Set Mpg signal on axis 0.
AxmMPGSetEnable (0, MPG_DIFF_ONE_PHASE, 1, 10, 50);
AxmMPGSetRatio(0, 1, 4096 );

// return Mpg signal setting on axis 0.
AxmMPGGetEnable (0, @lpInputMethod, @lpDriveMode, @dMPGPosition, @dVel, @
      dAccel);
AxmMPGGetRatio(0, @dMPGnumerator, @dMPGdenominator);

end;

```

See Also

[AxmMPGGetEnable](#), [AxmMPGSetRatio](#), [AxmMPGGetRatio](#), [AxmMPGReset](#)

AxmMPGGetEnable

Purpose

Return MPG input method, drive move mode, and moving distance for specific axis..

Format

C

```
DWORD AxmMPGGetEnable (long lAxisNo, long *lpInputMethod, long *lpDriveMode, double *dpMPGPos , double
*dpVel, double *dAccel);
```

Visual Basic

```
Function AxmMPGGetEnable Lib "AxI.dll" (ByVal lAxisNo As Long, ByRef lpInputMethod As Long, ByRef lpDriveMode
As Long, ByRef dpMPGPos As Double, ByRef dpVel As Double, ByRef dAccel As Double) As Long
```

Delphi

```
function AxmMPGGetEnable (lAxisNo : LongInt; lpInputMethod : PLongInt; lpDriveMode : PLongInt; dpMPGPos : PDouble;
dpVel: PDouble; dAccel: PDouble) : DWord; stdcall;
```

Input/ Output

Name	in/out	Format	Init Value	Explanation
AxisNo	in	long		channel(axis) number(starting from 0)
InputMethod	In	long		Input signal method
DriveMode	In	long		Mpg motion mode
MPGPos	In	double		Moving distance for MPG_PRESET_MODE mode
Vel	In	double		MPG VELOCITY
Accel	In	double		MPG Acceleration(ip only)

Return AXT_RT_SUCCESS(0000) : Successful execution of API.

[* See error code Table for more information on status error codes](#)

Description

Return MPG input method, drive move mode, and moving distance for specific axis set by '[AxmMPGSetEnable](#)'.

C Example

```
// Set Mpg signal on axis 0.
AxmMPGSetEnable (0, MPG_DIFF_ONE_PHASE, 1, 10, 50,200);
AxmMPGSetRatio(0, 1, 4096 );

// return Mpg signal setting on axis 0.
long lpInputMethod, lpDriveMode;
double dpVel, dpAccel, dpDecel, dpMPGPos;
double dMPGdenominator, dMPGnumerator, dVel, dAccel;

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 dMPGnumer, dMPGPosition, dVel As Double
Dim lInputMethod, lDriveMode, lDirMode, lUserDir As Long

' Set Mpg signal on axis 0.
AxmMPGSetEnable 0, MPG_DIFF_ONE_PHASE, 1, 10, 50, 200, 400
AxmMPGSetRatio 0, 1, 4096

' return Mpg signal setting on axis 0.
AxmMPGGetEnable 0, lInputMethod, lDriveMode, dMPGPosition, dVel, dAccel
AxmMPGGetRatio 0, dMPGnumer, dMPGdenominator

```

Delphi Example

```

{ return Mpg signal setting on axis 0. }
var
dVelocity : Double;
dAccel : Double;
dDecel : Double;
dMPGdenominator : Double;
dMPGnumer : Double;
lpInputMethod,lpDriveMode : longInt;
dMPGPosition, dVel : Double;

begin
  // Set Mpg signal on axis 0.
  AxmMPGSetEnable (0, MPG_DIFF_ONE_PHASE, 1, 10, 50, 200, 400);
  AxmMPGSetRatio(0, 1, 4096 );

  // return Mpg signal setting on axis 0.
  AxmMPGGetEnable (0, @lpInputMethod, @lpDriveMode, @dMPGPosition, @dVel,
    @dAccel);
  AxmMPGGetRatio(0, @dMPGnumer, @dMPGdenominator);

end;

```

See Also

[AxmMPGSetEnable](#), [AxmMPGSetRatio](#), [AxmMPGGetRatio](#), [AxmMPGReset](#),

AxmMPGSetRatio

Purpose

Set pulse ratio of how much to move for a pulse in MPG drive move mode for specific axis.

Format

C

```
DWORD AxmMPGSetRatio(long lAxisNo, double dMPGnumerator, double dMPGdenominator);
```

Visual Basic

```
Function AxmMPGSetRatio Lib "AxI.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		Channel (axis) number(starting from 0)
MPGnumerator	In	double		(Manual pulse occurrence device input) numerator value in move(input value is calculated) maximum(from 0 to 64), but only for PCI-N804/404., MPG PULSE = (dwNumerator)* (dwDenominator)/ 4096 Parameter setting will not be applied for SMC-2V03
MPGdenominator	In	double		(Manual pulse occurrence device input) denominator value in move (input value is calculated) maximum(from 0 to 4096), but only for PCI-N804/404, Parameter setting will not be applied for SMC-2V03

Return AXT_RT_SUCCESS(0000) : Successful execution of API.

[* See error code Table for more information on status error codes](#)

Description

Set distance, move mode, and move direction which will be used in MPG drive pulse move mode for specific axis.

Here, what dMPGdenominator = 4095, MPGnumerator=0 means is that for 200 pulses per one MPG cycle, it outputs 1pulse which is 1:1. In the case of dMPGdenominator = 4095, MPGnumerator=1 , it outputs 2 pulses which is 1:2.

Formula for internal chip output is MPG PULSE = ((Numerator) * (Denominator) / 4096).

C Example

```
// Set Mpg signal on axis 0.
AxmMPGSetEnable (0, MPG_DIFF_ONE_PHASE, 1, 10, 50);
AxmMPGSetRatio(0, 1, 4096 );

// return Mpg signal setting on axis 0.
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

' Set Mpg signal on axis 0.
AxmMPGSetEnable 0, MPG_DIFF_ONE_PHASE, 1, 10, 50
AxmMPGSetRatio 0, 1, 4096

' return Mpg signal setting on axis 0.
AxmMPGGetEnable 0, lInputMethod, lDriveMode, dMPGPosition, dVel
AxmMPGGetRatio 0, dMPGnumerator, dMPGdenominator
```

Delphi Example

```
{ return Mpg signal setting on axis 0. }
var
dVelocity : Double;
dAccel : Double;
dDecel : Double;
dMPGdenominator : Double;
dMPGnumerator : Double;
lInputMethod,lpDriveMode : longInt;
dMPGPosition, dVel : Double;

begin

// Set Mpg signal on axis 0.
AxmMPGSetEnable (0, MPG_DIFF_ONE_PHASE, 1, 10, 50);
AxmMPGSetRatio(0, 1, 4096 );

// return Mpg signal setting on axis 0.
AxmMPGGetEnable (0, @lpInputMethod, @lpDriveMode, @dMPGPosition, @dVel);
AxmMPGGetRatio(0, @dMPGnumerator, @dMPGdenominator);

end;
```

See Also

[AxmMPGSetEnable](#), [AxmMPGGetEnable](#), [AxmMPGGetRatio](#), [AxmMPGReset](#)

AxmMPGGetRatio

Purpose

Return pulse ratio of how much to move for a pulse in MPG drive move mode for specific axis..

Format

C

```
DWORD AxmMPGGetRatio(long IAxisNo, DWORD *upMPGnumerator, DWORD *upMPGdenominator);
```

Visual Basic

```
Function AxmMPGGetRatio Lib "Axl.dll" (ByVal IAxisNo As Long, ByRef dpMPGnumerator As Double, ByRef dpMPGdenominator As Double) As Long
```

Delphi

```
function AxmMPGGetRatio (IAxisNo : LongInt; dpMPGnumerator : PDouble; dpMPGdenominator : PDouble) : DWord;
  stdcall;
```

Input/ Output

Name	in/out	Format	Init Value	Explanation
AxisNo	in	long		Channel (axis) number(starting from 0)
MPGdenominator	In	double		Manual pulse occurrence device input) denominator value in move (input value is calculated) maximum(from 0 to 4096), but only for PCI-N804/404, Parameter setting will not be applied for SMC-2V03
MPGnumerator	In	double		(Manual pulse occurrence device input) numerator value in move(input value is calculated) maximum(from 0 to 64), but only for PCI-N804/404., MPG PULSE = (dwNumerator)* (dwDenominator)/ 4096 Parameter setting will not be applied for SMC-2V03

Return AXT_RT_SUCCESS(0000) : Successful execution of API.

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

Description

Return pulse ratio of how much to move for a pulse in MPG drive move mode for specific axis set by
[‘AxmMPGSetRatio.’](#)

C Example

```
// Set Mpg signal on axis 0.
AxmMPGSetEnable (0, MPG_DIFF_ONE_PHASE, 1, 10, 50);
AxmMPGSetRatio(0, 1, 4096 );

// return Mpg signal setting on axis 0.
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

' Set Mpg signal on axis 0.
AxmMPGSetEnable 0, MPG_DIFF_ONE_PHASE, 1, 10, 50
AxmMPGSetRatio 0, 1, 4096

' return Mpg signal setting on axis 0.
AxmMPGGetEnable 0, lInputMethod, lDriveMode, dMPGPosition, dVel
AxmMPGGetRatio 0, dMPGnumerator, dMPGdenominator
```

Delphi Example

```
{ return Mpg signal setting on axis 0. }
var
dVelocity : Double;
dAccel : Double;
dDecel : Double;
dMPGdenominator : Double;
dMPGnumerator : Double;
lpInputMethod,lpDriveMode : longInt;
dMPGPosition, dVel : Double;

begin

// Set Mpg signal on axis 0.
AxmMPGSetEnable (0, MPG_DIFF_ONE_PHASE, 1, 10, 50);
AxmMPGSetRatio(0, 1, 4096 );

// return Mpg signal setting on axis 0.
AxmMPGGetEnable (0, @lpInputMethod, @lpDriveMode, @dMPGPosition, @dVel);
AxmMPGGetRatio(0, @dMPGnumerator, @dMPGdenominator);

end;
```

See Also

[AxmMPGSetEnable](#), [AxmMPGGetEnable](#), [AxmMPGSetRatio](#), [AxmMPGReset](#)

AxmMPGReset

Purpose

Release MPG drive setting for specific axis.

Format

C

```
DWORD AxmMPGReset (long IAxisNo);
```

Visual Basic

```
Function AxmMPGReset (ByVal IAxisNo As Long) As Long
```

Delphi

```
function AxmMPGReset (IAxisNo : LongInt) : DWord; stdcall;
```

Input/ Output

Name	in/out	Format	Init Value	Explanation
AxisNo	in	long		Channel (axis) number(starting from 0)

Return AXT_RT_SUCCESS(0000) : Successful execution of API.

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

Description

Release MPG drive settings such as distance, move mode, and move direction for specific axis set by '[AxmMPGSetEnable](#)'.

C Example

```
// release MPG drive setting on axis 0.
AxmMPGReset(0);
```

VB Example

```
' release MPG drive setting on axis 0.
AxmMPGReset 0
```

Delphi Example

```
begin
{release MPG drive setting on axis 0.}
AxmMPGReset(0);
End;
```

See Also

[AxmMPGSetEnable](#), [AxmMPGGetEnable](#), [AxmMPGSetRatio](#), [AxmMPGGetRatio](#)

Helical interpolation setting and move API

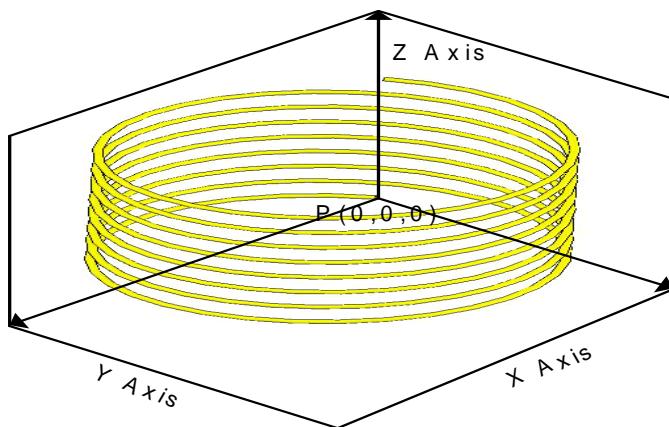
This API support only PCI-N804/404 specialized API.

Helical interpolation control is the function which has combined 4 axes including 2 axes of circular interpolation and 2 axes of line interpolation. Thus, it can execute synchronized line and circular interpolation of 2 axes. It provides function that can execute one for a time, and that can be used with continuous interpolation function. To process helical interpolation, speed and distance of circular interpolation move should be synchronized with axis U(axis 4). The final axis should be axis U, and because axis U is moving with circular interpolation distance fallows axis Z and speed cannot be different from circular interpolation. In the case of motion board of 4 axes one group consists of axis X(0),Y(1),Z(2),U(3), and In the case of motion board of 8 axes one group consists of axis X(0),Y(1),Z(2),U(3)와 X2(4),Y2(5),Z2(6),U(7).

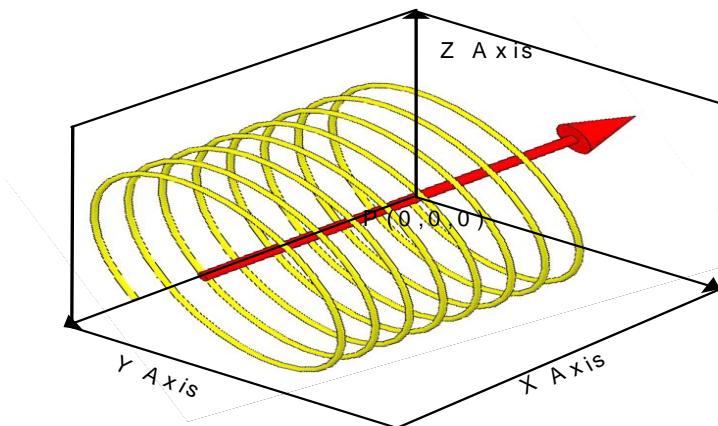
Function	Description
<u>AxmHelixCenterMove</u>	<p>API to move helical interpolation with specification of start point, end point, and center point for specific axis.</p> <p>API to move helical interpolation with specification of start point, end point, and center point for specific axis in using with <u>AxmContiBeginNode</u>, <u>AxmContiEndNode</u>. It saves to internal Queue for circular continuous interpolation move. It starts with the use of <u>AxmContiStart</u> API.(It is used with continuous interpolation API.)</p>
<u>AxmHelixPointMove</u>	<p>API to move helical interpolation with specification of start point, end point, and radius for specific axis.</p> <p>API to move helical interpolation with specification of start point and center point for specific axis in using with <u>AxmContiBeginNode</u>, <u>AxmContiEndNode</u>. It saves to internal Queue for circular continuous interpolation move. It starts with the use of <u>AxmContiStart</u> API.(It is used with continuous interpolation API.)</p>
<u>AxmHelixRadiusMove</u>	<p>API to move helical interpolation with specification of start point, end point, and radius for specific axis.</p> <p>API to move helical interpolation with specification of start point, end point, and radius for specific axis in using with <u>AxmContiBeginNode</u>, <u>AxmContiEndNode</u>. It saves to internal Queue for circular continuous interpolation move. It starts with the use of <u>AxmContiStart</u> API.(It is used with continuous interpolation API.)</p>

<u>AxmHelixAngleMove</u>	<p>API to move helical interpolation with specification of start point, rotational angle, and radius for specific axis.</p> <p>API to move helical interpolation with specification of start point, rotational angle, and radius for specific axis in using with <u>AxmContiBeginNode</u>, <u>AxmContiEndNode</u>. It saves to internal Queue for circular continuous interpolation move. It starts with the use of <u>AxmContiStart</u> API.(It is used with continuous interpolation API.)</p>
--	---

Ex 1> Use helical interpolation with continuous interpolation



Ex 2> Use helical interpolation with continuous interpolation



AxmHelixCenterMove

Purpose

API to move helical interpolation with specification of start point, end point, and center point for specific axis. It exits the function at the point of starting pulse output..

Format

C

```
DWORD AxmHelixCenterMove (long lCoord, double dCenterXPos, double dCenterYPos, double dEndXPos, double
dEndYPos, double dZPos, double dVel, double dAccel, double dDecel, DWORD uCWDdir);
```

Visual Basic

```
Function AxmHelixCenterMove (ByVal lCoord As Long, ByVal dCenterXPos As Double, ByVal dCenterYPos As Double,
ByVal dEndXPos As Double, ByVal dEndYPos As Double, ByVal dZPos As Double, ByVal dVel As Double, ByVal
dAccel As Double, ByVal dDecel As Double, ByVal uCWDdir As Long) As Long
```

Delphi

```
function AxmHelixCenterMove (lCoord : LongInt; dCenterXPos : Double; dCenterYPos : Double; dEndXPos : Double;
dEndYPos : Double; dZPos : Double; dVel : Double; dAccel : Double; dDecel : Double; uCWDdir : DWord) : DWord;
stdcall;
```

Input/ Output

Name	in/out	Format	Init Value	Explanation
Coord	in	long		Coordination System number(starting from 0)
CenterXPos	In	double		Home position value of axis X
CenterYPos	In	double		Home position value of axis Y
EndXPos	In	double		End position value of axis X
EndYPos	In	double		End position value of axis Y
ZPos	In	double		Distance to move to the direction of axis Z at a time.
Vel	In	double		Move speed (Move speed unit is PPS[Pulses/sec] given that Unit/Pulse is 1/1)
Accel	In	double		Move acceleration(accleration unit is PPS[Pulses/sec^2] given that Unit/pulse is 1/1)
Decel	In	double		Move deceleration(deceleration unit is PPS[Pulses/sec^2] given that Unit/pulse is 1/1)
CWDdir	In	DWORD		Circular direction

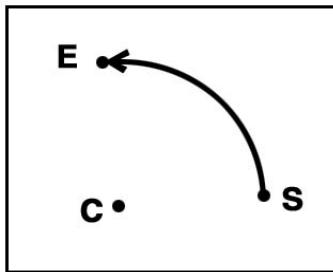
CWDir

#define	Value	Explanation
DIR_CCW	00h	Clockwise
DIR_CW	01h	Counterclockwise

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

Description

API to move helical interpolation with specification of start point, end point, and center point for specific axis in using with [AxmContiBeginNode](#), [AxmContiEndNode](#). It saves to internal Queue for circular continuous interpolation move. It starts with the use of [AxmContiStart](#) API.(It is used with continuous interpolation API.) Current position becomes beginning point of circular move, and it process circular move on the basis of center and end point.



- ▶ By AxmSStop and AxmEstop API, slow down stop and emergency stop are possible. It always stops by transmitting master axis of Coordinate to parameter.
- ▶ [AxmStatusReadInMotion](#) API can verify whether it is on motion or not.
- ▶ When using [AxmSignalSetInpos](#) API, if Inpos is set to Enable, it doesn't return until INP input is set to ON although Command pulse output is completed because it considers move is not completed. Refer to [AxmSignalSetInpos](#) for further information. **
- ▶ Caution: Axes which interpolate Unit/Pulse should be equally set. If Unit/Pulse is not equally set, their unit will different with each other. Always equally set, and do interpolation control.
- ▶ Caution: When using motion board with more than 8 (PCI-N804/404)
 In the board with more than 8 axes, 4 axes from Axis 0–3 becomes axis group 1, 4 axes from Axis 4–7 becomes axis group 2, and 4 axes from Axis 8–11 becomes axis group 3. Axes from the same group can process helical interpolation move, but axes from the different group cannot. The reason is that PCI-N804/404 chip has 4 axes, and it doesn't support this. Keep in mind when specify Coordinate.
- ▶ Caution: When using Helix on continuous interpolation, Spline, linear interpolation and circular interpolation can not be used together.

▶ continuous interpolation API list

Function	Description	Note

AxmContiSetAxisMap	Setting continuous interpolation axis mapping for specified Coordinate system.	
AxmContiGetAxisMap	Setting continuous interpolation axis mapping for specified Coordinate system.	
AxmContiSetAbsRelMode	Setting continuous interpolation axis absolute/relative mode for specified coordinate system.	
AxmContiGetAbsRelMode	Return continuous interpolation axis absolute/relative mode for specified coordinate system.	
AxmContiBeginNode	Start registering tasks which will be processed in continuous interpolation for specified coordinate system. After calling this function, all motion tasks processed before calling AxmContiEndNode function are registered as continuous interpolation motion, but not processing real motion. When AxmContiStart function is called, registered motion is actually started.	
AxmContiEndNode	End registering tasks to process continuous interpolation in specified coordinate system.	
AxmContiReadFree	Identify internal Queue is empty for interpolation move in specified coordinate system.	
AxmContiReadIndex	Identify the number of interpolation move in internalQueue stored for interpolation move in specified coordinate system.	
AxmContiWriteClear	Clear all of the internal Queue stored for continuous interpolation move in a specified coordinate system.	
AxmContiStart	Start continuous interpolation move from internal Queue stored in a specified coordinate system.	
AxmContiIsMotion	Identify continuous interpolation move is running or not in specified coordinate system.	
AxmContiGetNodeNum	Identify the index number of currently running	

	continuous interpolation among continuous interpolation moves in specified coordinate system.	
AxmContiGetTotalNodeNum	Identify total number of index of continuous interpolation move set in specified coordinate system.	

(Normal helical interpolation move example)

C Example

```
double dVelocity = 2000, dAccel = 4000;
long lAxis[4];
long lCoordinate = 0;

for(int i=0; i<4; i++)
{
    lAxis[i] = i;
}

AxmContiWriteClear(lCoordinate);
AxmContiSetAxisMap(lCoordinate, 4, lAxis);
AxmContiSetAbsRelMode(lCoordinate, 1);

AxmHelixCenterMove(lCoordinate, 500, 500, 0, 0, 100, dVelocity, dAccel,
                    dAccel, DIR_CW);
```

VB Example

```
Dim dVelocity As Double
Dim dAccel As Double
Dim lCoordinate As Long

Dim axes(0 To 3) As Long

Dim i As Long
dVelocity = 2000
dAccel = 4000
lCoordinate = 0

For i = 0 To 3
    axes(i) = i
Next i

AxmContiWriteClear lCoordinate
AxmContiSetAxisMap lCoordinate, 4, axes(0)
AxmContiSetAbsRelMode lCoordinate, 1

AxmHelixCenterMove lCoordinate, 500, 500, 0, 0, 100, dVelocity, dAccel,
                    dAccel, 1
```

Delphi Example

```
var
  dVelocity, dAccel : Double;
  i : LongInt;
```

```

lAxis : array [0..3] of LongInt;
lCoordinate : LongInt;

begin
  dVelocity := 2000;
  dAccel := 4000;
  lCoordinate := 0;

  for i := 0 to 3 do
  begin
    lAxis[i] := i;
  end;

  AxmContiWriteClear (lCoordinate);
  AxmContiSetAxisMap (lCoordinate, 4, @lAxis);
  AxmContiSetAbsRelMode(lCoordinate, 1);

  AxmHelixCenterMove(lCoordinate, 500, 500, 0, 0, 100, dVelocity, dAccel,
                     dAccel, DIR_CW);
end;

```

(continuous helical interpolation move example)

C Example

```

double dVelocity = 2000, dAccel = 4000;
long lAxis[4];
long lCoordinate = 0;

for(int i=0; i<4; i++)
{
  lAxis[i] = i;
}

AxmContiWriteClear(lCoordinate);
AxmContiSetAxisMap (lCoordinate, 4, lAxis);
AxmContiSetAbsRelMode(lCoordinate, 1);

AxmContiBeginNode(lCoordinate);
AxmHelixCenterMove(lCoordinate, 500, 500, 0, 0, 100, dVelocity, dAccel,
                   dAccel, DIR_CW);
AxmHelixCenterMove(lCoordinate, 500, 500, 0, 0, 100, dVelocity, dAccel,
                   dAccel, DIR_CW);
AxmHelixCenterMove(lCoordinate, 500, 500, 0, 0, 100, dVelocity, dAccel,
                   dAccel, DIR_CW);
AxmHelixCenterMove(lCoordinate, 500, 500, 0, 0, 100, dVelocity, dAccel,
                   dAccel, DIR_CW);
AxmHelixCenterMove(lCoordinate, 500, 500, 0, 0, 100, dVelocity, dAccel,
                   dAccel, DIR_CW);
AxmContiEndNode (lCoordinate);

AxmContiSetAbsRelMode(lCoordinate, 1);
AxmContiStart (lCoordinate , 0, 0);

```

VB Example

```

Dim dVelocity As Double
Dim dAccel As Double
Dim lCoordinate As Long

Dim axes(0 To 3) As Long

Dim i As Long
dVelocity = 2000

```

```

dAccel = 4000
lCoordinate = 0

For i = 0 To 3
    axes(i) = i

Next i

AxmContiWriteClear lCoordinate
AxmContiSetAxisMap lCoordinate, 4, axes(0)
AxmContiSetAbsRelMode lCoordinate, 1

AxmContiBeginNode lCoordinate
AxmHelixCenterMove lCoordinate, 500, 500, 0, 0, 100, dVelocity, dAccel,
    dAccel, 1
AxmHelixCenterMove lCoordinate, 500, 500, 0, 0, 100, dVelocity, dAccel,
    dAccel, 1
AxmHelixCenterMove lCoordinate, 500, 500, 0, 0, 100, dVelocity, dAccel,
    dAccel, 1
AxmHelixCenterMove lCoordinate, 500, 500, 0, 0, 100, dVelocity, dAccel,
    dAccel, 1
AxmHelixCenterMove lCoordinate, 500, 500, 0, 0, 100, dVelocity, dAccel,
    dAccel, 1
AxmContiEndNode lCoordinate

AxmContiSetAbsRelMode lCoordinate, 1
AxmContiStart lCoordinate, 0, 0

```

Delphi Example

```

var
  dVelocity, dAccel : Double;
  i : LongInt;
  lAxis : array [0..3] of LongInt;
  lCoordinate : LongInt;

begin
  dVelocity := 2000;
  dAccel := 4000;
  lCoordinate := 0;

  for i := 0 to 3 do
  begin
    lAxis[i] := i;
  end;

  AxmContiWriteClear(lCoordinate);
  AxmContiSetAxisMap (lCoordinate, 4, @lAxis);
  AxmContiSetAbsRelMode(lCoordinate, 1);

  AxmContiBeginNode(lCoordinate);
  AxmHelixCenterMove(lCoordinate, 500, 500, 0, 0, 100, dVelocity, dAccel,
    dAccel, DIR_CW);
  AxmHelixCenterMove(lCoordinate, 500, 500, 0, 0, 100, dVelocity, dAccel,
    dAccel, DIR_CW);
  AxmHelixCenterMove(lCoordinate, 500, 500, 0, 0, 100, dVelocity, dAccel,
    dAccel, DIR_CW);
  AxmHelixCenterMove(lCoordinate, 500, 500, 0, 0, 100, dVelocity, dAccel,
    dAccel, DIR_CW);
  AxmHelixCenterMove(lCoordinate, 500, 500, 0, 0, 100, dVelocity, dAccel,
    dAccel, DIR_CW);
  AxmContiEndNode (lCoordinate);

  AxmContiSetAbsRelMode(lCoordinate, 1);
  AxmContiStart (lCoordinate, 0, 0);

```

```
end;
```

See Also

[AxmContiSetAxisMap](#), [AxmContiGetAxisMap](#), [AxmContiSetAbsRelMode](#), [AxmContiGetAbsRelMode](#),
[AxmContiBeginNode](#), [AxmContiEndNode](#), [AxmHelixPointMove](#), [AxmHelixRadiusMove](#), [AxmHelixAngleMove](#),
[AxmContiReadFree](#), [AxmContiReadIndex](#), [AxmContiWriteClear](#), [AxmContiStart](#), [AxmContiIsMotion](#),
[AxmContiGetNodeNum](#), [AxmContiGetTotalNodeNum](#),

AxmHelixPointMove

Purpose

API to move helical interpolation with specification of center point, end point for specific axis. It exits function at the point of beginning of pulse output.

Format

C

```
DWORD AxmHelixPointMove (long lCoord, double dMidXPos, double dMidYPos, double dEndXPos, double
dEndYPos, double dZPos, double dVel, double dAccel, double dDecel);
```

Visual Basic

```
Function AxmHelixPointMove (ByVal lCoord As Long, ByVal dMidXPos As Double, ByVal dMidYPos As Double, ByVal
dEndXPos As Double, ByVal dEndYPos As Double, ByVal dZPos As Double, ByVal dVel As Double, ByVal dAccel As
Double, ByVal dDecel As Double) As Long
```

Delphi

```
function AxmHelixPointMove (lCoord : LongInt; dMidXPos : Double; dMidYPos : Double; dEndXPos : Double;
dEndYPos : Double; dZPos : Double; dVel : Double; dAccel : Double; dDecel : Double) : DWord; stdcall;
```

Input/ Output

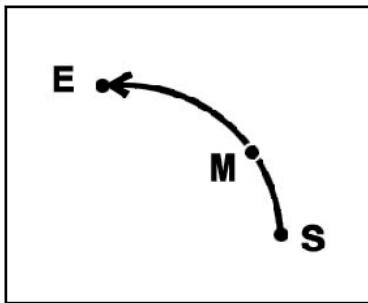
Name	in/out	Format	Init Value	Explanation
Coord	in	long	–	Coordination System number(starting from 0)
MidXPos	In	double		Midle position value of X axis
MidYPos	In	double		Midle position value of Y axis
EndXPos	In	double		End position value of X axis
EndYPos	In	double		End position value of Y axis
ZPos	In	double		Distance to move to the direction of Z axis at a time.
Vel	In	double		Move speed (Move speed unit is PPS[Pulses/sec] given that Unit/Pulse is 1/1)
Accel	In	double		Move acceleration(accleration unit is PPS[Pulses/sec^2] given that Unit/pulse is 1/1)
Decel	In	double		Move deceleration(deceleration unit is PPS[Pulses/sec^2] given that Unit/pulse is 1/1)

Return AXT_RT_SUCCESS(0000) : Successful execution of API.

[* See error code Table for more information on status error codes](#)

Description

API to move helical interpolation with specification of start point, end point, and radius. API to move helical interpolation with specification of middle point and end point for specific axis in using with [AxmContiBeginNode](#), [AxmContiEndNode](#). It saves to internal Queue for circular continuous interpolation move. It starts with the use of [AxmContiStart](#) API.(It is used with continuous interpolation API.) Current position becomes beginning point of circular move, and it process circular move going through middle point.



- ▶ By AxmSStop and AxmEstop API, slow down stop and emergency stop are possible. It always stops by transmitting master axis of Coordinate to parameter.
- ▶ [AxmStatusReadInMotion](#) API can verify whether it is on motion or not.
- ▶ When using [AxmSignalSetInpos](#) API, if Inpos is set to Enable, it doesn't return until INP input is set to ON although Command pulse output is completed because it considers move is not completed. Refer to [AxmSignalSetInpos](#) for futher information. **
- ▶ Caution: Axes which interpolate Unit/Pulse should be equally set. If Unit/Pulse is not equally set, their unit will different with each other. Always equally set, and do interpolation control.
- ▶ Caution: When using motion board with more than 8 (PCI-N804/404)
In the board with more than 8 axes, 4 axes from Axis 0–3 becomes axis group 1, 4 axes from Axis 4–7 becomes axis group 2, and 4 axes from Axis 8–11 becomes axis group 3. Axes from the same group can process helical Interpolation move, but axes from the different group cannot. The reason is that PCI-N804/404 chip has 4 axes, and it doesn't support this. Keep in mind when specify Coordinate.
- ▶ Caution: When using Helix on continuous interpolation, Spline, linear interpolation and circular interpolation can not be used together.

▶ continuous interpolation APIs list

Function	Description	Note
AxmContiSetAxisMap	Setting continous interpolation axis mapping for specified Coordinate system.	
AxmContiGetAxisMap	Setting continous interpolation axis mapping for specified Coordinate system.	
AxmContiSetAbsRelMode	Setting continous interpolation axis absolute/relative mode for specified coordinate system.	

AxmContiGetAbsRelMode	Return continous interpolation axis absolute/relative mode for specified coordinate system.	
AxmContiBeginNode	Start registering tasks which will be processed in continous interpolation for specified coordinate system. After calling this function, all motion tasks processed before calling AxmContiEndNode function are registered as continous interpolation motion, but not processing real motion. When AxmContiStart function is called, registered motion is actually started.	
AxmContiEndNode	End registering tasks to process continous interpolation in specified coordinate system.	
AxmContiReadFree	Identify internal Queue is empty for interpolation move in specified coordinate system.	
AxmContiReadIndex	Identify the number of interpolation move in internalQueue stored for interpolation move in specified coordinate system.	
AxmContiWriteClear	Clear all of the internal Queue stored for continuous interpolation move in a specified coordinate system.	
AxmContiStart	Start continuous interpolation move from internal Queue stored in a specified coordinate system.	
AxmContiIsMotion	Identify countinuous interpolation move is running or not in specified coordinate system.	
AxmContiGetNodeNum	Identify the index number of currently running continuous interpolation among continous interpolation moves in specified coordinate system.	
AxmContiGetTotalNodeNum	Identify total number of index of continuous interpolation move set in specified coordinate system.	

(Normal helical interpolation move example)

C Example

```

double dVelocity = 2000, dAccel = 4000;
long lAxis[4];
long lCoordinate = 0;

for(int i=0; i<4; i++)
{
    lAxis[i] = i;

AxmContiWriteClear(lCoordinate);
AxmContiSetAxisMap (lCoordinate, 4, lAxis);
AxmContiSetAbsRelMode(lCoordinate, 1);

AxmHelixPointMove(lCoordinate, 500, 500, 1000, 0, 100, dVelocity, dAccel,
dAccel);

```

VB Example

```

Dim dVelocity As Double
Dim dAccel As Double
Dim lCoordinate As Long

Dim axes(0 To 3) As Long
Dim i As Long
dVelocity = 2000
dAccel = 4000
lCoordinate = 0

For i = 0 To 3
    axes(i) = i

Next i

AxmContiWriteClear lCoordinate
AxmContiSetAxisMap lCoordinate, 4, axes(0)
AxmContiSetAbsRelMode lCoordinate, 1

AxmHelixPointMove lCoordinate, 500, 500, 1000, 0, 100, dVelocity, dAccel
dAccel

```

Delphi Example

```

var
  dVelocity, dAccel : Double;
  i : LongInt;
  lAxis : array [0..3] of LongInt;
  lCoordinate : LongInt;

begin
  dVelocity := 2000;
  dAccel := 4000;
  lCoordinate := 0;

  for i := 0 to 3 do
  begin
    lAxis[i] := i;
  end;

AxmContiWriteClear(lCoordinate);
AxmContiSetAxisMap (lCoordinate, 4, @lAxis);
AxmContiSetAbsRelMode(lCoordinate, 1);

```

```
AxmHelixPointMove(lCoordinate, 500, 500, 1000, 0, 100, dVelocity, dAccel,
                  dAccel);
end;
```

(continuous helical interpolation move example)

C Example

```
double dVelocity = 2000, dAccel = 4000;
long lAxis[4];
long lCoordinate = 0;

for(int i=0; i<4; i++)
{
    lAxis[i] = i;

AxmContiWriteClear(lCoordinate);
AxmContiSetAxisMap(lCoordinate, 4, lAxis);
AxmContiSetAbsRelMode(lCoordinate, 1);

AxmContiBeginNode(lCoordinate);
AxmHelixPointMove(lCoordinate, 500, 500, 1000, 0, 100, dVelocity, dAccel,
                  dAccel);
AxmHelixPointMove(lCoordinate, 500, 500, 1000, 0, 100, dVelocity, dAccel,
                  dAccel);
AxmHelixPointMove(lCoordinate, 500, 500, 1000, 0, 100, dVelocity, dAccel,
                  dAccel);
AxmHelixPointMove(lCoordinate, 500, 500, 1000, 0, 100, dVelocity, dAccel,
                  dAccel);
AxmHelixPointMove(lCoordinate, 500, 500, 1000, 0, 100, dVelocity, dAccel,
                  dAccel);
AxmContiEndNode(lCoordinate);

AxmContiSetAbsRelMode(lCoordinate, 1);
AxmContiStart(lCoordinate, 0, 0);
```

VB Example

```
Dim dVelocity As Double
Dim dAccel As Double
Dim lCoordinate As Long

Dim axes(0 To 3) As Long

Dim i As Long
dVelocity = 2000
dAccel = 4000
lCoordinate = 0

For i = 0 To 3
    axes(i) = i

Next i

AxmContiWriteClear lCoordinate
AxmContiSetAxisMap lCoordinate, 4, axes(0)
AxmContiSetAbsRelMode lCoordinate, 1

AxmContiBeginNode lCoordinate
AxmHelixPointMove lCoordinate, 500, 500, 1000, 0, 100, dVelocity, dAccel,
                  dAccel
```

```

AxmHelixPointMove lCoordinate, 500, 500, 1000, 0, 100, dVelocity, dAccel,
    dAccel
AxmHelixPointMove lCoordinate, 500, 500, 1000, 0, 100, dVelocity, dAccel,
    dAccel
AxmHelixPointMove lCoordinate, 500, 500, 1000, 0, 100, dVelocity, dAccel,
    dAccel
AxmHelixPointMove lCoordinate, 500, 500, 1000, 0, 100, dVelocity, dAccel,
    dAccel
AxmContiEndNode lCoordinate

AxmContiSetAbsRelMode lCoordinate, 1
AxmContiStart lCoordinate, 0, 0

```

Delphi Example

```

var
  dVelocity, dAccel : Double;
  i : LongInt;
  lAxis : array [0..3] of LongInt;
  lCoordinate : LongInt;

begin
  dVelocity := 2000;
  dAccel := 4000;
  lCoordinate := 0;

  for i := 0 to 3 do
  begin
    lAxis[i]      := i;
  end;

  AxmContiWriteClear(lCoordinate);
  AxmContiSetAxisMap (lCoordinate, 4, @lAxis);
  AxmContiSetAbsRelMode(lCoordinate, 1);

  AxmContiBeginNode(lCoordinate);
  AxmHelixPointMove(lCoordinate, 500, 500, 1000, 0, 100, dVelocity, dAccel,
    dAccel);
  AxmHelixPointMove(lCoordinate, 500, 500, 1000, 0, 100, dVelocity, dAccel,
    dAccel);
  AxmHelixPointMove(lCoordinate, 500, 500, 1000, 0, 100, dVelocity, dAccel,
    dAccel);
  AxmHelixPointMove(lCoordinate, 500, 500, 1000, 0, 100, dVelocity, dAccel,
    dAccel);
  AxmHelixPointMove(lCoordinate, 500, 500, 1000, 0, 100, dVelocity, dAccel,
    dAccel);
  AxmContiEndNode (lCoordinate);

  AxmContiSetAbsRelMode(lCoordinate, 1);
  AxmContiStart (lCoordinate, 0, 0);
end;

```

See Also

[AxmContiSetAxisMap](#), [AxmContiGetAxisMap](#), [AxmContiSetAbsRelMode](#), [AxmContiGetAbsRelMode](#),
[AxmContiBeginNode](#), [AxmContiEndNode](#), [AxmHelixCenterMove](#), [AxmHelixRadiusMove](#), [AxmHelixAngleMove](#),
[AxmContiReadFree](#), [AxmContiReadIndex](#), [AxmContiWriteClear](#), [AxmContiStart](#), [AxmContiIsMotion](#),
[AxmContiGetNodeNum](#), [AxmContiGetTotalNodeNum](#)

AxmHelixRadiusMove

Purpose

API to move helical interpolation with specification of start point, end point, and radius for specific axis. It exits function at the point of beginning of pulse output.

Format

C

```
DWORD AxmHelixRadiusMove (long lCoord, double dRadius, double dEndXPos, double dEndYPos, double dZPos,
    double dVel, double dAccel, double dDecel, DWORD uCWDdir, DWORD uShortDistance);
```

Visual Basic

```
Function AxmHelixRadiusMove (ByVal lCoord As Long, ByVal dRadius As Double, ByVal dEndXPos As Double, ByVal
    dEndYPos As Double, ByVal dZPos As Double, ByVal dVel As Double, ByVal dAccel As Double, ByVal dDecel As
    Double, ByVal uCWDdir As Long, ByVal uShortDistance As Long) As Long
```

Delphi

```
function AxmHelixRadiusMove (lCoord : LongInt; dRadius : Double; dEndXPos : Double; dEndYPos : Double;
    dZPos : Double; dVel : Double; dAccel : Double; dDecel : Double; uCWDdir : DWord; uShortDistance : DWord) :
    DWord; stdcall;
```

Input/ Output

Name	in/out	Format	Init Value	Explanation
Coord	in	long		Coordination System number(starting from 0)
dRadius	In	double		Circular radius for move
EndXPos	In	double		End position value of X axis
EndYPos	In	double		End position value of Y axis
ZPos	In	double		Distance to move to the direction of Z axis at a time.
Vel	In	double		Move speed (Move speed unit is PPS[Pulses/sec] given that Unit/Pulse is 1/1)
Accel	In	double		Move acceleration(accleration unit is PPS[Pulses/sec^2] given that Unit/pulse is 1/1)
Decel	In	double		Move deceleration(deceleration unit is PPS[Pulses/sec^2] given that Unit/pulse is 1/1)
CWDdir	In	DWORD		Circular direction
ShortDistance	In	DWORD		Distance of circle which goes to the middle point

CWDir

#define	Value	Explanation
DIR_CCW	00h	clockwise
DIR_CW	01h	counterclockwise

ShortDistance

#define	Value	Explanation
SHORT_DISTANCE	00h	Circular move for short distance
LONG_DISTANCE	01h	Circular move for long distance

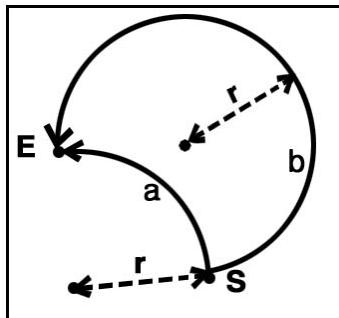
Return AXT_RT_SUCCESS(0000) : Successful execution of API.

[* See error code Table for more information on status error codes](#)

Description

API to move helical interpolation with specification of start point, middle point and radius for specific axis in using with [AxmContiBeginNode](#), [AxmContiEndNode](#). It saves to internal Queue for circular continuous interpolation move. It starts with the use of [AxmContiStart](#) API.(It is used with continuous interpolation API.) Current position becomes beginning point of circular move, and it process circular move in the basis of specified radius and end point.

When we define certain circular arc which connects beginning point and end point, we can have two middle points. Because of these, two paths(a,b) can be defined even if it has same rotational direction.



► By AxmSStop and AxmEstop function, slow down stop and emergency stop are possible. It always stops by transmitting master axis of Coordinate to parameter.

► [AxmStatusReadInMotion](#) API can verify whether it is on motion or not.

► When using [AxmSignalSetInpos](#) API, if Inpos is set to Enable, it doesn't return until INP input is set to ON although Command pulse output is completed because it considers move is not completed. Refer to [AxmSignalSetInpos](#) for futher information. **

► Caution: Axes which interpolate Unit/Pulse should be equally set. If Unit/Pulse is not equally set, their unit will different with each other. Always equally set, and do interpolation control.

► Caution: When using motion board with more than 8 (PCI-N804/404)

In the board with more than 8 axes, 4 axes from Axis 0–3 becomes axis group 1, 4 axes from Axis 4–7 becomes axis group 2, and 4 axes from Axis 8–11 becomes axis group 3. Axes from the same group

can process helical interpolation move, but axes from the different group cannot. The reason is that PCI-N804/404 chip has 4 axes, and it doesn't support this. Keep in mind when specify Coordinate.

►Caution: When using Helix on continuous interpolation, Spline, linear interpolation and circular interpolation can not be used together.

► continuous interpolation APIs list

Function	Description	Note
AxmContiSetAxisMap	Setting continuous interpolation axis mapping for specified Coordinate system.	
AxmContiGetAxisMap	Setting continuous interpolation axis mapping for specified Coordinate system.	
AxmContiSetAbsRelMode	Setting continuous interpolation axis absolute/relative mode for specified coordinate system.	
AxmContiGetAbsRelMode	Return continuous interpolation axis absolute/relative mode for specified coordinate system.	
AxmContiBeginNode	Start registering tasks which will be processed in continuous interpolation for specified coordinate system. After calling this function, all motion tasks processed before calling AxmContiEndNode function are registered as continuous interpolation motion, but not processing real motion. When AxmContiStart function is called, registered motion is actually started.	
AxmContiEndNode	End registering tasks to process continuous interpolation in specified coordinate system.	
AxmContiReadFree	Identify internal Queue is empty for interpolation move in specified coordinate system.	
AxmContiReadIndex	Identify the number of interpolation move in internalQueue stored for interpolation move in specified coordinate system.	
AxmContiWriteClear	Clear all of the internal Queue stored for continuous interpolation move in a specified coordinate system.	

<u>AxmContiStart</u>	Start continuous interpolation move from internal Queue stored in a specified coordinate system.	
<u>AxmContiIsMotion</u>	Identify countinuous interpolation move is running or not in specified coordinate system.	
<u>AxmContiGetNodeNum</u>	Identify the index number of currently running continuous interpolation among continuous interpolation moves in specified coordinate system.	
<u>AxmContiGetTotalNodeNum</u>	Identify total number of index of continuous interpolation move set in specified coordinate system.	

(Normal helical interpolation move example)

C Example

```
double dVelocity = 2000, dAccel = 4000;
long lAxis[4];
long lCoordinate = 0;

for(int i=0; i<4; i++)
{
    lAxis[i] = i;

AxmContiWriteClear(lCoordinate);
AxmContiSetAxisMap (lCoordinate, 4, lAxis);

AxmContiSetAbsRelMode (lCoordinate, 1);

AxmHelixRadiusMove(lCoordinate, 50, 100, 0, 100, dVelocity, dAccel,
dAccel, DIR_CW, SHORT_DISTANCE);
```

VB Example

```
Dim dVelocity As Double
Dim dAccel As Double
Dim lCoordinate As Long

Dim axes(0 To 3) As Long

Dim i As Long
dVelocity = 2000
dAccel = 4000
lCoordinate = 0

For i = 0 To 3
    axes(i) = i

Next i

AxmContiWriteClear lCoordinate
AxmContiSetAxisMap lCoordinate, 4, axes(0)
```

```
AxmContiSetAbsRelMode lCoordinate, 1
AxmHelixRadiusMove lCoordinate, 50, 100, 0, 100, dVelocity, dAccel,
    dAccel, DIR_CW, SHORT_DISTANCE
```

Delphi Example

```
var
  dVelocity, dAccel : Double;
  i : LongInt;
  lAxis : array [0..3] of LongInt;
  lCoordinate : LongInt;

begin
  dVelocity := 2000;
  dAccel := 4000;
  lCoordinate := 0;

  for i := 0 to 3 do
  begin
    lAxis[i] := i;
  end;

  AxmContiWriteClear(lCoordinate);
  AxmContiSetAxisMap (lCoordinate, 4, @lAxis);
  AxmContiSetAbsRelMode(lCoordinate, 1);

  AxmHelixRadiusMove(lCoordinate, 50, 100, 0, 100, dVelocity, dAccel,
    dAccel, DIR_CW, SHORT_DISTANCE);
end;
```

(continuous helical interpolation move example)

C Example

```
double dVelocity = 2000, dAccel = 4000;
long lAxis[4];
long lCoordinate = 0;

for(int i=0; i<4; i++)
{
    lAxis[i] = i;
}

AxmContiWriteClear(lCoordinate);
AxmContiSetAxisMap (lCoordinate, 4, lAxis);
AxmContiSetAbsRelMode(lCoordinate, 1);

AxmContiBeginNode(lCoordinate);
AxmHelixRadiusMove(lCoordinate, 50, 100, 0, 100, dVelocity, dAccel,
    dAccel, DIR_CW, SHORT_DISTANCE);
AxmHelixRadiusMove(lCoordinate, 50, 100, 0, 100, dVelocity, dAccel,
    dAccel, DIR_CW, SHORT_DISTANCE);
AxmHelixRadiusMove(lCoordinate, 50, 100, 0, 100, dVelocity, dAccel,
    dAccel, DIR_CW, SHORT_DISTANCE);
AxmHelixRadiusMove(lCoordinate, 50, 100, 0, 100, dVelocity, dAccel,
    dAccel, DIR_CW, SHORT_DISTANCE);
AxmContiEndNode (lCoordinate);

AxmContiSetAbsRelMode(lCoordinate, 1);
AxmContiStart (lCoordinate , 0, 0);
```

VB Example

```

Dim dVelocity As Double
Dim dAccel As Double
Dim lCoordinate As Long

Dim axes(0 To 3) As Long

Dim i As Long
dVelocity = 2000
dAccel = 4000
lCoordinate = 0

For i = 0 To 3

    axes(i) = i

Next i

AxmContiWriteClear lCoordinate
AxmContiSetAxisMap lCoordinate, 4, axes(0)
AxmContiSetAbsRelMode lCoordinate, 1
AxmContiBeginNode lCoordinate
AxmHelixRadiusMove lCoordinate, 50, 100, 0, 100, dVelocity, dAccel,
    dAccel, DIR_CW, SHORT_DISTANCE
AxmHelixRadiusMove lCoordinate, 50, 100, 0, 100, dVelocity, dAccel,
    dAccel, DIR_CW, SHORT_DISTANCE
AxmHelixRadiusMove lCoordinate, 50, 100, 0, 100, dVelocity, dAccel,
    dAccel, DIR_CW, SHORT_DISTANCE
AxmHelixRadiusMove lCoordinate, 50, 100, 0, 100, dVelocity, dAccel,
    dAccel, DIR_CW, SHORT_DISTANCE
AxmHelixRadiusMove lCoordinate, 50, 100, 0, 100, dVelocity, dAccel,
    dAccel, DIR_CW, SHORT_DISTANCE
AxmContiEndNode lCoordinate

AxmContiSetAbsRelMode lCoordinate, 1
AxmContiStart lCoordinate, 0, 0

```

Delphi Example

```

var
  dVelocity, dAccel : Double;
  i : LongInt;
  lAxis : array [0..3] of LongInt;
  lCoordinate : LongInt;

begin
  dVelocity := 2000;
  dAccel := 4000;
  lCoordinate := 0;

  for i := 0 to 3 do
  begin
    lAxis[i] := i;
  end;

  AxmContiWriteClear(lCoordinate);
  AxmContiSetAxisMap (lCoordinate, 4, @lAxis);
  AxmContiSetAbsRelMode(lCoordinate, 1);
  AxmContiBeginNode(lCoordinate);
  AxmHelixRadiusMove(lCoordinate, 50, 100, 0, 100, dVelocity, dAccel,
    dAccel, DIR_CW, SHORT_DISTANCE);
  AxmHelixRadiusMove(lCoordinate, 50, 100, 0, 100, dVelocity, dAccel,
    dAccel, DIR_CW, SHORT_DISTANCE);
  AxmHelixRadiusMove(lCoordinate, 50, 100, 0, 100, dVelocity, dAccel,
    dAccel, DIR_CW, SHORT_DISTANCE);

```

```
AxmHelixRadiusMove(lCoordinate, 50, 100, 0, 100, dVelocity, dAccel,  
    dAccel, DIR_CW, SHORT_DISTANCE);  
AxmHelixRadiusMove(lCoordinate, 50, 100, 0, 100, dVelocity, dAccel,  
    dAccel, DIR_CW, SHORT_DISTANCE);  
AxmContiEndNode (lCoordinate);  
  
AxmContiSetAbsRelMode(lCoordinate, 1);  
AxmContiStart (lCoordinate, 0, 0);  
end;
```

See Also

[AxmContiSetAxisMap](#), [AxmContiGetAxisMap](#), [AxmContiSetAbsRelMode](#), [AxmContiGetAbsRelMode](#),
[AxmContiBeginNode](#), [AxmContiEndNode](#), [AxmHelixCenterMove](#), [AxmHelixPointMove](#), [AxmHelixAngleMove](#),
[AxmContiReadFree](#), [AxmContiReadIndex](#), [AxmContiWriteClear](#), [AxmContiStart](#), [AxmContiIsMotion](#),
[AxmContiGetNodeNum](#), [AxmContiGetTotalNodeNum](#)

AxmHelixAngleMove

Purpose

API to move helical interpolation with specification of start point, rotational angle, and radius for specific axis. It exits function at the point of beginning of pulse output.

Format

C

```
DWORD AxmHelixAngleMove (long lCoord, double dCenterXPos, double dCenterYPos, double dAngle, double
dZPos, double dVel, double dAccel, double dDecel, DWORD uCWDdir);
```

Visual Basic

```
Function AxmHelixAngleMove (ByVal lCoord As Long, ByVal dCenterXPos As Double, ByVal dCenterYPos As Double,
ByVal dAngle As Double, ByVal dZPos As Double, ByVal dVel As Double, ByVal dAccel As Double, ByVal dDecel As
Double, ByVal uCWDdir As Long) As Long
```

Delphi

```
function AxmHelixAngleMove (lCoord : LongInt; dCenterXPos : Double; dCenterYPos : Double; dAngle : Double;
dZPos : Double; dVel : Double; dAccel : Double; dDecel : Double; uCWDdir : DWord) : DWord; stdcall;
```

Input/ Output

Name	in/out	Format	Init Value	Explanation
Coord	in	long		Coordination System number(starting from 0)
CenterXPos	In	double		Middle position value of X axis
CenterYPos	In	double		Middle position value of Y axis
Angle	In	double		Angle for circular move. Unit is “
ZPos	In	double		Distance to move to the direction of Z axis at a time.
Vel	In	double		Move speed (Move speed unit is PPS[Pulses/sec] given that Unit/Pulse is 1/1)
Accel	In	double		Move acceleration(accleration unit is PPS[Pulses/sec^2] given that Unit/pulse is 1/1)
Decel	In	double		Move deceleration(deceleration unit is PPS[Pulses/sec^2] given that Unit/pulse is 1/1)
CWDdir	In	DWORD		Circular direction

CWDdir

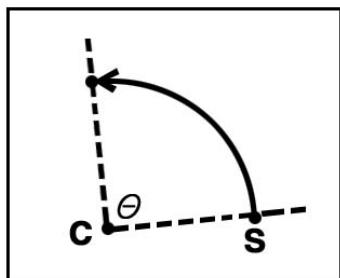
#define	Value	Explanation

DIR_CCW	00h	clockwise
DIR_CW	01h	counterclockwise

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

Description

API to move helical interpolation with specification of start point, rotational angle and radius for specified coordinate system. API to move helical interpolation with specification of start point, rotational angle and radius for specific coordinate system in using with [AxmContiBeginNode](#), [AxmContiEndNode](#). It saves to internal Queue for circular continuous interpolation move. It starts with the use of [AxmContiStart](#) API.(It is used with continuous interpolation API.) Current position becomes beginning point of circular move, and it process circular move to specified angle in the basis of end point.



- ▶ By AxmSStop and AxmEstop function, slow down stop and emergency stop are possible. It always stops by transmitting master axis of Coordinate to parameter.
- ▶ [AxmStatusReadInMotion](#) API can verify whether it is on motion or not.
- ▶ When using [AxmSignalSetInpos](#) API, if Inpos is set to Enable, it doesn't return until INP input is set to ON although Command pulse output is completed because it considers move is not completed. Refer to [AxmSignalSetInpos](#) for futher information. **
- ▶ Caution: Axes which interpolate Unit/Pulse should be equally set. If Unit/Pulse is not equally set, their unit will different with each other. Always equally set, and do interpolation control.
- ▶ Caution: When using motion board with more than 8 (PCI-N804/404)
 In the board with more than 8 axes, 4 axes from Axis 0–3 becomes axis group 1, 4 axes from Axis 4–7 becomes axis group 2, and 4 axes from Axis 8–11 becomes axis group 3. Axes from the same group can process helical interpolation move, but axes from the different group cannot. The reason is that PCI-N804/404 chip has 4 axes, and it doesn't support this. Keep in mind when specify Coordinate.
- ▶ Caution: When using Helix on continuous interpolation, Spline, linear interpolation and circular interpolation can not be used together.

▶ continuous interpolation APIs list

Function	Description	Note
AxmContiSetAxisMap	Setting continous interpolation axis mapping	

	for specified Coordinate system.	
AxmContiGetAxisMap	Setting continuous interpolation axis mapping for specified Coordinate system.	
AxmContiSetAbsRelMode	Setting continuous interpolation axis absolute/relative mode for specified coordinate system.	
AxmContiGetAbsRelMode	Return continuous interpolation axis absolute/relative mode for specified coordinate system.	
AxmContiBeginNode	Start registering tasks which will be processed in continuous interpolation for specified coordinate system. After calling this function, all motion tasks processed before calling AxmContiEndNode function are registered as continuous interpolation motion, but not processing real motion. When AxmContiStart function is called, registered motion is actually started.	
AxmContiEndNode	End registering tasks to process continuous interpolation in specified coordinate system.	
AxmContiReadFree	Identify internal Queue is empty for interpolation move in specified coordinate system.	
AxmContiReadIndex	Identify the number of interpolation move in internalQueue stored for interpolation move in specified coordinate system.	
AxmContiWriteClear	Clear all of the internal Queue stored for continuous interpolation move in a specified coordinate system.	
AxmContiStart	Start continuous interpolation move from internal Queue stored in a specified coordinate system.	
AxmContiIsMotion	Identify continuous interpolation move is running or not in specified coordinate system.	
AxmContiGetNodeNum	Identify the index number of currently running continuous interpolation among continuous	

	interpolation moves in specified coordinate system.	
AxmContiGetTotalNodeNum	Identify total number of index of continuous interpolation move set in specified coordinate system.	

(Normal helical interpolation move example)

C Example

```
double dVelocity = 2000, dAccel = 4000;
long lAxis[4];
long lCoordinate = 0;

for(int i=0; i<4; i++)
{
    lAxis[i] = i;

AxmContiWriteClear(lCoordinate);
AxmContiSetAxisMap(lCoordinate, 4, lAxis);
AxmContiSetAbsRelMode(lCoordinate, 1);

AxmHelixAngleMove(lCoordinate, 500, 500, 360, 100, dVelocity, dAccel,
dAccel, DIR_CW);
```

VB Example

```
Dim dVelocity As Double
Dim dAccel As Double
Dim lCoordinate As Long

Dim axes(0 To 3) As Long

Dim i As Long
dVelocity = 2000
dAccel = 4000
lCoordinate = 0

For i = 0 To 3
    axes(i) = i

Next i

AxmContiWriteClear lCoordinate
AxmContiSetAxisMap lCoordinate, 4, axes(0)
AxmContiSetAbsRelMode lCoordinate, 1

AxmHelixAngleMove lCoordinate, 500, 500, 360, 100, dVelocity, dAccel,
dAccel, DIR_CW
```

Delphi Example

```
var
  dVelocity, dAccel : Double;
  i : LongInt;
  lAxis : array [0..3] of LongInt;
  lCoordinate : LongInt;

begin
```

```

dVelocity := 2000;
dAccel := 4000;
lCoordinate := 0;

for i := 0 to 3 do
begin
    lAxis[i]           := i;
end;

AxmContiWriteClear(lCoordinate);
AxmContiSetAxisMap (lCoordinate, 4, @lAxis);
AxmContiSetAbsRelMode(lCoordinate, 1);

AxmHelixAngleMove(lCoordinate, 500, 500, 360, 100, dVelocity, dAccel,
                  dAccel, DIR_CW);
end;

```

(continuous helical interpolation move example)

C Example

```

double dVelocity = 2000, dAccel = 4000;
long   lAxis[4];
long   lCoordinate = 0;

for(int i=0; i<4; i++)
{
    lAxis[i]           = i;
}

AxmContiWriteClear(lCoordinate);
AxmContiSetAxisMap (lCoordinate, 4, lAxis);
AxmContiSetAbsRelMode(lCoordinate, 1);

AxmContiBeginNode(lCoordinate);
AxmHelixAngleMove(lCoordinate, 500, 500, 360, 100, dVelocity, dAccel,
                  dAccel, DIR_CW);
AxmHelixAngleMove(lCoordinate, 500, 500, 360, 100, dVelocity, dAccel,
                  dAccel, DIR_CW);
AxmHelixAngleMove(lCoordinate, 500, 500, 360, 100, dVelocity, dAccel,
                  dAccel, DIR_CW);
AxmHelixAngleMove(lCoordinate, 500, 500, 360, 100, dVelocity, dAccel,
                  dAccel, DIR_CW);
AxmContiEndNode (lCoordinate);

AxmContiSetAbsRelMode(lCoordinate, 1);
AxmContiStart (lCoordinate , 0, 0);

```

VB Example

```

Dim dVelocity As Double
Dim dAccel As Double
Dim lCoordinate As Long

Dim axes(0 To 3) As Long

Dim i As Long
dVelocity = 2000
dAccel = 4000
lCoordinate = 0

For i = 0 To 3

```

```

axes(i) = i

Next i

AxmContiWriteClear lCoordinate
AxmContiSetAxisMap lCoordinate, 4, axes(0)
AxmContiSetAbsRelMode lCoordinate, 1

AxmContiBeginNode lCoordinate
AxmHelixAngleMove lCoordinate, 500, 500, 360, 100, dVelocity, dAccel,
    dAccel, DIR_CW
AxmHelixAngleMove lCoordinate, 500, 500, 360, 100, dVelocity, dAccel,
    dAccel, DIR_CW
AxmHelixAngleMove lCoordinate, 500, 500, 360, 100, dVelocity, dAccel,
    dAccel, DIR_CW
AxmHelixAngleMove lCoordinate, 500, 500, 360, 100, dVelocity, dAccel,
    dAccel, DIR_CW
AxmHelixAngleMove lCoordinate, 500, 500, 360, 100, dVelocity, dAccel,
    dAccel, DIR_CW
AxmContiEndNode lCoordinate

AxmContiSetAbsRelMode lCoordinate, 1
AxmContiStart lCoordinate, 0, 0

```

Delphi Example

```

var
  dVelocity, dAccel : Double;
  i : LongInt;
  lAxis : array [0..3] of LongInt;
  lCoordinate : LongInt;

begin
  dVelocity := 2000;
  dAccel := 4000;
  lCoordinate := 0;

  for i := 0 to 3 do
  begin
    lAxis[i] := i;
  end;

  AxmContiWriteClear(lCoordinate);
  AxmContiSetAxisMap (lCoordinate, 4, @lAxis);
  AxmContiSetAbsRelMode(lCoordinate, 1);

  AxmContiBeginNode(lCoordinate);
  AxmHelixAngleMove(lCoordinate, 500, 500, 360, 100, dVelocity, dAccel,
    dAccel, DIR_CW);
  AxmHelixAngleMove(lCoordinate, 500, 500, 360, 100, dVelocity, dAccel,
    dAccel, DIR_CW);
  AxmHelixAngleMove(lCoordinate, 500, 500, 360, 100, dVelocity, dAccel,
    dAccel, DIR_CW);
  AxmHelixAngleMove(lCoordinate, 500, 500, 360, 100, dVelocity, dAccel,
    dAccel, DIR_CW);
  AxmHelixAngleMove(lCoordinate, 500, 500, 360, 100, dVelocity, dAccel,
    dAccel, DIR_CW);
  AxmContiEndNode (lCoordinate);

  AxmContiSetAbsRelMode(lCoordinate, 1);
  AxmContiStart (lCoordinate, 0, 0);
end;

```

See Also

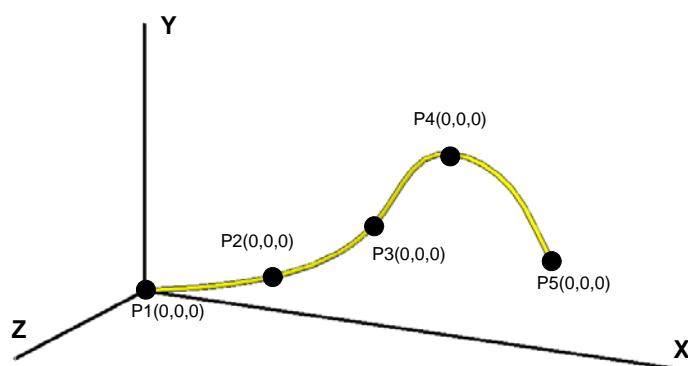
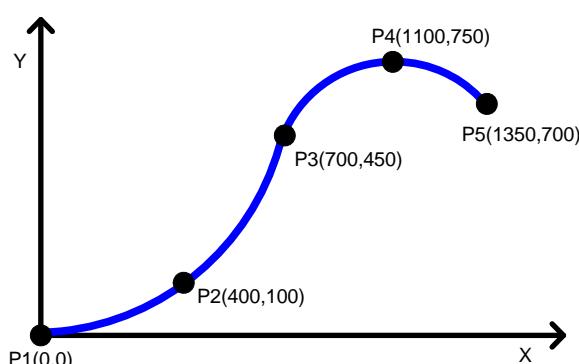
[AxmContiSetAxisMap](#), [AxmContiGetAxisMap](#), [AxmContiSetAbsRelMode](#), [AxmContiGetAbsRelMode](#),
[AxmContiBeginNode](#), [AxmContiEndNode](#), [AxmHelixCenterMove](#), [AxmHelixPointMove](#), [AxmHelixRadiusMove](#),
[AxmContiReadFree](#), [AxmContiReadIndex](#), [AxmContiWriteClear](#), [AxmContiStart](#), [AxmContiIsMotion](#),
[AxmContiGetNodeNum](#), [AxmContiGetTotalNodeNum](#)

Spline interpolation setting and move API

This API support only PCI-N804/404 specialized API.

Spline interpolation control provides B Spline interpolation function. B Spline interpolation executes interpolation by changing data points which the user specifies to free curve. To process this interpolation as a direct motion, it should be used with continuous interpolation motion function. It supports basically axis X, Y, Z. For 4 axes motion board it can select three axes from X(0),Y(1),Z(2),U(3). For 8 axes motion board it can selects three axes from X(0),Y(1),Z(2),U(3) in one chip and three axes from X(4),Y(5),Z(6),U(7) in another chip.

Function	Description
<u>AxmSplineWrite</u>	It is used with <u>AxmContiBeginNode</u> , <u>AxmContiEndNode</u> . It processes spline continuous interpolation move. It stores to internal queue for circular continuous interpolation move. It starts by using <u>AxmContiStart</u> API.(It is used with continuous interpolation function)



AxmSplineWrite

Purpose

Following Pos Point which user specified, it processes B Spline interpolation, which saves curve data to array user specified. This API supports spline for 2 axes and 3 axes, and used with [AxmContiStart](#) API.

Format

C

```
DWORD AxmSplineWrite (long lCoord, long lPosSize, double *dpPosX, double *dpPosY, double dVel, double dAccel,
double dDecel, double dPosZ, long lPointFactor);
```

Visual Basic

```
Function AxmSplineWrite (ByVal lCoord As Long, ByVal lPosSize As Long, ByRef dpPosX As Double, ByRef dpPosY
As Double, ByVal dVel As Double, ByVal dAccel As Double, ByVal dDecel As Double, ByVal dPosZ As Double, ByVal
lPointFactor As Long) As Long
```

Delphi

```
function AxmSplineWrite (lCoord : LongInt; lPosSize : LongInt; dpPosX : PDouble; dpPosY : PDouble; dVel : Double;
dAccel : Double; dDecel : Double; dPosZ : Double; lPointFactor: LongInt) : DWord; stdcall;
```

Input/ Output

Name	in/out	Format	Init Value	Explanation
Coord	in	long	-	Coordination System number(starting from 0)
PosSize	In	long		The number of Points for moving(minimum 3)
PosX	In	double*		Array of X values for moving
PosY	In	double*		Array of Y values for moving
Vel	In	double		Move speed (Move speed unit is PPS[Pulses/sec] given that Unit/Pulse is 1/1)
Accel	In	double		Move acceleration(acceleration unit is PPS[Pulses/sec^2] given that Unit/pulse is 1/1)
Decel	In	double		Move deceleration(deceleration unit is PPS[Pulses/sec^2] given that Unit/pulse is 1/1)
PosZ		double		+Z value for each position when using Z axis. Initial value is set to 0. It is used in 3 dimentional Spline.
PointFactor	In	long		Factor of Point. Minimum is 10. Path of point deffers by Factor.

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

Description

Spline interpolation function is to automatically generate points of free curve by using BSpline interpolation technique based on points user specified. To move this directly, it should be used with continuous interpolation.

- ▶ [AxmStatusReadInMotion](#) API can verify whether it is on motion or not.
- ▶ When using [AxmSignalSetInpos](#) API, if Inpos is set to Enable, it doesn't return until INP input is set to ON although Command pulse output is completed because it considers move is not completed. Refer to [AxmSignalSetInpos](#) for futher information. **
- ▶ Caution: Axes which interpolate Unit/Pulse should be equally set. If Unit/Pulse is not equally set, their unit will different with each other. Always set equally and do interpolation control.
- ▶ Caution: When using motion board with more than 8 (PCI-N804/404)
 In the board with more than 8 axes, 4 axes from Axis 0–3 becomes axis group 1, 4 axes from Axis 4–7 becomes axis group 2, and 4 axes from Axis 8–11 becomes axis group 3. Axes from the same group can process spline interpolation move, but axes from the different group cannot. The reason is that PCI-N804/404 chip has 4 axes, and it doesn't support this. Keep in mind when specifying Coordinate.
- ▶ Caution: when using spline on continuous interpolation, Helix, linear interpolation and circular interpolation can not be used together.

▶ continuous interpolation API list

Function	Description	Note
AxmContiSetAxisMap	Setting continous interpolation axis mapping for specified Coordinate system.	
AxmContiGetAxisMap	Setting continous interpolation axis mapping for specified Coordinate system.	
AxmContiSetAbsRelMode	Setting continous interpolation axis absolute/relative mode for specified coordinate system.	
AxmContiGetAbsRelMode	Return continous interpolation axis absolute/relative mode for specified coordinate system.	
AxmContiBeginNode	Start registering tasks which will be processed in continous interpolation for specified coordinate system. After calling this function, all motion tasks processed before calling AxmContiEndNode function are registered as continous interpolation motion, but not	

	processing real motion. When AxmContiStart function is called, registered motion is actually started.	
AxmContiEndNode	End registering tasks to process continuous interpolation in specified coordinate system.	
AxmContiReadFree	Identify internal Queue is empty for interpolation move in specified coordinate system.	
AxmContiReadIndex	Identify the number of interpolation move in internalQueue stored for interpolation move in specified coordinate system.	
AxmContiWriteClear	Clear all of the internal Queue stored for continuous interpolation move in a specified coordinate system.	
AxmContiStart	Start continuous interpolation move from internal Queue stored in a specified coordinate system.	
AxmContiIsMotion	Identify continuous interpolation move is running or not in specified coordinate system.	
AxmContiGetNodeNum	Identify the index number of currently running continuous interpolation among continuous interpolation moves in specified coordinate system.	
AxmContiGetTotalNodeNum	Identify total number of index of continuous interpolation move set in specified coordinate system.	

C Example

```

double dVelocity = 2000, dAccel = 4000;
long lAxis[3];
double dPosX[5];
double dPosY[5];
double dPosZ[5];
int iSize = 3;
long lCoordinate = 0;

for(int i=0; i<3; i++)
{
    lAxis[i] = i;
}

AxmContiWriteClear(lCoordinate);

```

```

AxmContiSetAxisMap (lCoordinate, 3, lAxis);
AxmContiSetAbsRelMode(lCoordinate, 0);

dPosX[0] = 0 , dPosY[0] = 0;
dPosX[1] = 450, dPosY[1] = 100;
dPosX[2] = 750, dPosY[2] = 450;
dPosX[3] = 1080, dPosY[3] = 750;
dPosX[4] = 1300, dPosY[4] = 700;

AxmContiBeginNode(lCoordinate);
AxmSplineWrite(lCoordinate, 5, dPosX, dPosY, dVelocity, dAccel, dAccel, 1,
               100);
AxmContiEndNode (lCoordinate);

AxmContiSetAbsRelMode(lCoordinate, 0);
AxmContiStart (lCoordinate, 0, 0);

```

VB Example

```

Dim dVelocity As Double
Dim dAccel As Double
Dim lPosSize As Long
Dim lCoordinate As Long

Dim dPosX(0 To 4) As Double
Dim dPosY(0 To 4) As Double
Dim dPosZ(0 To 4) As Double

Dim axes(0 To 2) As Long

Dim i As Long
dVelocity = 2000
dAccel = 4000
lCoordinate = 0

For i = 0 To 2
    axes(i) = i
Next i

lCoordinate = 0

AxmContiWriteClear lCoordinate
AxmContiSetAxisMap lCoordinate, 3, axes(0)
AxmContiSetAbsRelMode lCoordinate, 0

dPosX(0) = 0: dPosY(0) = 0
dPosX(1) = 450: dPosY(1) = 100
dPosX(2) = 750: dPosY(2) = 450
dPosX(3) = 1080: dPosY(3) = 750
dPosX(4) = 1300: dPosY(4) = 700

AxmContiBeginNode lCoordinate
AxmSplineWrite lCoordinate, 5, dPosX(0), dPosY(0), dVelocity, dAccel,
                dAccel, 1, 100
AxmContiEndNode lCoordinate

AxmContiSetAbsRelMode lCoordinate, 0
AxmContiStart lCoordinate, 0, 0

```

Delphi Example

```

var
  i : LongInt;
  dVelocity, dAccel : Double;
  lAxis : array [0..2] of LongInt;

```

```
dPosX : array [0..4] of Double;
dPosY : array [0..4] of Double;
lCoordinate : LongInt;
begin

  dVelocity := 2000;
  dAccel := 4000;
  lCoordinate := 0;

  for i := 0 to 2 do
  begin
    lAxis[i] := i;
  end;

  AxmContiWriteClear(lCoordinate);
  AxmContiSetAxisMap (lCoordinate, 3, @lAxis);
  AxmContiSetAbsRelMode(lCoordinate, 0);

  dPosX[0] := 0;   dPosY[0] := 0;
  dPosX[1] := 450; dPosY[1] := 240;
  dPosX[2] := 750; dPosY[2] := 610;
  dPosX[3] := 1080; dPosY[3] := 500;
  dPosX[4] := 1300; dPosY[4] := 800;

  AxmContiBeginNode(lCoordinate);
  AxmSplineWrite(lCoordinate, 5, @dPosX, @dPosY, dVelocity, dAccel,
                 dAccel,
                 1, 100);
  AxmContiEndNode (lCoordinate);

  AxmContiSetAbsRelMode(lCoordinate, 0);
  AxmContiStart (lCoordinate, 0, 0);

end;
```

See Also

[AxmContiSetAxisMap](#), [AxmContiGetAxisMap](#), [AxmContiSetAbsRelMode](#), [AxmContiGetAbsRelMode](#),
[AxmContiBeginNode](#), [AxmContiEndNode](#), [AxmContiReadFree](#), [AxmContiReadIndex](#),
[AxmContiWriteClear](#), [AxmContiStart](#), [AxmContiIsMotion](#), [AxmContiGetNodeNum](#),
[AxmContiGetTotalNodeNum](#)

Error Code Table Check

In order to check and process error code during library use, you can return error code. Error code is returning argument for each API. When an error occurs, you can check error code in order to know why this API is not in its function. For the stability of system and program debugging, you can check the status of AXT_RT_SUCCESS.

Error Code	Description
AXT_RT_SUCCESS(0000)	API function executed successfully
AXT_RT_OPEN_ERROR (1001)	Library is not open
AXT_RT_OPEN_ALREADY (1002)	Library is already open
AXT_RT_NOT_OPEN (1053)	Failed to initialize the library
AXT_RT_NOT_SUPPORT_VERSION(1054)	Hardware not supported
AXT_RT_INVALID_BOARD_NO (1101)	Invalid board number
AXT_RT_INVALID_MODULE_POS (1102)	Invalid module position
AXT_RT_INVALID_LEVEL (1103)	Invalid level
AXT_RT_ERROR_VERSION_READ(1151)	Failed to read library version
AXT_RT_1ST_BELOW_MIN_VALUE (1160)	First parameter is below the minimum value
AXT_RT_1ST_ABOVE_MAX_VALUE (1161)	First parameter is above the maximum value
AXT_RT_2ND_BELOW_MIN_VALUE (1170)	Second parameter is below the minimum value
AXT_RT_2ND_ABOVE_MAX_VALUE (1171)	Second parameter is above the maximum value
AXT_RT_3RD_BELOW_MIN_VALUE (1180)	Third parameter is below the minimum value
AXT_RT_3RD_ABOVE_MAX_VALUE (1181)	Third parameter is above the maximum value
AXT_RT_4TH_BELOW_MIN_VALUE (1190)	Fourth parameter is below the minimum value
AXT_RT_4TH_ABOVE_MAX_VALUE (1191)	Fourth parameter is above the maximum value
AXT_RT_5TH_BELOW_MIN_VALUE (1200)	Fifth parameter is below the minimum value
AXT_RT_5TH_ABOVE_MAX_VALUE (1201)	Fifth parameter is above the maximum value
AXT_RT_6TH_BELOW_MIN_VALUE (1210)	Sixth parameter is below the minimum value
AXT_RT_6TH_ABOVE_MAX_VALUE (1211)	Sixth parameter is above the maximum value
AXT_RT_7TH_BELOW_MIN_VALUE (1220)	Seventh parameter is below the minimum value
AXT_RT_7TH_ABOVE_MAX_VALUE (1221)	Seventh parameter is above the maximum value
sAXT_RT_8TH_BELOW_MIN_VALUE (1230)	Eighth parameter is below the minimum value
AXT_RT_8TH_ABOVE_MAX_VALUE (1231)	Eighth parameter is above the maximum value
AXT_RT_9TH_BELOW_MIN_VALUE (1240)	Nineth parameter is below the minimum value
AXT_RT_9TH_ABOVE_MAX_VALUE (1241)	Nineth parameter is above the maximum value
AXT_RT_AIO_OPEN_ERROR (2001)	Failed to open AIO module
AXT_RT_AIO_NOT_MODULE (2051)	No AIO module

AXT_RT_AIO_NOT_EVENT (2052)	Failed to read AIO event
AXT_RT_AIO_INVALID_MODULE_NO (2101)	Invalid AIO module
AXT_RT_AIO_INVALID_CHANNEL_NO (2102)	Invalid AIO channel number
AXT_RT_AIO_INVALID_USE (2106)	Can not use AIO
AXT_RT_AIO_INVALID_TRIGGER_MODE (2107)	Invalid trigger mode
AXT_RT_DIO_OPEN_ERROR (3001)	Failed to open DIO module
AXT_RT_DIO_NOT_MODULE (3051)	No DIO module
AXT_RT_DIO_NOT_INTERRUPT (3052)	DIO Interrupt not set
AXT_RT_DIO_INVALID_MODULE_NO (3101)	Invalid DIO module number
AXT_RT_DIO_INVALID_OFFSET_NO (3102)	Invalid DIO OFFSET number
AXT_RT_DIO_INVALID_LEVEL (3103)	Invalid DIO level
AXT_RT_DIO_INVALID_MODE (3104)	Invalid DIO mode
AXT_RT_DIO_INVALID_VALUE (3105)	Invalid setting value
AXT_RT_DIO_INVALID_USE (3106)	Can not use DIO API function
AXT_RT_MOTION_OPEN_ERROR(4001)	Failed to open motion library
AXT_RT_MOTION_NOT_MODULE(4051)	No motion module is installed in the system
AXT_RT_MOTION_NOT_INTERRUPT(4052)	Failed to read the resul of interrupt
AXT_RT_MOTION_NOT_INITIAL_AXIS_NO(4053)	Failed to initialize the motion of corresponding axis
AXT_RT_MOTION_NOT_IN_CONT_INTERPOL(4054)	Command to stop continuous interpolation while not in continuous interpolation motion
AXT_RT_MOTION_NOT_PARA_READ(4055)	Failed to load parameters for home return drive
AXT_RT_MOTION_INVALID_AXIS_NO(4101)	Corresponding axis does not exist
AXT_RT_MOTION_INVALID_METHOD(4102)	Invalid setting for corresponding axis drive
AXT_RT_MOTION_INVALID_USE(4103)	Invalid setting for 'uUse' parameter
AXT_RT_MOTION_INVALID_LEVEL(4104)	Invalid setting for 'uLevel' parameter
AXT_RT_MOTION_INVALID_BIT_NO(4105)	Invalid setting for general purpose input/output bit
AXT_RT_MOTION_INVALID_STOP_MODE(4106)	Invalid setting values for motion stop mode
AXT_RT_MOTION_INVALID_TRIGGER_MODE(4107)	Invalid setting for trigger setting mode
AXT_RT_MOTION_INVALID_TRIGGER_LEVEL(4108)	Invalid setting for trigger output level
AXT_RT_MOTION_INVALID_SELECTION(4109)	'uSelection' parameter is set to a value other than COMMAND or ACTUAL
AXT_RT_MOTION_INVALID_TIME(4110)	Invalid setting for Trigger output time value
AXT_RT_MOTION_INVALID_FILE_LOAD(4111)	Failed to load the file containing motion setting values
AXT_RT_MOTION_INVALID_FILE_SAVE(4112)	Failed to save the file containing motion setting values

AXT_RT_MOTION_INVALID_VELOCITY(4113)	Motion error occurred since the motion drive velocity value is set as zero
AXT_RT_MOTION_INVALID_ACCELTIME(4114)	Motion error occurred since motion drive acceleration time value is set as zero
AXT_RT_MOTION_INVALID_PULSE_VALUE(4115)	Input pulse value is set to a value less than zero when setting motion unit
AXT_RT_MOTION_INVALID_NODE_NUMBER (4116)	Called position or velocity override function while not in motion
AXT_RT_MOTION_INVALID_TARGET(4117)	Flag indicating the cause of multi axis motion stop
AXT_RT_MOTION_ERROR_IN_NONMOTION(4151)	Not in motion drive when it should be
AXT_RT_MOTION_ERROR_IN_MOTION(4152)	Called another motion drive fuction before the completetion of current motion
AXT_RT_MOTION_ERROR(4153)	Error occurred during the execution of multi axis motion stop function
AXT_RT_MOTION_ERROR_GANTRY_ENABLE(4154)	Gantry-enable is activated again while in motion with gantry enabled
AXT_RT_MOTION_ERROR_GANTRY_AXIS(4155)	Invalid input of gantry axis master channel (axis) number (starting from zero)
AXT_RT_MOTION_ERROR_MASTER_SERVOON(4156)	Servo ON is not enabled for the master axis
AXT_RT_MOTION_ERROR_SLAVE_SERVOON(4157)	Servo ON is not enabled for the slavve axis
AXT_RT_MOTION_INVALID_POSITION(4158)	Not in a valid position
AXT_RT_ERROR_NOT_SAME_MODULE (4159)	Not in the same module
AXT_RT_ERROR_NOT_SAME_BOARD (4160)	Not in the same board
AXT_RT_ERROR_NOT_SAME_PRODUCT (4161)	When products are different each other
AXT_RT_NOT_CAPTURED(4162)	Failed to capture the position
AXT_RT_ERROR_NOT_SAME_IC(4163)	Not in the same chip
AXT_RT_ERROR_NOT_GEARMODE(4164)	Failed to change to the gear mode
AXT_ERROR_CONTI_INVALID_AXIS_NO(4165)	Invalid axis during continuous interpolation axix mapping
AXT_ERROR_CONTI_INVALID_MAP_NO(4166)	Invalid mapping number during continuous interpolation mapping
AXT_ERROR_CONTI_EMPTY_MAP_NO(4167)	Continuous interpolation mapping number is empty
AXT_RT_MOTION_ERROR_CACULATION(4168)	Error in calculation
AXT_ERROR_HELICAL_INVALID_AXIS_NO(4170)	Invalid axis for helical axis mapping
AXT_ERROR_HELICAL_INVALID_MAP_NO(4171)	Invalid mapping number for helical mapping
AXT_ERROR_HELICAL_EMPTY_MAP_NO(4172)	Helical mapping number is empty

AXT_ERROR_SPLINE_INVALID_AXIS_NO (4180)	Invalid axis for spline axis mapping
AXT_ERROR_SPLINE_INVALID_MAP_NO(4181)	Invalid mapping number for spline mapping
AXT_ERROR_SPLINE_EMPTY_MAP_NO (4182)	Spline Mapping number is empty
AXT_ERROR_SPLINE_NUM_ERROR(4183)	Spline point value is inapplicable
AXT_RT_MOTION_INTERPOL_VALUE(4184)	Invalid input value for interpolation
AXT_RT_ERROR_NOT_CONTIBEGIN(4185)	CONTIBEGIN function is not called in continuous interpolation
AXT_RT_ERROR_NOT_CONTIEND(4186)	CONTIEND function is not called in continuous interpolation
AXT_RT_MOTION_HOME_SEARCHING(4201)	Other function is called while in home search motion
AXT_RT_MOTION_HOME_ERROR_SEARCHING(4202)	Home search motion is forced to stop by the user or something from external
AXT_RT_MOTION_HOME_ERROR_START(4203)	Failed to start home search drive due to initialization problem
AXT_RT_MOTION_HOME_ERROR_GANTRY(4204)	Failed to execute Gantry enable during home search motion
AXT_RT_MOTION_POS_OUTOFCBOUND(4251)	Configured position is either above the maximum value or below the minimum value
AXT_RT_MOTION_PROFILE_INVALID(4252)	Invalid setting for velocity profile
AXT_RT_MOTION_VELOCITY_OUTOFCBOUND(4253)	Configured velocity is either above the maximum value or below the minimum value
AXT_RT_MOTION_MOVE_UNIT_IS_ZERO(4254)	Unit of the motion values is set as zero
AXT_RT_MOTION_SETTING_ERROR(4255)	Invalid setting for Velocity, Acceleration, Jerk, or Profile
AXT_RT_MOTION_IN_CONT_INTERPOL(4256)	Execution of motion start or restart function before the completion of current continuous interpolated motion
AXT_RT_MOTION_DISABLE_TRIGGER(4257)	Trigger output is disabled
AXT_RT_MOTION_INVALID_CONT_INDEX(4258)	Invalid setting for continuous interpolation index value
AXT_RT_MOTION_CONT_QUEUE_FULL(4259)	Continuous Interpolation queue of motion chip is full

The contents of this manual can be changed without notice. The name of company, institution, production, people and event used in the examples are not real data. None of actual company, institution, product, people, or events are intended to be connected, and it shall not be interpreted that way. It is user's responsibility to comply with the corresponding copyright law. Separately from the right in the copyright, any part of this manual shall not be copied in any form or means (electronic, mechanical, paper copy, disk copy, or other method), shall not be copied for other purpose, and shall not be stored, introduced, or transmitted without explicit written approval of Ajinextek Co., Ltd.

Ajinextek Co., Ltd reserves all rights including patents, trademark right, copyright, or other intellectual property associated with the contents of this manual. Except the right explicitly given to you from Ajinextek Co., Ltd. by the written end user license agreement, providing this manual does not allow you to use any of patents, copyright, or other intellectual properties.