MAXk
DLL API MANUAL

(Revision A)

Pro-Dex, Inc. – Oregon Micro Systems
15201 NW Greenbrier Pkwy, B-1
BEAVERTON, OR 97006
PHONE 503-629-8081
FAX 503-629-0688
support@pro-dex.com
http://www.pro-dexoms.com

COPYRIGHT NOTICE

TRADEMARKS

IBM, IBM PC, IBM PC/XT, IBM PC/AT, IBM PS/2, and IBM PC DOS are registered trademarks of International Business Machines Corporation. Windows and Win NT are registered trademarks of Microsoft Corporation.

DISCLAIMER

Pro-Dex, Inc., Oregon Micro Systems makes no representations or warranties regarding the contents of this document. We reserve the right to revise this document, or make changes to the specifications of the product described within it at any time without notice and without obligation to notify any person of such revision or change.

REVISION NOTES

Revision	Date	Notes
A	17-Nov-2011	- Initial creation of the document

Table of Contents

▪ **Pro-Dex, Inc. – Oregon Micro Systems ▪ http://www.pro-dexoms.com** ▪
▪ 15201 NW Greenbrier Pkwy ▪ B-1 ▪ Beaverton, OR 97006 USA ▪ Ph: 503-629-8081 ▪ Fax: 503-629-0688 ▪

Table of Contents

▪ **Pro-Dex, Inc. – Oregon Micro Systems ▪ http://www.pro-dexoms.com** ▪
▪ 15201 NW Greenbrier Pkwy ▪ B-1 ▪ Beaverton, OR 97006 USA ▪ Ph: 503-629-8081 ▪ Fax: 503-629-0688 ▪

Table of Contents

Table of Contents

Appendixes

1 OVERVIEW

1.01 Synopsis

The OmsMAXkMC.DLL provides a set of functions that offer single function call to some of the more popular motion control uses. Not every command in the MAXk's command structure has a corresponding API in the DLL but the functions SendString or SendAndGetString allow the programmer to make any type of sequence of commands possible. (See MAXk User Manual for the list of all available commands.)

OmsMAXkMC.dll has been tested to run on Windows NT 4.0, 2000 and XP operating systems and was built using Visual Studio 6 with the latest service pack (SP6).

1.02 Installation

Support for the MAXk is only for Microsoft NT based operating systems and Linux. No drivers exist for Windows 95, 98 and ME.

1.02.1 Windows NT / 2000 / XP

Copy the OmsMAXkMC.dll to the System directory. Normally this is under C:\WinNT\System or C:\Windows\System directory.

1.03 DLL Linking

Application of the OmsMAXkMC.dll is done via Implicit Linking or sometimes referred to as static load or load-time dynamic linking. The executable using the DLL links to an import library file (.lib) that defines all the items that the DLL exports.

It's important that the DLL is located somewhere within the system's path for Windows to find. The default location for the installation of the MAXk DLLs is C:\Windows\System32. Windows will search for the DLL in the following sequence. If no success then an error will display.

- Directory where the executable is running from
- Current directory
- Windows system directory
- Windows directory
- All other directories listed in the PATH environment variable

1.04 Project Settings

It is highly recommended that the development environment is updated with the latest Service Packs before starting on building the application. The recommend compiler and IDE is Microsoft Visual Studio 6. However the Dll and library will work from version 5 thru version 2005 of Visual Studio.

1.04.1 Microsoft Visual C++ ver. 6

This procedure works with all C, C++ and MFC projects.

1. Copy OmsMAXkMC.lib and OmsMAXkMC.h to the project folder.
2. In Visual C++, click on Project | Settings…
3. In the Project Settings dialog box, the right side, change Settings For to All Configurations and then click on the Link Tab
4. Select General from Category and add the library OmsMAXkMC.lib to object/library modules
5. Click OK
6. Finished

1.04.2 Microsoft Visual Basic ver. 6

1. Copy OmsMAXkMC.bas to the working Visual Basic project folder
2. Click on Project | Add Module
3. Click on the Existing Tab and locate OmsMAXkMC.bas module
4. Click Open
5. Finished

2 GENERAL FUNCTIONS

2.01 GetOmsHandle

Get a device handle to a MAXk motion controller.

C / C++
```
HANDLE GetOmsHandle(LPSTR DeviceName);
```

VB
```
Declare Function GetOmsHandle Lib "OmsMAXkMC" (ByVal DeviceName As
String) As Long
```

Return Value

Handle to the controller or NULL if unsuccessful.

Parameter

DeviceName	[in] String containing the name of the controller

Equivalent Command String

None

Remarks

NULL can be returned if the device name is invalid or the device driver has not been installed. The string passed in must be NULL terminated.

Example

C / C++

```
// Define a global device handle variable
HANDLE hDevice;
...
hDevice = GetOmsHandle("OMSMAXk1");
...
// Close the device handle
CloseOmsHandle(hDevice);
```

VB

```
Dim hDevice As Long

hDevice = GetOmsHandle (“OMSMAXk1”)
If( hDevice = 0 ) Then
  MsgBox(“Error getting Handle to controller”)
End If

‘ do operations…

CloseOmsHandle (hDevice)
```

2.02 CloseOmsHandle

Close a handle to a MAXk motion controller.

C / C++
```
void CloseOmsHandle(HANDLE hDevice);
```

VB
```
Declare Sub CloseOmsHandle Lib "OmsMAXkMC" (ByVal hDevice As Long)
```

Return Value

None

Parameter

hDevice	[in] Handle to the controller

Equivalent Command String

None

Remarks

None

Example

See GetOmsHandle

2.03 ResetOmsController

Send software reset command to the controller.

C / C++
```
long ResetOmsController(HANDLE hDevice);
```

VB
```
Declare Function ResetOmsController Lib "OmsMAXkMC" (ByVal hDevice As
Long) As Long
```

Return Values

SUCCESS	Command was sent
COMMAND_TIME_OUT	Command timed out

Parameter

hDevice	[in] Handle to the controller

Equivalent Command String

RS

Remarks

It's recommended that reset be used as a last resort. All parameters will be set to their default values stored in flash and hardware is set to its power up state. In risk of an application crash, any communication to the controller should not happen for 3 to 6 seconds.

Example

C / C++

```
long lRetVal;
...
lRetVal = ResetOmsController(hDevice);
if(lRetVal == COMMAND_TIME_OUT)
 // The controller did not accept the reset command...
```

VB

```
Dim lRetVal As Long;
lRetVal = ResetOmsController(hDevice)
If(lRetVal = COMMAND_TIME_OUT) Then
 ' The controller did not accept the reset command
```

2.04 SendOmsQueryCommand

Same as SendAndGetString

2.05 SendAndGetString

Sends a query command string to the controller and receives a response in return.

C / C++

```
long SendAndGetString(HANDLE hDevice, LPSTR pCommand, LPSTR pResponse);
```

VB

```
Declare Function SendAndGetString Lib "OmsMAXkMC" (ByVal hDevice As
Long, ByVal pCommand As String, ByVal pResponse As String) As Long
```

Return Values

SUCCESS	Query command was successful
RESPONSE_TIME_OUT	Timeout waiting for response

Parameters

hDevice	[in] Handle to the controller
pCommand	[in] Pointer to NULL terminated string
pResponse	[out] Pointer to receive string buffer

Equivalent Command String

None

Remarks

Only use this call for commands that expect a response back from the controller. Non query commands will result with a return of RESPONSE_TIME_OUT.

Care should be taken that the response buffer is large enough to contain the controller's response.

Example

C / C++

```
long lRetVal;
char strCmd[] = "axrp";
char strRsp[128];
...
lRetVal = SendAndGetString(OmsHandle, strCmd, strRsp);
if(lRetVal == RESPONSE_TIME_OUT)
  // The controller did not respond with in 100 milliseconds.
```

VB

```
Dim lRetVal As Long
' Allocate a variable length string for the command string
Dim strCmd As String
' Allocate a fixed length string to receive the controller response
Dim strRsp As String * 128
strCmd = "AXRP"
lRetVal = SendAndGetString(hDevice, strCmd, strRsp)
If (lRetVal = RESPONSE_TIME_OUT) Then
 ' Handle error
```

2.06 SendOmsTextCommand

Same as SendString

2.07 SendString

Send a command string to the controller.

C / C++
```
long SendString(HANDLE hDevice, LPSTR pCommand);
```

VB
```
Declare Function SendString Lib "OmsMAXkMC" (ByVal hDevice As Long,
ByVal pCommand As String) As Long
```

Return Values

SUCCESS	Command was sent
COMMAND_TIME_OUT	Command timed out

Parameters

hDevice	[in] Handle to the controller
pCommand	[in] char pointer to NULL terminated string

Equivalent Command String

None

Remarks

Commands can be sent via individual calls or together in one call.

Example

C / C++

```
long lRetVal;
char strCmd[] = " AXVL5000; AYVL5000; AZVL5000; ATVL5000;";

lRetVal = SendString(hDevice, strCmd);
if(lRetVal == COMMAND_TIME_OUT)
// Controller did not accept the command string with in 100 milliseconds.
```

VB

```
Dim lRetVal As Long
' Allocate a variable length string for the command string
Dim strCmd As String

strCmd = “AXVL5000; AYVL5000; AZVL5000; ATVL5000;”
lRetVal = SendString ( hDevice, strCmd)

If( lRetVal = COMMAND_TIME_OUT) Then
  ‘ error sending the command
End If
```

2.08 GetCmdBufferFree

Get the number of free bytes in the controller's command buffer.

C / C++
```
long GetCmdBufferFree(HANDLE hDevice);
```

VB
```
Declare Function GetCmdBufferFree Lib "OmsMAXkMC" (ByVal hDevice As
Long) As Long
```

Return Values

The number of free bytes in the controller's command buffer.

Parameters

hDevice	[in] Handle to the controller

Equivalent Command String

None

Remarks

None

Example

C / C++

```
lCmdBuffSize = GetCmdBufferFree (hDevice);

if( lRetVal == COMMAND_TIME_OUT )
  //  Error
```

VB

```
lCmdBuffSize = GetCmdBufferFree (hDevice)

If( lRetVal = COMMAND_TIME_OUT ) Then
  ' Error
End If
```

2.09 GetOmsControllerDescription

Get the ID of the MAXk controller.

C / C++
```
long GetOmsControllerDescription (HANDLE hDevice, LPSTR Description);
```

VB
```
Declare Function GetOmsControllerDescription Lib "OmsMAXkMC" (ByVal
hDevice As Long, ByVal Description As String) As Long
```

Return Values

SUCCESS	Command was sent
COMMAND_TIME_OUT	Command timed out

Parameters

hDevice	[in] Handle to the controller
Description	[out] char array pointer for ID string

Equivalent Command String

WY

Remarks

The *Description* buffer should be a minimum of 50 characters in length or larger.

Example

See GetOmsDllVersion

2.10 GetOmsDriverVersion

Return the driver version string.

C / C++
```
void GetOmsDriverVersion(HANDLE hDevice, LPSTR DrvVersion);
```

VB
```
Declare Sub GetOmsDriverVersion Lib "OmsMAXkMC" (ByVal hDevice As Long,
ByVal DrvVersion As String)
```

Return Values

None

Parameters

hDevice	[in] Handle to the controller
DrvVersion	[out] char array pointer for driver info. string

Equivalent Command String

None

Remarks

It is recommended that the *DrvVersion* be of size 30 characters in length or larger.

Example

See GetOmsDllVersion

2.11 GetOmsDllVersion

Returns the DLL version string.

C / C++
```
void GetOmsDllVersion(HANDLE hDevice, LPSTR pDllVersion, long lLen);
```

VB
```
Declare Sub GetOmsDllVersion Lib "OmsMAXkMC" (ByVal hDevice As Long,
ByVal pDllVersion As String, ByVal lLen As Long) As Long
```

Return Value

None

Parameters

hDevice	[in] Handle to the controller
pDllVersion	[out] char array pointer for dll info. string
lLen	[in] size of *pDllVersion* char array buffer

Equivalent Command String

None

Remarks

It is recommended that char array should be 30 characters or larger. Any size smaller the response will be cut off to the length of the buffer (*lLen*).

Example

C / C++

```
lRetVal = GetOmsDriverVersion ( hDevice, strDriverVer, 30);

if( lRetVal == COMMAND_TIME_OUT )
  //  Error: command was not accepted by the controller

GetOmsDllVersion ( hDevice, strDllVer, 30);
lRetVal = GetOmsControllerDescription (hDevice, strID, 50);

if( lRetVal == COMMAND_TIME_OUT )
  //  Error: command was not accepted by the controller
```

VB

```
GetOmsDriverVersion (hDevice, strDriverVer, 30)
If( lRetVal = COMMAND_TIME_OUT ) Then
  ‘ Error: command was not accepted by the controller
End If

GetOmsDllVersion ( hDevice, strDllVer, 30)
lRetVal = GetOmsControllerDescription (hDevice, sID, 50)

If( lRetVal = COMMAND_TIME_OUT ) Then
  ‘ Error: command was not accepted by the controller
End If
```

3 MOVE SINGLE AXIS MOTION FUNCTIONS

3.01 MoveOmsAxisAbs

Move an axis to an absolute position using it's preset velocity and acceleration.

C / C++
```
long MoveOmsAxisAbs(HANDLE hDevice, long AxisSelection, long
lPosition);
```

VB
```
Declare Function MoveOmsAxisAbs Lib "OmsMAXkMC" (ByVal hDevice As Long,
ByVal AxisSelection As Long, ByVal lPosition As Long) As Long
```

Return Values

SUCCESS	Request was sent
COMMAND_TIME_OUT	Command timed out
INVALID_AXIS_SELECTION	Invalid axis value
INVALID_PARAMETER	Invalid position value

Parameters

hDevice	[in] Handle to the controller
AxisSelection	[in] Selected axis
lPosition	[in] Destination position of the axis

Equivalent ASCII Command String

`A*MA#;GDID;`

`*` - axis character corresponding to the value passed in *AxisSelection*.

`#` - value passed in via *lPosition*.

Remarks

The axis done flag will be cleared when the move begins and will be set when the move is complete. This function returns control to the application as soon as the command has been accepted by the controller. It does not wait for the move to complete. The GetOmsAxisDoneFlag function can be used to determine when the move is complete.

Example

C / C++

```
//  move X axis to absolute position 5000, no wait
lPosition = 5000;
lRetVal = MoveOmsAxisAbs ( hDevice, OMS_X_AXIS, lPosition);

if( lRetVal != SUCCESS )
  //  Error as occurred
```

VB

```
‘ move X axis to absolute position 5000, no wait
lPosition = 5000
lRetVal = MoveOmsAxisAbs ( hDevice, OMS_X_AXIS, lPosition)

If ( lRetVal <> SUCCESS ) Then
  ‘ Error has occurred
End If
```

3.02 MoveOmsAxisAbsWait

Move an axis to an absolute position using it's preset velocity and acceleration.

C / C++

```
long MoveOmsAxisAbsWait (HANDLE hDevice, long AxisSelection, long
lPosition, long lTimeLimit);
```

VB

```
Declare Function MoveOmsAxisAbsWait Lib "OmsMAXkMC" (ByVal hDevice As
Long, ByVal AxisSelection As Long, ByVal lPosition As Long, ByVal
lTimeLimit As Long) As Long
```

Return Values

SUCCESS	Request was sent
COMMAND_TIME_OUT	Command timed out
INVALID_AXIS_SELECTION	Invalid axis value
INVALID_PARAMETER	Invalid position value
MOVE_TIME_OUT	Timeout occurred before the move completed

Parameters

hDevice	[in] Handle to the controller
AxisSelection	[in] Selected axis
lPosition	[in] Destination position of the axis
lTimeLimit	[in] Desired timeout for axis move

Equivalent Command String

A*MA#;GDID;

***** - axis character corresponding to the value passed in *AxisSelection*.

- value passed in via *lPosition*.

Remarks

Control will be returned to the application when the axis reaches it's target position or the move timeout has expired. The axis done flag will be cleared when the move begins and will be set when the move completes.

The function queries the done flag until it is set or the specified time has elapsed.

Example

C / C++

```
//  move X axis to absolute position 9000
lPosition = 9000;
lTimeout  = 5000;  // timeout in 5 seconds if move does not complete

lRetVal = MoveOmsAxisAbsWait( hDevice, OMS_X_AXIS, lPosition, lTimeout);

if( lRetVal != SUCCESS )
  //  Error as occurred
```

VB

```
‘ move X axis to absolute position 9000
lPosition = 9000
lTimeout = 5000  ‘ timeout in 5 seconds if move does not complete

lRetVal = MoveOmsAxisAbs ( hDevice, OMS_X_AXIS, lPosition)

If ( lRetVal <> SUCCESS ) Then
  ‘ Error has occurred
End If
```

3.03 MoveOmsAxisRel

Move an axis a relative position using it's preset velocity and acceleration.

C / C++
```
long MoveOmsAxisRel(HANDLE hDevice, long AxisSelection, long
lPosition);
```

VB
```
Declare Function MoveOmsAxisRel Lib "OmsMAXkMC" (ByVal hDevice As Long,
ByVal AxisSelection As Long, ByVal lPosition As Long) As Long
```

Return Values

SUCCESS	Request was sent
COMMAND_TIME_OUT	Command timed out
INVALID_AXIS_SELECTION	Invalid axis value
INVALID_PARAMETER	Invalid position value

Parameters

hDevice	[in] Handle to the controller
AxisSelection	[in] Selected axis
lPosition	[in] Destination position of the axis

Equivalent Command String

A*MR#;GDID;

***** - axis character corresponding to the value passed in *AxisSelection*.

- value passed in via *lPosition*.

Remarks

Similar to MoveOmsAxisAbs but moves the axis to relative position. The axis done flag will be cleared when the move begins and will be set when the move is complete. This function returns control to the application as soon as the command has been accepted by the controller. It does not wait for the move to complete.

The GetOmsAxisDoneFlag function can be used to determine when the move is complete.

Example

See MoveOmsAxisAbs for a related example

3.04 MoveOmsAxisRelWait

Move an axis a relative position using it's preset velocity and acceleration.

C / C++

```
long MoveOmsAxisRelWait (HANDLE hDevice, long AxisSelection, long
lPosition, long lTimeLimit);
```

VB

```
Declare Function MoveOmsAxisRelWait Lib "OmsMAXkMC" (ByVal hDevice As
Long, ByVal AxisSelection As Long, ByVal lPosition As Long, ByVal
lTimeLimit As Long) As Long
```

Return Values

SUCCESS	Request was sent
COMMAND_TIME_OUT	Command timed out
INVALID_AXIS_SELECTION	Invalid axis value
INVALID_PARAMETER	Invalid position value
MOVE_TIME_OUT	Timeout occurred before the move completed

Parameters

hDevice	[in] Handle to the controller
AxisSelection	[in] Selected axis
lPosition	[in] Destination position of the axis
lTimeLimit	[in] Desired timeout for axis move

Equivalent Command String

A*MR#;GDID;

***** - axis character corresponding to the value passed in *AxisSelection*.

- value passed in via *lPosition*.

Remarks

Similar to MoveOmsAxisAbsWait but moves the axis to relative position. Control will be returned to the application when the axis reaches it's target position or the move timeout has expired. The axis done flag will be cleared when the move begins and will be set when the move completes.

The function queries the done flag until it is set or the specified time has elapsed.

Example

See MoveOmsAxisAbsWait for a related example

3.05 MoveOmsAxisIndefinite

Move an axis at a specified velocity indefinitely until it is commanded to stop.

C / C++
```
long MoveOmsAxisIndefinite(HANDLE hDevice, long AxisSelection, long Velocity);
```

VB
```
Declare Function MoveOmsAxisIndefinite Lib "OmsMAXkMC" (ByVal hDevice As Long, ByVal AxisSelection As Long, ByVal Velocity As Long) As Long
```

Return Values

SUCCESS	Request was sent
COMMAND_TIME_OUT	Command timed out
INVALID_AXIS_SELECTION	Invalid axis value
INVALID_PARAMETER	Invalid velocity value

Parameters

hDevice	[in] Handle to the controller
AxisSelection	[in] Selected axis
Velocity	[in] Velocity to run the axis at

Equivalent Command String

`A*JG#;`

`*` - axis character corresponding to the value passed in *AxisSelection*.

`#` - value passed in via *Velocity*.

Remarks

Use any of the STOP MOTION FUNCTIONS or SendString("ST") to stop the current axis.

Example

C / C++

```
//  move Y axis at velocity of 5000
lAxis = OMS_Y_AXIS;
lVelocity = 5000;

lRetVal = MoveOmsAxisIndefinite ( hDevice, lAxis, lVelocity);

if ( lRetVal != SUCCESS )
  //  Error has occurred
```

VB

```
‘ move X axis at velocity of 5000
lAxis = OMS_X_AXIS
lVelocity = 5000

lRetVal = MoveOmsAxisIndefinite ( hDevice, lAxis, lVelocity)

If ( lRetVal <> SUCCESS ) Then
  ‘ Error has occurred
End If
```

3.06 MoveOmsAxisFractional

Move an axis at a specified fractional velocity rate until it is commanded to stop.

C / C++
```
long MoveOmsAxisFractional(HANDLE hDevice, long AxisSelection, double Velocity);
```

VB
```
Declare Function MoveOmsAxisFractional Lib "OmsMAXkMC" (ByVal hDevice As Long, ByVal AxisSelection As Long, ByVal Velocity As Double) As Long
```

Return Values

SUCCESS	Request was sent
COMMAND_TIME_OUT	Command timed out
INVALID_AXIS_SELECTION	Invalid axis value
INVALID_PARAMETER	Invalid velocity value

Parameters

hDevice	[in] Handle to the controller
AxisSelection	[in] Selected axis
Velocity	[in] Fractional velocity to run the axis at

Equivalent Command String

A*JF#;

***** - axis character corresponding to the value passed in *AxisSelection*.

- value passed in via *Velocity*.

Remarks

This function allows axes to move at fractional step rates. The velocity is limited to the range of -2000.0 to 2000.0 steps per second.

Use any of the STOP MOTION FUNCTIONS or SendString("ST") to stop the current axis.

Example

C / C++

```
//  move Y axis at a fractional velocity of 50.75
lAxis = OMS_Y_AXIS;
dVelocity = 50.75;

lRetVal = MoveOmsAxisFractional ( hDevice, lAxis, dVelocity);

if ( lRetVal != SUCCESS )
  //  Error has occurred
```

VB

```
‘ move Y axis at a fractional velocity of 50.75
lAxis = OMS_Y_AXIS
dVelocity = 50.75

lRetVal = MoveOmsAxisFractional ( hDevice, lAxis, dVelocity)

If ( lRetVal <> SUCCESS ) Then
  ‘ Error has occurred
End If
```

3.07 MoveOmsAxisOneStep

Move an axis one step in the specified direction.

C / C++
```
long MoveOmsAxisOneStep(HANDLE hDevice, long AxisSelection, long Direction);
```

VB
```
Declare Function MoveOmsAxisOneStep Lib "OmsMAXkMC" (ByVal hDevice As Long, ByVal AxisSelection As Long, ByVal Direction As Long) As Long
```

Return Values

SUCCESS	Request was sent
COMMAND_TIME_OUT	Command timed out
INVALID_AXIS_SELECTION	Invalid axis value
INVALID_PARAMETER	Invalid direction

Parameters

hDevice	[in] Handle to the controller
AxisSelection	[in] Selected axis
Direction	[in] Direction value: POSITIVE or NEGATIVE

Equivalent Command String

[move positive] A***MP**MO#; or [move negative] A***MM**MO#;

***** - axis character corresponding to the value passed in *AxisSelection*.

MP or MM – selected based on the value of *Direction.*

Remarks

None

Example

C / C++

```
lRetVal = MoveOmsAxisOneStep ( hDevice, OMS_X_AXIS, POSITIVE);

if ( lRetVal != SUCCESS )
  //  Error has occurred
```

VB

```
lRetVal = MoveOmsAxisOneStep ( hDevice, OMS_X_AXIS, NEGATIVE)

If ( lRetVal <> SUCCESS ) Then
  ‘ Error has occurred
End If
```

4 MOVE MULTIPLE AXIS FUNCTIONS

4.01 MoveOmsIndependentAbs

Move a set of axes to an absolute positions using each axis own predefined velocity and acceleration values.

C / C++
```
long MoveOmsIndependentAbs (HANDLE hDevice, long AxisSelections,
PAXES_DATA lPosition);
```

VB
```
Declare Function MoveOmsIndependentAbs Lib "OmsMAXkMC" (ByVal hDevice
As Long, ByVal AxisSelection As Long, ByRef lPosition As AXES_DATA) As
Long
```

Return Values

SUCCESS	Request was sent
COMMAND_TIME_OUT	Command timed out
INVALID_AXIS_SELECTION	Invalid axis value
INVALID_PARAMETER	Invalid position value

Parameters

hDevice	[in] Handle to the controller
AxisSelections	[in] Axis selection bit mask
lPosition	[in] Pointer to AXES_DATA structure

Equivalent Command String

AAMA#,#,…#; GDID;
- values passed in via *lPosition* structure.

Remarks

All selected axes will start their moves at the same time. Each axis will move to its destination, independently, based on its own velocity and acceleration parameters.

The move is executed in all axis mode so all controller axes will suspend further command execution until the move is complete. All axis done flags will be cleared when the move begins and set when the move is complete. The function will return control to the application as soon as the command has been accepted by the controller. It does not wait until the move is complete.

Use GetAllOmsDoneFlags to determine when the moves are complete.

Example

C / C++

```
//  Move X and T axes 4000, move Y 8000 relative
lAxes = OMS_X_AXIS + OMS_Y_AXIS + OMS_T_AXIS;
axesPosition.X = 4000;
axesPosition.Y = 8000;
axesPosition.T = 4000;

lRetVal = MoveOmsIndependentAbs ( hDevice, lAxes, &axesPosition);

if ( lRetVal != SUCCESS )  //  Error has occurred
  //  Handle any errors
```

VB

```
‘ Move X and T axes 4000, move Y 8000 relative
lAxes = OMS_X_AXIS + OMS_Y_AXIS + OMS_T_AXIS
axesPosition.X = 4000
axesPosition.Y = 8000
axesPosition.T = 4000

lRetVal = MoveOmsIndependentAbs ( hDevice, lAxes, axesPosition)

If ( lRetVal <> SUCCESS ) Then  ‘ Error has occurred
  ‘ Handle any errors
End If
```

4.02 MoveOmsIndependentAbsWait

Move a set of axes to an absolute and wait (block) until the moves are complete or timeout has expired.

C / C++
```
long MoveOmsIndependentAbsWait(HANDLE hDevice, long AxisSelections,
AXES_DATA lPosition, long lTimeLimit);
```

VB
```
Declare Function MoveOmsIndependentAbsWait Lib "OmsMAXkMC" (ByVal
hDevice As Long, ByVal AxisSelection As Long, ByRef lPosition As
AXES_DATA, ByVal lTimeLimit As Long) As Long
```

Return Values

SUCCESS	Request was sent
COMMAND_TIME_OUT	Command timed out
INVALID_AXIS_SELECTION	Invalid axis value
INVALID_PARAMETER	Invalid position value
MOVE_TIME_OUT	Timeout limit reached before the moves completed

Parameters

hDevice	[in] Handle to the controller
AxisSelections	[in] Axis selection bit mask
lPosition	[in] Pointer to AXES_DATA structure
lTimeLimit	[in] Motion timeout limit in milliseconds.

Equivalent Command String

AAMA#,#,…#; GDID;
- values passed in via *lPosition* structure.

Remarks

The selected axes will start their moves at the same time. Each axis will move to its destination, independently, based on its own velocity and acceleration parameters. The move is executed in all axis mode so all controller axes will suspend further command execution until the move is complete. All axes done flags will be cleared when the move begins and set when the move is complete.

Example

C / C++

```
//  Move X and T axes 4000, move Y 8000 relative with 5 second wait
lAxes = OMS_X_AXIS + OMS_Y_AXIS + OMS_T_AXIS;
axesPosition.X = 4000;
axesPosition.Y = 8000;
axesPosition.T = 4000;

lRetVal = MoveOmsIndependentAbsWait ( hDevice, lAxes, &axesPosition, 5000);

if ( lRetVal != SUCCESS )  //  Error has occurred
  //  Handle any errors
```

VB

```
‘ Move Y and Z axes -4000, move T 4000 absolute with 5 sec. wait;
lAxes = OMS_X_AXIS + OMS_Y_AXIS + OMS_T_AXIS
axesPosition.Y = -4000
axesPosition.Z = -4000
axesPosition.T = 4000

lRetVal = MoveOmsIndependentAbsWait ( hDevice, lAxes, axesPosition, 5000)

If ( lRetVal <> SUCCESS ) Then  ‘ Error has occurred
  ‘ Handle any errors
End If
```

4.03 MoveOmsIndependentRel

Move a set of axes relative positions using each axis own predefined velocity and acceleration values.

C / C++
```
long MoveOmsIndependentRel (HANDLE hDevice, long AxisSelections,
PAXES_DATA lPosition);
```

VB
```
Declare Function MoveOmsIndependentRel Lib "OmsMAXkMC" (ByVal hDevice
As Long, ByVal AxisSelection As Long, ByRef lPosition As AXES_DATA) As
Long
```

Return Values

SUCCESS	Request was sent
COMMAND_TIME_OUT	Command timed out
INVALID_AXIS_SELECTION	Invalid axis value
INVALID_PARAMETER	Invalid position value

Parameters

hDevice	[in] Handle to the controller
AxisSelections	[in] Axis selection bit mask
lPosition	[in] Pointer to AXES_DATA structure

Equivalent Command String

AAMR#,#,…#; GDID;
#- values passed in via *lPosition* structure.

Remarks

All selected axes will start their moves at the same time. Each axis will move to its destination, independently, based on its own velocity and acceleration parameters. The moves are executed in all axis mode so all controller axes will suspend further command execution until the move is complete. All axis done flags will be cleared when the move begins and set when the move is complete. Non-blocking call, it does not wait for the moves to complete. Use GetAllOmsDoneFlags to determine when the moves are complete.

Example

See MoveOmsIndependentAbs for a similar example.

4.04 MoveOmsIndependentRelWait

Move a set of axes relative positions and wait (block) until the moves are complete or timeout has expired.

C / C++
```
long MoveOmsIndependentRelWait(HANDLE hDevice, long AxisSelections,
AXES_DATA Position, long lTimeLimit);
```

VB
```
Declare Function MoveOmsIndependentRelWait Lib "OmsMAXkMC" (ByVal
hDevice As Long, ByVal AxisSelection As Long, ByRef Position As
AXES_DATA, ByVal lTimeLimit As Long) As Long
```

Return Values

SUCCESS	Request was sent
COMMAND_TIME_OUT	Command timed out
INVALID_AXIS_SELECTION	Invalid axis value
INVALID_PARAMETER	Invalid position value
MOVE_TIME_OUT	Timeout limit reached before the moves completed

Parameters

hDevice	[in] Handle to the controller
AxisSelections	[in] Axis selection bit mask
Position	[in] Pointer to AXES_DATA structure
lTimeLimit	[in] Motion timeout limit in milliseconds.

Equivalent Command String

AAMR#,#,…#; GDID;
  # - values passed in via *lPosition* structure.

Remarks

The selected axes will start their moves at the same time. Each axis will move to its destination, independently, based on its own velocity and acceleration parameters. The move is executed in all axis mode so all controller axes will suspend further command execution until the move is complete. All axes done flags will be cleared when the move begins and set when the move is complete.

Example

See MoveOmsIndependentAbsWait for a similar example.

4.05 MoveOmsLinearAbs

Moves selected axes to an absolute position. Starting and stopping all selected axes at the same time.

C / C++
```
long MoveOmsLinearAbs (HANDLE hDevice, long AxisSelections, PAXES_DATA lPosition);
```

VB
```
Declare Function MoveOmsLinearAbs Lib "OmsMAXkMC" (ByVal hDevice As Long, ByVal AxisSelection As Long, ByRef lPosition As AXES_DATA) As Long
```

Return Values

SUCCESS	Request was sent
COMMAND_TIME_OUT	Command timed out
INVALID_AXIS_SELECTION	Invalid axis value
INVALID_PARAMETER	Invalid position value

Parameters

hDevice	[in] Handle to the controller
AxisSelections	[in] Axis selection bit mask
lPosition	[in] Pointer to AXES_DATA structure

Equivalent Command String

AAMT#,#,…#; GDID;
#- values passed in via *lPosition* structure.

Remarks

Linear interpolation is used to adjust the velocity and acceleration of each axis so that they start and end their motion at the same time. The move is executed in all axis mode so all controller axes will suspend further command execution until the move is complete. All axis done flags will be cleared when the move begins and set when the move is complete.

Use GetAllOmsDoneFlagsto determine when the moves are complete.

Example

C / C++

```
//  Move X and T axes 4000, move Y 8000 relative
lAxes = OMS_X_AXIS + OMS_Y_AXIS + OMS_T_AXIS;
axesPosition.X = 4000;
axesPosition.Y = 8000;
axesPosition.T = 4000;

lRetVal = MoveOmsLinearAbs ( hDevice, lAxes, &axesPosition);

if ( lRetVal != SUCCESS )  //  Error has occurred
  //  Handle any errors
```

VB

```
‘ Move X and T axes 4000, move Y 8000 relative
lAxes = OMS_X_AXIS + OMS_Y_AXIS + OMS_T_AXIS
axesPosition.X = 4000
axesPosition.Y = 8000
axesPosition.T = 4000

lRetVal = MoveOmsLinearAbs ( hDevice, lAxes, axesPosition)

If ( lRetVal <> SUCCESS ) Then  ‘ Error has occurred
  ‘ Handle any errors
End If
```

4.06 MoveOmsLinearAbsWait

Move a set of axes to an absolute and wait (block) until the moves are complete or timeout has expired. Starting and stopping all selected axes at the same time.

C / C++
```
long MoveOmsLinearAbsWait(HANDLE hDevice, long AxisSelections,
AXES_DATA lPosition, long lTimeLimit);
```

VB
```
Declare Function MoveOmsLinearAbsWait Lib "OmsMAXkMC" (ByVal hDevice As
Long, ByVal AxisSelection As Long, ByRef lPosition As AXES_DATA, ByVal
lTimeLimit As Long) As Long
```

Return Values

SUCCESS	Request was sent
COMMAND_TIME_OUT	Command timed out
INVALID_AXIS_SELECTION	Invalid axis value
INVALID_PARAMETER	Invalid position value
MOVE_TIME_OUT	Timeout limit reached before the moves completed

Parameters

hDevice	[in] Handle to the controller
AxisSelections	[in] Axis selection bit mask
lPosition	[in] Pointer to AXES_DATA structure
lTimeLimit	[in] Motion timeout limit in milliseconds.

Equivalent Command String

AAMT#,#,…#; GDID;
- values passed in via *lPosition* structure.

Remarks

Linear interpolation is used to adjust the velocity and acceleration of each axis so that they start and end their motion at the same time. The move is executed in all axis mode so all controller axes will suspend further command execution until the move is complete. All axis done flags will be cleared when the move begins and set when the move is complete.

Example

C / C++

```
//  Move X and T axes 4000, move Y 8000 relative with 5 second wait
lAxes = OMS_X_AXIS + OMS_Y_AXIS + OMS_T_AXIS;
axesPosition.X = 4000;
axesPosition.Y = 8000;
axesPosition.T = 4000;

lRetVal = MoveOmsLinearAbsWait ( hDevice, lAxes, &axesPosition, 5000);

if ( lRetVal != SUCCESS )  //  Error has occurred
  //  Handle any errors
```

VB

```
‘ Move Y and Z axes -4000, move T 4000 absolute with 5 sec. wait;
lAxes = OMS_X_AXIS + OMS_Y_AXIS + OMS_T_AXIS
axesPosition.Y = -4000
axesPosition.Z = -4000
axesPosition.T = 4000

lRetVal = MoveOmsLinearAbsWait ( hDevice, lAxes, axesPosition, 5000)

If ( lRetVal <> SUCCESS ) Then  ‘ Error has occurred
  ‘ Handle any errors
End If
```

4.07 MoveOmsLinearRel

Move a set of axes relative positions. Starting and stopping all selected axes at the same time.

C / C++
```
long MoveOmsLinearRel (HANDLE hDevice, long AxisSelections, PAXES_DATA
lPosition);
```

VB
```
Declare Function MoveOmsLinearRel Lib "OmsMAXkMC" (ByVal hDevice As
Long, ByVal AxisSelection As Long, ByRef lPosition As AXES_DATA) As
Long
```

Return Values

SUCCESS	Request was sent
COMMAND_TIME_OUT	Command timed out
INVALID_AXIS_SELECTION	Invalid axis value
INVALID_PARAMETER	Invalid position value

Parameters

hDevice	[in] Handle to the controller
AxisSelections	[in] Axis selection bit mask
lPosition	[in] Pointer to AXES_DATA structure

Equivalent Command String

AAML#,#,…#; GDID;
#- values passed in via *lPosition* structure.

Remarks

Linear interpolation is used to adjust the velocity and acceleration of each axis so that they start and end their motion at the same time. The move is executed in all axis mode so all controller axes will suspend further command execution until the move is complete. All axis done flags will be cleared when the move begins and set when the move is complete.

Use GetAllOmsDoneFlags to determine when the moves are complete.

Example

See MoveOmsLinearAbs for a comparable example.

4.08 MoveOmsLinearRelWait

Move a set of axes relative positions and wait (block) until the moves are complete or timeout has expired. Starting and stopping all selected axes at the same time

C / C++
```
long MoveOmsLinearRelWait(HANDLE hDevice, long AxisSelections,
AXES_DATA Position, long lTimeLimit);
```

VB
```
Declare Function MoveOmsLinearRelWait Lib "OmsMAXkMC" (ByVal hDevice As
Long, ByVal AxisSelection As Long, ByRef Position As AXES_DATA, ByVal
lTimeLimit As Long) As Long
```

Return Values

SUCCESS	Request was sent
COMMAND_TIME_OUT	Command timed out
INVALID_AXIS_SELECTION	Invalid axis value
INVALID_PARAMETER	Invalid position value
MOVE_TIME_OUT	Timeout limit reached before the moves completed

Parameters

hDevice	[in] Handle to the controller
AxisSelections	[in] Axis selection bit mask
lPosition	[in] Pointer to AXES_DATA structure
lTimeLimit	[in] Motion timeout limit in milliseconds.

Equivalent Command String

AAML#,#,…#; GDID;

- values passed in via *lPosition* structure.

Remarks

Linear interpolation is used to adjust the velocity and acceleration of each axis so that they start and end their motion at the same time. The move is executed in all axis mode so all controller axes will suspend further command execution until the move is complete. All axis done flags will be cleared when the move begins and set when the move is complete.

Example

See MoveOmsLinearAbsWait for a comparable example.

4.09 MoveOmsIndependentAbsMt

Move a set of axes to an absolute positions using each axis own predefined velocity and acceleration values.

C / C++
```
long MoveOmsIndependentAbsMt (HANDLE hDevice, long AxisSelections,
PAXES_DATA lPosition);
```

VB
```
Declare Function MoveOmsIndependentAbsMt Lib "OmsMAXkMC" (ByVal hDevice
As Long, ByVal AxisSelection As Long, ByRef lPosition As AXES_DATA) As
Long
```

Return Values

SUCCESS	Request was sent
COMMAND_TIME_OUT	Command timed out
INVALID_AXIS_SELECTION	Invalid axis value
INVALID_PARAMETER	Invalid position value

Parameters

hDevice	[in] Handle to the controller
AxisSelections	[in] Axis selection bit mask
lPosition	[in] Pointer to AXES_DATA structure

Equivalent Command String

AMMA#,#,…#; GDID;
#- values passed in via *lPosition* structure.

Remarks

The selected axes will start their moves at the same time. Each axis will move to its destination, independently, based on its own velocity and acceleration parameters. The move is executed in multitask mode so that any axes, not involved in the move, will continue to execute commands in their queues. The selected axes done flags will be cleared at the start of the move and set when the move is complete.

Use GetAllOmsDoneFlags to determine when the moves are complete.

Example

C / C++

```
//  Move X 4000 and Z 8000 absolute position
lAxes = OMS_X_AXIS + OMS_Z_AXIS
axesPosition.X = 4000;
axesPosition.Z = 8000;

lRetVal = MoveOmsIndependentAbsMt ( hDevice, lAxes, &axesPosition);

if ( lRetVal != SUCCESS )  //  Error has occurred
  //  Handle any errors
```

VB

```
‘ Move X and T axes 4000, move Y 8000 relative
lAxes = OMS_X_AXIS + OMS_Y_AXIS + OMS_T_AXIS
axesPosition.X = 4000
axesPosition.Y = 8000
axesPosition.T = 4000

lRetVal = MoveOmsIndependentAbsMt ( hDevice, lAxes, axesPosition)

If ( lRetVal <> SUCCESS ) Then  ‘ Error has occurred
  ‘ Handle any errors
End If
```

4.10 MoveOmsIndependentAbsMtWait

Move a set of axes to an absolute and wait (block) until the moves are complete or timeout has expired.

C / C++
```
long MoveOmsIndependentAbsMtWait(HANDLE hDevice, long AxisSelections,
AXES_DATA Position, long lTimeLimit);
```

VB
```
Declare Function MoveOmsIndependentAbsMtWait Lib "OmsMAXkMC" (ByVal
hDevice As Long, ByVal AxisSelection As Long, ByRef Position As
AXES_DATA, ByVal lTimeLimit As Long) As Long
```

Return Values

SUCCESS	Request was sent
COMMAND_TIME_OUT	Command timed out
INVALID_AXIS_SELECTION	Invalid axis value
INVALID_PARAMETER	Invalid position value
MOVE_TIME_OUT	Timeout limit reached before the moves completed

Parameters

hDevice	[in] Handle to the controller
AxisSelections	[in] Axis selection bit mask
lPosition	[in] Pointer to AXES_DATA structure
lTimeLimit	[in] Motion timeout limit in milliseconds.

Equivalent Command String

AMMA#,#,…#; GDID;
- values passed in via *lPosition* structure.

Remarks

The selected axes will start their moves at the same time. Each axis will move to its destination, independently, based on its own velocity and acceleration parameters. The move is executed in multitask mode so that any axes, not involved in the move, will continue to execute commands in their queues. The selected axes done flags will be cleared at the start of the move and set when the move is complete.

Example

C / C++

```
//  Move X and T axes 4000, move Y 8000 relative with 5 second wait
lAxes = OMS_X_AXIS + OMS_Y_AXIS + OMS_T_AXIS;
axesPosition.X = 4000;
axesPosition.Y = 8000;
axesPosition.T = 4000;

lRetVal = MoveOmsIndependentAbsMtWait ( hDevice, lAxes, &axesPosition,
                                        5000);

if ( lRetVal != SUCCESS )  //  Error has occurred
  //  Handle any errors
```

VB

```
‘ Move Y and Z axes -4000, move T 4000 absolute with 5 sec. wait;
lAxes = OMS_X_AXIS + OMS_Y_AXIS + OMS_T_AXIS
axesPosition.Y = -4000
axesPosition.Z = -4000
axesPosition.T = 4000

lRetVal = MoveOmsIndependentAbsMtWait ( hDevice, lAxes, axesPosition, 5000)

If ( lRetVal <> SUCCESS ) Then  ‘ Error has occurred
  ‘ Handle any errors
End If
```

4.11 MoveOmsIndependentRelMt

Move a set of axes relative positions using each axis own predefined velocity and acceleration values.

C / C++
```
long MoveOmsIndependentRelMt (HANDLE hDevice, long AxisSelections,
PAXES_DATA lPosition);
```

VB
```
Declare Function MoveOmsIndependentRelMt Lib "OmsMAXkMC" (ByVal hDevice
As Long, ByVal AxisSelection As Long, ByRef lPosition As AXES_DATA) As
Long
```

Return Values

SUCCESS	Request was sent
COMMAND_TIME_OUT	Command timed out
INVALID_AXIS_SELECTION	Invalid axis value
INVALID_PARAMETER	Invalid position value

Parameters

hDevice	[in] Handle to the controller
AxisSelections	[in] Axis selection bit mask
lPosition	[in] Pointer to AXES_DATA structure

Equivalent Command String

```
AMMR#,#,…#; GDID;
```
- values passed in via *lPosition* structure.

Remarks

The selected axes will start their moves at the same time. Each axis will move to its destination, independently, based on its own velocity and acceleration parameters. The move is executed in multitask mode so that any axes, not involved in the move, will continue to execute commands in their queues. The selected axes done flags will be cleared at the start of the move and set when the move is complete. Use GetAllOmsDoneFlags to determine when the moves are complete.

Example

See MoveOmsIndependentAbsMT for similar example.

4.12 MoveOmsIndependentRelMtWait

Move a set of axes relative positions and wait (block) until the moves are complete or timeout has expired.

C / C++
```
long MoveOmsIndependentRelMtWait(HANDLE hDevice, long AxisSelections,
AXES_DATA lPosition, long lTimeLimit);
```

VB
```
Declare Function MoveOmsIndependentRelMtWait Lib "OmsMAXkMC" (ByVal
hDevice As Long, ByVal AxisSelection As Long, ByRef lPosition As
AXES_DATA, ByVal lTimeLimit As Long) As Long
```

Return Values

SUCCESS	Request was sent
COMMAND_TIME_OUT	Command timed out
INVALID_AXIS_SELECTION	Invalid axis value
INVALID_PARAMETER	Invalid position value
MOVE_TIME_OUT	Timeout limit reached before the moves completed

Parameters

hDevice	[in] Handle to the controller
AxisSelections	[in] Axis selection bit mask
Position	[in] Pointer to AXES_DATA structure
lTimeLimit	[in] Motion timeout limit in milliseconds.

Equivalent Command String

AMMR#,#,…#; GDID;

- values passed in via *lPosition* structure.

Remarks

The selected axes will start their moves at the same time. Each axis will move to its destination, independently, based on its own velocity and acceleration parameters. The move is executed in multitask mode so that any axes, not involved in the move, will continue to execute commands in their queues. The selected axes done flags will be cleared at the start of the move and set when the move is complete.

Example

See MoveOmsIndependentAbsWait for similar example.

4.13 MoveOmsLinearAbsMt

Moves selected axes to an absolute position. Starting and stopping all selected axes at the same time.

C / C++
```
long MoveOmsLinearAbs (HANDLE hDevice, long AxisSelections, PAXES_DATA lPosition);
```

VB
```
Declare Function MoveOmsLinearAbs Lib "OmsMAXkMC" (ByVal hDevice As Long, ByVal AxisSelection As Long, ByRef lPosition As AXES_DATA) As Long
```

Return Values

SUCCESS	Request was sent
COMMAND_TIME_OUT	Command timed out
INVALID_AXIS_SELECTION	Invalid axis value
INVALID_PARAMETER	Invalid position value

Parameters

hDevice	[in] Handle to the controller
AxisSelections	[in] Axis selection bit mask
lPosition	[in] Pointer to AXES_DATA structure

Equivalent Command String

AMMT#,#,…#; GDID;
#- values passed in via *lPosition* structure.

Remarks

Linear interpolation is used to adjust the velocity and acceleration of each axis so that all selected axes start and stop their motion at the same time. The move is executed in multitasking mode so all controller axes will suspend further command execution until the move is complete. All selected axes done flags will be cleared when the move begins and set when the move is complete.

Use GetAllOmsDoneFlags to determine when the moves are complete.

Example

C / C++

```
//  Move X and T axes 4000, move Y 8000 relative
lAxes = OMS_X_AXIS + OMS_Y_AXIS + OMS_T_AXIS;
axesPosition.X = 4000;
axesPosition.Y = 8000;
axesPosition.T = 4000;

lRetVal = MoveOmsLinearAbsMt ( hDevice, lAxes, &axesPosition);

if ( lRetVal != SUCCESS )  //  Error has occurred
  //  Handle any errors
```

VB

```
‘ Move X and T axes 4000, move Y 8000 relative
lAxes = OMS_X_AXIS + OMS_Y_AXIS + OMS_T_AXIS
axesPosition.X = 4000
axesPosition.Y = 8000
axesPosition.T = 4000

lRetVal = MoveOmsLinearAbsMt ( hDevice, lAxes, axesPosition)

If ( lRetVal <> SUCCESS ) Then  ‘ Error has occurred
  ‘ Handle any errors
End If
```

4.14 MoveOmsLinearAbsMtWait

Move a set of axes to an absolute and wait (block) until the moves are complete or timeout has expired. Starting and stopping all selected axes at the same time.

C / C++
```
long MoveOmsLinearAbsMtWait(HANDLE hDevice, long AxisSelections,
AXES_DATA lPosition, long lTimeLimit);
```

VB
```
Declare Function MoveOmsLinearAbsMtWait Lib "OmsMAXkMC" (ByVal hDevice
As Long, ByVal AxisSelection As Long, ByRef lPosition As AXES_DATA,
ByVal lTimeLimit As Long) As Long
```

Return Values

SUCCESS	Request was sent
COMMAND_TIME_OUT	Command timed out
INVALID_AXIS_SELECTION	Invalid axis value
INVALID_PARAMETER	Invalid position value
MOVE_TIME_OUT	Timeout limit reached before the moves completed

Parameters

hDevice	[in] Handle to the controller
AxisSelections	[in] Axis selection bit mask
lPosition	[in] Pointer to AXES_DATA structure
lTimeLimit	[in] Motion timeout limit in milliseconds.

Equivalent Command String

AMMT#,#,…#; GDID;
- values passed in via *lPosition* structure.

Remarks

Linear interpolation is used to adjust the velocity and acceleration of each axis so that all selected axes start and stop their motion at the same time. The move is executed in multitasking mode so all controller axes will suspend further command execution until the move is complete. All selected axes done flags will be cleared when the move begins and set when the move is complete.

Example

C / C++

```
//  Move X and T axes 4000, move Y 8000 relative with 5 second wait
lAxes = OMS_X_AXIS + OMS_Y_AXIS + OMS_T_AXIS;
axesPosition.X = 4000;
axesPosition.Y = 8000;
axesPosition.T = 4000;

lRetVal = MoveOmsLinearAbsMtWait ( hDevice, lAxes, &axesPosition, 5000);

if ( lRetVal != SUCCESS )  //  Error has occurred
  //  Handle any errors
```

VB

```
‘ Move Y and Z axes -4000, move T 4000 absolute with 5 sec. wait;
lAxes = OMS_X_AXIS + OMS_Y_AXIS + OMS_T_AXIS
axesPosition.Y = -4000
axesPosition.Z = -4000
axesPosition.T = 4000

lRetVal = MoveOmsLinearAbsMtWait ( hDevice, lAxes, axesPosition, 5000)

If ( lRetVal <> SUCCESS ) Then  ‘ Error has occurred
  ‘ Handle any errors
End If
```

4.15 MoveOmsLinearRelMt

Move a set of axes relative positions. Starting and stopping all selected axes at the same time.

C / C++
```
long MoveOmsLinearRelMt (HANDLE hDevice, long AxisSelections,
PAXES_DATA lPosition);
```

VB
```
Declare Function MoveOmsLinearRelMt Lib "OmsMAXkMC" (ByVal hDevice As
Long, ByVal AxisSelection As Long, ByRef lPosition As AXES_DATA) As
Long
```

Return Values

SUCCESS	Request was sent
COMMAND_TIME_OUT	Command timed out
INVALID_AXIS_SELECTION	Invalid axis value
INVALID_PARAMETER	Invalid position value

Parameters

hDevice	[in] Handle to the controller
AxisSelections	[in] Axis selection bit mask
lPosition	[in] Pointer to AXES_DATA structure

Equivalent Command String

AAML#,#,…#; GDID;
- values passed in via *lPosition* structure.

Remarks

Linear interpolation is used to adjust the velocity and acceleration of each axis so that they start and end their motion at the same time. The move is executed in all axis mode so all controller axes will suspend further command execution until the move is complete. All axis done flags will be cleared when the move begins and set when the move is complete.

Use GetAllOmsDoneFlags to determine when the moves are complete.

Example

See MoveOmsLinearAbsMt for a comparable example.

4.16 MoveOmsLinearRelMtWait

Move a set of axes relative positions and wait (block) until the moves are complete or timeout has expired. Starting and stopping all selected axes at the same time

C / C++
```
long MoveOmsLinearRelMtWait(HANDLE hDevice, long AxisSelections,
AXES_DATA lPosition, long lTimeLimit);
```

VB
```
Declare Function MoveOmsLinearRelMtWait Lib "OmsMAXkMC" (ByVal hDevice
As Long, ByVal AxisSelection As Long, ByRef lPosition As AXES_DATA,
ByVal lTimeLimit As Long) As Long
```

Return Values

SUCCESS	Request was sent
COMMAND_TIME_OUT	Command timed out
INVALID_AXIS_SELECTION	Invalid axis value
INVALID_PARAMETER	Invalid position value
MOVE_TIME_OUT	Timeout limit reached before the moves completed

Parameters

hDevice	[in] Handle to the controller
AxisSelections	[in] Axis selection bit mask
lPosition	[in] Pointer to AXES_DATA structure
lTimeLimit	[in] Motion timeout limit in milliseconds.

Equivalent Command String

AAML#,#,…#; GDID;
#- values passed in via *lPosition* structure.

Remarks

Linear interpolation is used to adjust the velocity and acceleration of each axis so that all selected axes start and stop their motion at the same time. The move is executed in multitasking mode so all controller axes will suspend further command execution until the move is complete. All selected axes done flags will be cleared when the move begins and set when the move is complete.

Example

See MoveOmsLinearAbsMtWait for a comparable example.

5 STOP MOTION FUNCTIONS

5.01 StopOmsAxis

Stop an axis using a deceleration ramp and flush its command queue

C / C++
```
long StopOmsAxis(HANDLE hDevice, long AxisSelection);
```

VB
```
Declare Function StopOmsAxis Lib "OmsMAXkMC" (ByVal hDevice As Long,
ByVal AxisSelection As Long) As Long
```

Return Values

SUCCESS	Request was sent
COMMAND_TIME_OUT	Command timed out
INVALID_AXIS_SELECTION	Invalid axis value

Parameters

hDevice	[in] Handle to the controller
AxisSelection	[in] Selected Axis

Equivalent Command String

A*ST

***** - axis character corresponding to the value passed in *AxisSelection*.

Remarks

None

Example

See MoveOmsAxisIndefinite or MoveOmsAxisFractional

5.02 StopAllOmsAxes

Stop all axes using their deceleration ramp and flush the command queues.

C / C++
```
long StopAllOmsAxes(HANDLE hDevice);
```

VB
```
Declare Function StopAllOmsAxes Lib "OmsMAXkMC" (ByVal hDevice As Long)
As Long
```

Return Values

SUCCESS	Request was sent
COMMAND_TIME_OUT	Command timed out

Parameter

hDevice	[in] Handle to the controller

Equivalent Command String

SA or AAST

Remarks

None

Example

See MoveOmsAxisIndefinite or MoveOmsAxisFractional for similar example using StopOmsAxis

5.03 KillAllOmsMotion

Stop all axes immediately, without using their deceleration ramps and flush their command queues.

C / C++
```
long KillAllOmsMotion(HANDLE hDevice);
```

VB
```
Declare Function KillAllOmsMotion Lib "OmsMAXkMC" (ByVal hDevice As Long) As Long
```

Return Values

SUCCESS	Request was sent
COMMAND_TIME_OUT	Command timed out

Parameter

hDevice	[in] Handle to the controller

Equivalent Command String

AAKL

Remarks

None

Example

C / C++

```
//  Initiate jog move on three axes
MoveOmsAxisIndefinite ( hDevice, OMS_X_AXIS, 12500);
MoveOmsAxisIndefinite ( hDevice, OMS_Y_AXIS, 5000);
MoveOmsAxisIndefinite ( hDevice, OMS_Z_AXIS, 7500);

//  Stop all axes
lRetVal = KillAllOmsMotion (hDevice);

if ( lRetVal != SUCCESS )  //  Error has occurred
  //  Handle any errors
```

VB

```
‘ Initiate jog move on three axes
MoveOmsAxisIndefinite ( hDevice, OMS_X_AXIS, 12500)
MoveOmsAxisIndefinite ( hDevice, OMS_Y_AXIS, 5000)
MoveOmsAxisIndefinite ( hDevice, OMS_Z_AXIS, 7500)

‘ Stop all axes
lRetVal = KillAllOmsMotion (hDevice)

If ( lRetVal <> SUCCESS ) Then
  ‘ Error has occurred
End If
```

6 MOTOR and ENCODER POSITION FUNCTIONS

6.01 GetOmsAxisMotorPosition

Return the motor position of an axis

C / C++
```
long GetOmsAxisMotorPosition(HANDLE hDevice, long AxisSelection, long* lPosition);
```

VB
```
Declare Function GetOmsAxisMotorPosition Lib "OmsMAXkMC" (ByVal hDevice As Long, ByVal AxisSelection As Long, ByRef lPosition As Long) As Long
```

Return Values

SUCCESS	Request was sent
COMMAND_TIME_OUT	Command timed out
INVALID_AXIS_SELECTION	Invalid axis value

Parameters

hDevice	[in] Handle to the controller
AxisSelection	[in] Selected Axis
lPosition	[out] Long pointer to hold the motor position

Equivalent Command String

A*****RP

***** - axis character corresponding to the value passed in *AxisSelection*.

Remarks

None

Example

C / C++

```
//  Get Z axis position
lRetVal = GetOmsAxisMotorPosition ( hDevice, OMS_Z_AXIS, &lAxisPos);

if ( lRetVal != SUCCESS )  //  Error has occurred
  //  Handle any errors

//  Get Z axes encoder position
lRetVal = GetOmsAxisEncoderPosition ( hDevice, OMS_Z_AXIS, &lEncPos);
//  Handle any errors

if ( lRetVal != SUCCESS )  //  Error has occurred
  //  Handle any errors
```

VB

```
‘ Get Z axes position
lRetVal = GetOmsAxisMotorPosition ( hDevice, OMS_Z_AXIS, lAxisPos)

If ( lRetVal <> SUCCESS ) Then  ‘ Error has occurred
  ‘ Handle any errors
End If

‘ Get Z axes encoder position
lRetVal = GetOmsAxisEncoderPosition ( hDevice, OMS_Z_AXIS, lEncPos)

If ( lRetVal <> SUCCESS ) Then  ‘ Error has occurred
  ‘ Handle any errors
End If
```

6.02 GetOmsAxisEncoderPosition

Return the encoder position of selected axis.

C / C++
```
long GetOmsAxisEncoderPosition(HANDLE hDevice, long AxisSelection, long
* lPosition);
```

VB
```
Declare Function GetOmsAxisEncoderPosition Lib "OmsMAXkMC" (ByVal
hDevice As Long, ByVal AxisSelection As Long, ByRef lPosition As Long)
As Long
```

Return Values

SUCCESS	Request was sent
COMMAND_TIME_OUT	Command timed out
INVALID_AXIS_SELECTION	Invalid axis value

Parameters

hDevice	[in] Handle to the controller
AxisSelection	[in] Selected Axis
lPosition	[out] Long pointer to hold encoder position value

Equivalent Command String

A*RP

***** - axis character corresponding to the value passed in *AxisSelection*.

Remarks

None

Example

See GetOmsAxisMotorPosition

6.03 GetSelectedOmsMotorPositions

Return the current motor positions for the selected axes.

C / C++

```
long GetSelectedOmsMotorPositions(HANDLE hDevice, long AxisSelections,
PAXES_DATA lPosition);
```

VB

```
Declare Function GetSelectedOmsMotorPositions Lib "OmsMAXkMC" (ByVal
hDevice As Long, ByRef lPosition As AXES_DATA) As Long
```

Return Values

SUCCESS	Request was sent
COMMAND_TIME_OUT	Command timed out
INVALID_AXIS_SELECTION	Invalid axis value

Parameters

hDevice	[in] Handle to the controller
AxisSelections	[in] Axis selection bit mask
lPosition	[out] Pointer to AXES_DATA structure

Equivalent Command String

PP

Remarks

None

Example

C / C++

```
lRetVal = GetSelectedOmsMotorPositions ( hDevice, &AxesPos);

if ( lRetVal != SUCCESS )  //  Error has occurred
  //  Handle any errors

lRetVal = GetSelectedOmsEncoderPositions ( hDevice, &AxesEncPos);

if ( lRetVal != SUCCESS )  //  Error has occurred
  //  Handle any errors
```

VB

```
lRetVal = GetSelectedOmsMotorPositions ( hDevice, AxesPos)

If ( lRetVal <> SUCCESS ) Then  ‘ Error has occurred
  ‘ Handle any errors
End If

lRetVal = GetSelectedOmsEncoderPositions ( hDevice, AxesEncPos)

If ( lRetVal <> SUCCESS ) Then  ‘ Error has occurred
  ‘ Handle any errors
End If
```

6.04 GetSelectedOmsEncoderPositions

Return encoder positions for selected axes on the controller.

C / C++

```
long GetSelectedOmsEncoderPositions(HANDLE hDevice, long
AxisSelections, PAXES_DATA lPosition);
```

VB

```
Declare Function GetSelectedOmsEncoderPositions Lib "OmsMAXkMC" (ByVal
hDevice As Long, ByVal AxisSelections As Long, ByRef lPosition As
AXES_DATA) As Long
```

Return Values

SUCCESS	Request was sent
COMMAND_TIME_OUT	Command timed out

Parameters

hDevice	[in] Handle to the controller
AxisSelections	[in] Axis selection bit mask
lPosition	[out] Pointer to AXES_DATA structure

Equivalent Command String

PE

Remarks

The function makes a call to retrieve all axes encoder positions but only populates the None

Example

See GetSelectedOmsMotorPositions

6.05 GetMotorPositions

Return the current motor positions for all axes.

C / C++
```
long GetMotorPositions(HANDLE hDevice, PAXES_DATA lPosition);
```

VB
```
Declare Function GetMotorPositions Lib "OmsMAXkMC" (ByVal hDevice As
Long, ByRef lPosition As AXES_DATA) As Long
```

Return Values

SUCCESS	Request was sent
COMMAND_TIME_OUT	Command timed out

Parameters

hDevice	[in] Handle to the controller
lPosition	[out] Pointer to AXES_DATA structure

Equivalent Command String

PP

Remarks

None

Example

C / C++

```
AXES_DATA MotorPositions;
long lRetVal;
...
lRetVal = GetMotorPositions(hDevice, &MotorPositions);
if(lRetVal != SUCCESS)
// Handle the error conditions
else
{
 printf("X axis position = %d\n", MotorPositions.X);
 printf("Y axis position = %d\n", MotorPositions.Y);
}
```

VB

```
Dim MotorPositions As AXES_DATA
Dim lRetVal As Long
...
lRetVal = GetMotorPositions(hDevice, MotorPositions)
If(lRetVal <> SUCCESS) Then
 'Handle the error conditions
```

6.06 GetEncoderPositions

Return the current incremental encoder positions for all axes.

C / C++
```
long GetEncoderPositions(HANDLE hDevice, PAXES_DATA lPosition);
```

VB
```
Declare Function GetEncoderPositions Lib "OmsMAXkMC" (ByVal hDevice As
Long, ByRef lPosition As AXES_DATA) As Long
```

Return Values

SUCCESS	Request was sent
COMMAND_TIME_OUT	Command timed out

Parameters

hDevice	[in] Handle to the controller
lPosition	[out] Pointer to AXES_DATA structure

Equivalent Command String

PE

Remarks

None

Example

Similar to GetMotorPositions

6.07 GetPositionData

Return motor and encoder positions for all axes on the controller.

C / C++
```
long GetPositionData (HANDLE hDevice, PPOSITION_DATA lPosition);
```

VB
```
Declare Function GetPositionData Lib "OmsMAXkMC" (ByVal hDevice As
Long, ByRef lPosition As POSITION_DATA) As Long
```

Return Values

SUCCESS	Request was sent
COMMAND_TIME_OUT	Command timed out

Parameters

hDevice	[in] Handle to the controller
lPosition	[out] Pointer to POSITION_DATA structure

Equivalent Command String

PE and PP combined usage

Remarks

None

Example

C / C++

```
lRetVal = GetPositionData ( hDevice, &AxesData);

if ( lRetVal != SUCCESS )  //  Error has occurred
{
  //  Handle any errors
}
```

VB

```
lRetVal = GetPositionData ( hDevice, AxesData)

If ( lRetVal <> SUCCESS ) Then  ‘ Error has occurred
  ‘ Handle any errors
End If
```

6.08 GetOmsAxisAbsoluteEncoderPosition

Return the absolute encoder position of an axis

C / C++
```
long GetOmsAxisAbsoluteEncoderPosition(HANDLE hDevice, long
lAxisSelection, long* lEncoderPosition);
```

VB
```
Declare Function GetOmsAxisAbsoluteEncoderPosition Lib "OmsMAXkMC"
(ByVal hDevice As Long, ByVal lAxisSelection As Long, ByRef
lEncoderPosition As Long) As Long
```

Return Values

SUCCESS	Request was sent
COMMAND_TIME_OUT	Command timed out
INVALID_AXIS_SELECTION	Invalid axis value

Parameters

hDevice	[in] Handle to the controller
lAxisSelection	[in] Selected Axis
lEncoderPosition	[out] Long pointer to hold the absolute encoder position

Equivalent Command String

A*RR

***** - axis character corresponding to the value passed in *AxisSelection*.

Remarks

None

Example

C / C++

```
long lAxisSelection;
long lEncoderPosition;
long lRetVal;
...
lAxisSelection = OMS_X_AXIS;
lRetVal = GetOmsAxisAbsoluteEncoderPosition( hDevice, lAxisSelection,
&lEncoderPosition);
if(lRetVal != SUCCESS)
// Handle the error conditions
```

VB

```
Dim lAxisSelection As Long
Dim lEncoderPosition As Long
Dim lRetVal As Long
...
lAxisSelection = OMS_X_AXIS
lRetVal = GetOmsAxisAbsoluteEncoderPosition( hDevice, lAxisSelection,
lEncoderPosition)
If(lRetVal <> SUCCESS) Then
 ' Handle the error conditions
```

6.09 GetAbsoluteEncoderPositions

Return the absolute encoder positions for all axes.

C / C++

```
long GetAbsoluteEncoderPositions(HANDLE hDevice, PAXES_DATA
EncoderPositions);
```

VB

```
Declare Function GetAbsoluteEncoderPositions Lib "OmsMAXkMC" (ByVal
hDevice As Long, ByRef EncoderPositions As AXES_DATA) As Long
```

Return Values

SUCCESS	Request was sent
COMMAND_TIME_OUT	Command timed out

Parameters

hDevice	[in] Handle to the controller
EncoderPositions	[out] Pointer to AXES_DATA structure

Equivalent Command String

AARR

Remarks

None

Example

C / C++

```
AXES_DATA EncoderPositions;
long lRetVal;
...
lRetVal = GetAbsoluteEncoderPositions( hDevice, &EncoderPositions);
if(lRetVal != SUCCESS)
// Handle the error conditions
else
{
 printf("X axis position = %d\n", EncoderPositions.X);
 printf("Y axis position = %d\n", EncoderPositions.Y);
}
```

VB

```
Dim EncoderPositions As AXES_DATA
Dim lRetVal As Long
...
lRetVal = GetAbsoluteEncoderPositions( hDevice, EncodderPositions)
If(lRetVal <> SUCCESS) Then
 'Handle the error conditions
```

7 POSITION INITIALIZATION FUNCTIONS

7.01 DefineOmsHomeAsSwitchClosed

Sets the sense of the axis home switch to user active TTL state Low.

C / C++
```
long DefineOmsHomeAsSwitchClosed(HANDLE hDevice, long AxisSelection);
```

VB
```
Declare Function DefineOmsHomeAsSwitchClosed Lib "OmsMAXkMC" (ByVal
hDevice As Long, ByVal AxisSelection As Long) As Long
```

Return Values

SUCCESS	Request was sent
COMMAND_TIME_OUT	Command timed out
INVALID_AXIS_SELECTION	Invalid axis value

Parameters

hDevice	[in] Handle to the controller
AxisSelection	[in] Selected Axis

Equivalent Command String

A*HL

***** - axis character corresponding to the value passed in *AxisSelection*.

Remarks

Low is the default power up and reset state unless High was saved in the non-volatile memory.

Example

C / C++

```
//  Set X axis home switch closed
lRetVal = DefineOmsHomeAsSwitchClosed ( hDevice, OMS_X_AXIS);

if ( lRetVal != SUCCESS )  //  Error has occurred
{
  //  Handle any errors
}

//  Set Z axis home switch open
lRetVal = DefineOmsHomeAsSwitchOpen ( hDevice, OMS_Z_AXIS);

if ( lRetVal != SUCCESS )  //  Error has occurred
{
  //  Handle any errors
}
```

VB

```
‘ Set X axis home switch closed
lRetVal = DefineOmsHomeAsSwitchClosed ( hDevice, OMS_X_AXIS)

If ( lRetVal <> SUCCESS ) Then  ‘ Error has occurred
  ‘ Handle any errors
End If

‘ Set Z axis home switch open
lRetVal = DefineOmsHomeAsSwitchOpen ( hDevice, OMS_Z_AXIS)

If ( lRetVal <> SUCCESS ) Then  ‘ Error has occurred
  ‘ Handle any errors
End If
```

7.02 DefineOmsHomeAsSwitchOpen

Sets the sense of the axis home switch to user active TTL state High.

C / C++
```
long DefineOmsHomeAsSwitchOpen(HANDLE hDevice, long AxisSelection);
```

VB
```
Declare Function DefineOmsHomeAsSwitchOpen Lib "OmsMAXkMC" (ByVal
hDevice As Long, ByVal AxisSelection As Long) As Long
```

Return Values

SUCCESS	Request was sent
COMMAND_TIME_OUT	Command timed out
INVALID_AXIS_SELECTION	Invalid axis value

Parameters

hDevice	[in] Handle to the controller
AxisSelection	[in] Selected Axis

Equivalent Command String

A*HH

***** - axis character corresponding to the value passed in *AxisSelection*.

Remarks

Low is the default power up and reset state unless High was saved in the non-volatile memory.

Example

See DefineOmsHomeAsSwitchClose

7.03 FindOmsAxisFwdLimit

Move selected axis forward direction at specified velocity until it arrives at it's forward overtravel limit.

C / C++
```
long FindOmsAxisFwdLimit (HANDLE hDevice, long AxisSelection, long
Velocity, long lTimeLimit);
```

VB
```
Declare Function FindOmsAxisFwdLimit Lib "OmsMAXkMC" (ByVal hDevice As
Long, ByVal AxisSelection As Long, ByVal Velocity As Long, ByVal
lTimeLimit As Long) As Long
```

Return Values

SUCCESS	Request was sent
COMMAND_TIME_OUT	Command timed out
INVALID_AXIS_SELECTION	Invalid axis value
INVALID_PARAMETER	Invalid direction
MOVE_TIME_OUT	Timeout limit reached before finding overtravel limit

Parameters

hDevice	[in] Handle to the controller
AxisSelection	[in] Selected Axis
Velocity	[in] Axis velocity
lTimeLimit	[in] Motion timeout limit in milliseconds.

Equivalent Command String

A*LNJG#;

***** - axis character corresponding to the value passed in *AxisSelection*.

- value passed in via *Velocity*.

Remarks

This is a **blocking** function, control to the application will return when the overtravel limit is detected or the motion timeout limit has expired.

Example

C / C++

```
lVelocity = 10000;
lTimeout = 10000;      //  wait up to 10 seconds

//  Find overtravel limit to Y axis, forward and reverse
lRetVal = FindOmsAxisFwdLimit ( hDevice, OMS_Y_AXIS, lVelocity, lTimeout);
if ( lRetVal != SUCCESS )  //  Error has occurred
{
  //  Handle any errors
}
```

VB

```
lVelocity = 10000
lTimeout = 10000      ‘ Timeout after 10 seconds

‘ Find overtravel limit to X axis, forward and reverse
lRetVal = FindOmsAxisFwdLimit ( hDevice, OMS_X_AXIS, lVelocity, lTimeout)

If ( lRetVal <> SUCCESS ) Then  ‘ Error has occurred
  ‘ Handle any errors
End If
```

7.04 FindOmsAxisRevLimit

Move selected axis reverse direction at specified velocity until it arrives at it's overtravel limit.

C / C++
```
long FindOmsAxisRevLimit (HANDLE hDevice, long AxisSelection, long
Velocity, long lTimeLimit );
```

VB
```
Declare Function FindOmsAxisRevLimit Lib "OmsMAXkMC" (ByVal hDevice As
Long, ByVal AxisSelection As Long, ByVal Velocity As Long, ByVal
lTimeLimit As Long) As Long
```

Return Values

SUCCESS	Request was sent
COMMAND_TIME_OUT	Command timed out
INVALID_AXIS_SELECTION	Invalid axis value
INVALID_PARAMETER	Invalid direction
MOVE_TIME_OUT	Timeout limit reached before finding overtravel limit

Parameters

hDevice	[in] Handle to the controller
AxisSelection	[in] Selected Axis
Velocity	[in] Axis velocity
lTimeLimit	[in] Motion timeout limit in milliseconds.

Equivalent Command String

A*LNJG-#;

***** - axis character corresponding to the value passed in *AxisSelection*.

- value passed in via *Velocity*.

Remarks

This is a **blocking** function, control to the application will return when the overtravel limit is detected or the motion timeout limit has passed.

Example

See FindOmsAxisFwdLimit

7.05 HomeOmsAxisFwdUseEncoder

Move selected axis forward direction to the logic home position using the encoder index, phase A and B signals and home switch input. Sets the position once the logic has been triggered.

C / C++
```
long HomeOmsAxisFwdUseEncoder (HANDLE hDevice, long AxisSelection, long Position);
```

VB
```
Declare Function HomeOmsAxisFwdUseEncoder Lib "OmsMAXkMC" (ByVal hDevice As Long, ByVal AxisSelection As Long, ByVal Position As Long) As Long
```

Return Values

SUCCESS	Request was sent
COMMAND_TIME_OUT	Command timed out
INVALID_AXIS_SELECTION	Invalid axis value

Parameters

hDevice	[in] Handle to the controller
AxisSelection	[in] Selected Axis
Position	[in] Set home position after triggering the home logic

Equivalent Command String

A*HEHM#;ID

***** - axis character corresponding to the value passed in *AxisSelection*.

- value passed in via *Position*.

Remarks

This is **not** a blocking function and control to the application will return immediately. When home position is detected the controller will set the axis position to its home value and stop the axis. The axis done flag will be cleared when the home operation begins and will be set when it is complete. Use GetAllOmsDoneFlagsfunction can be used to determine when the home operation is complete.

Note to minimize position error the velocity of the axis should be less than 1024 steps per second when the home switch is activated. If the axis is a large distance from home you might want to:

- Use the SetOmsAxisVelocity function to set the axis velocity to a higher value than 1024.
- Home the axis.

- Reverse the sense of the home switch. See the DefineOmsHomeAsSwitchOpen and DefineOmsHomeAsSwitchClosed functions.
- Use the SetOmsAxisVelocity function to set the axis velocity to less than 1024.
- Home the axis in the opposite direction.

Example

C / C++

```
//  Find home position of Y axis in forward direction
lPosition = 100;

SetOmsAxisVelocity ( hDevice, OMS_Y_AXIS, 5000);

lRetVal = HomeOmsAxisFwdUseEncoder ( hDevice, OMS_Y_AXIS, lPosition);

//  wait till finished
```

VB

```
‘ Find home position of Y axis in forward direction
lPosition = 100

SetOmsAxisVelocity ( hDevice, OMS_Y_AXIS, 5000)

lRetVal = HomeOmsAxisFwdUseEncoder ( hDevice, OMS_Y_AXIS, lPosition)

‘ wait till finished
```

7.06 HomeOmsAxisFwdUseEncoderWait

Move selected axis forward direction until it activates the home limit switch and the encoder index position is detected.

C / C++
```
long HomeOmsAxisFwdUseEncoderWait (HANDLE hDevice, long AxisSelection,
long Position, long lTimeLimit);
```

VB
```
Declare Function HomeOmsAxisFwdUseEncoderWait Lib "OmsMAXkMC" (ByVal
hDevice As Long, ByVal AxisSelection As Long, ByVal Position As Long,
ByVal lTimeLimit As Long) As Long
```

Return Values

SUCCESS	Request was sent
COMMAND_TIME_OUT	Command timed out
INVALID_AXIS_SELECTION	Invalid axis value
MOVE_TIME_OUT	Timeout limit reached before finding home position

Parameters

hDevice	[in] Handle to the controller
AxisSelection	[in] Selected Axis
Position	[in] Set home position after triggering the home logic
lTimeLimit	[in] Motion timeout limit in milliseconds.

Equivalent Command String

A*HEHM#;ID

***** - axis character corresponding to the value passed in *AxisSelection*.

- value passed in via *Position*.

Remarks

This is a **blocking** function and control returns to the application when home has been reached or the timeout has expired. When home position is detected the controller will set the axis position to its home value and stop the axis. The axis done flag will be cleared when the home operation begins and will be set when it is complete.

Note to minimize position error the velocity of the axis should be less than 1024 steps per second when the home switch is activated. If the axis is a large distance from home you might want to:

- Use the SetOmsAxisVelocity function to set the axis velocity to a higher value than 1024.

- Home the axis.
- Reverse the sense of the home switch. See the DefineOmsHomeAsSwitchOpen and DefineOmsHomeAsSwitchClosed functions.
- Use the SetOmsAxisVelocity function to set the axis velocity to less than 1024.
- Home the axis in the opposite direction.

Example

C / C++

```
SetOmsAxisVelocity ( hDevice, OMS_Y_AXIS, 500);

//  Find Y axis encoder home position, wait 10 seconds
lRetVal = HomeOmsAxisFwdUseEncoderWait ( hDevice, OMS_Y_AXIS, 100, 10000);

if ( lRetVal != SUCCESS )  //  Error has occurred
{
  //  Handle any errors
}
```

VB

```
‘ Find Y axis encoder home position, wait 10 seconds

SetOmsAxisVelocity ( hDevice, OMS_Y_AXIS, 500)

lRetVal = HomeOmsAxisFwdUseEncoderWait ( hDevice, OMS_Y_AXIS, 100, 10000)

If ( lRetVal <> SUCCESS ) Then  ‘ Error has occurred
  ‘ Handle any errors
End If
```

7.07 HomeOmsAxisFwdUseSwitch

Move selected axis forward direction until it activates the home limit switch.

C / C++
```
long HomeOmsAxisFwdUseSwitch (HANDLE hDevice, long AxisSelection, long
lPosition);
```

VB
```
Declare Function HomeOmsAxisFwdUseSwitch Lib "OmsMAXkMC" (ByVal hDevice
As Long, ByVal AxisSelection As Long, ByVal lPosition As Long) As Long
```

Return Values

SUCCESS	Request was sent
COMMAND_TIME_OUT	Command timed out
INVALID_AXIS_SELECTION	Invalid axis value

Parameters

hDevice	[in] Handle to the controller
AxisSelection	[in] Selected Axis
Position	[in] Set home position after triggering the home switch

Equivalent Command String

A*HSHM#;ID

***** - axis character corresponding to the value passed in *AxisSelection*.

- value passed in via *Position*.

Remarks

This is **not** a blocking function and control to the application will return immediately. When home position is detected the controller will set the axis position to its home value and stop the axis. The axis done flag will be cleared when the home operation begins and will be set when it is complete.

Use GetAllOmsDoneFlagsfunction can be used to determine when the home operation is complete.

Note to minimize position error the velocity of the axis should be less than 1024 steps per second when the home switch is activated. If the axis is a large distance from home you might want to:

- Use the SetOmsAxisVelocity function to set the axis velocity to a higher value than 1024.

- Home the axis.
- Reverse the sense of the home switch. See the DefineOmsHomeAsSwitchOpen and DefineOmsHomeAsSwitchClosed functions.
- Use the SetOmsAxisVelocity function to set the axis velocity to less than 1024.
- Home the axis in the opposite direction.

Example

C / C++

```
SetOmsAxisVelocity( hDevice, OMS_Y_AXIS, 500);

//  Find Y axis home position
lRetVal = HomeOmsAxisFwdUseSwitch ( hDevice, OMS_Y_AXIS, 100);
if ( lRetVal != SUCCESS )  //  Error has occurred
{
  //  Handle any errors
}
```

VB

```
SetOmsAxisVelocity ( hDevice, OMS_Y_AXIS, 500)

‘ Find Y axis home position
lRetVal = HomeOmsAxisFwdUseSwitch ( hDevice, OMS_Y_AXIS, 100)

If ( lRetVal <> SUCCESS ) Then  ‘ Error has occurred
  ‘ Handle any errors
End If
```

7.08 HomeOmsAxisFwdUseSwitchWait

Move selected axis forward direction until it activates the home limit switch.

C / C++

```
long HomeOmsAxisFwdUseSwitchWait (HANDLE hDevice, long AxisSelection,
long lPosition, long lTimeLimit);
```

VB

```
Declare Function HomeOmsAxisFwdUseSwitchWait Lib "OmsMAXkMC" (ByVal
hDevice As Long, ByVal AxisSelection As Long, ByVal lPosition As Long,
ByVal lTimeLimit As Long) As Long
```

Return Values

SUCCESS	Request was sent
COMMAND_TIME_OUT	Command timed out
INVALID_AXIS_SELECTION	Invalid axis value
MOVE_TIME_OUT	Timeout limit reached before finding home position

Parameters

hDevice	[in] Handle to the controller
AxisSelection	[in] Selected Axis
Position	[in] Set home position after triggering the home logic
lTimeLimit	[in] Motion timeout limit in milliseconds.

Equivalent Command String

A*HSHM#;ID

***** - axis character corresponding to the value passed in *AxisSelection*.

- value passed in via *Position*.

Remarks

This is a **blocking** function and control returns to the application when the position as been reached or the timeout has expired. When the home position is detected the controller will set the axis position to its home value and stop the axis. The axis done flag will be cleared when the home operation begins and will be set when it is complete.

Note to minimize position error the velocity of the axis should be less than 1024 steps per second when the home switch is activated. If the axis is a large distance from home you might want to:

- Use the SetOmsAxisVelocity function to set the axis velocity to a higher value than 1024.
- Home the axis.

- Reverse the sense of the home switch. See the DefineOmsHomeAsSwitchOpen and DefineOmsHomeAsSwitchClosed functions.
- Use the SetOmsAxisVelocity function to set the axis velocity to less than 1024.
- Home the axis in the opposite direction.

Example

C / C++

```
SetOmsAxisVelocity( hDevice, OMS_Y_AXIS, 500);

//  Find Y axis home position, timeout after 10 seconds
lRetVal = HomeOmsAxisFwdUseSwitchWait ( hDevice, OMS_Y_AXIS, 100, 10000);
if ( lRetVal != SUCCESS )  //  Error has occurred
{
  //  Handle any errors
}
```

VB

```
SetOmsAxisVelocity ( hDevice, OMS_Y_AXIS, 500)

‘ Find Y axis home position, timeout after 10 seconds
lRetVal = HomeOmsAxisFwdUseSwitchWait ( hDevice, OMS_Y_AXIS, 100, 10000)

If ( lRetVal <> SUCCESS ) Then  ‘ Error has occurred
  ‘ Handle any errors
End If
```

7.09 HomeOmsAxisRevUseEncoder

Move selected axis reverse direction until it activates the home limit switch and the encoder index position is detected.

C / C++
```
long HomeOmsAxisRevUseEncoder (HANDLE hDevice, long AxisSelection, long lPosition);
```

VB
```
Declare Function HomeOmsAxisRevUseEncoder Lib "OmsMAXkMC" (ByVal hDevice As Long, ByVal AxisSelection As Long, ByVal lPosition As Long) As Long
```

Return Values

SUCCESS	Request was sent
COMMAND_TIME_OUT	Command timed out
INVALID_AXIS_SELECTION	Invalid axis value

Parameters

hDevice	[in] Handle to the controller
AxisSelection	[in] Selected Axis
Position	[in] Set home position after triggering the home logic

Equivalent Command String

A*HEHR#;ID

***** - axis character corresponding to the value passed in *AxisSelection*.

- value passed in via *Position*.

Remarks

This is **not** a blocking function and control to the application will return immediately. When encoder home position is detected the controller will set the axis position to its home value and stop the axis. The axis done flag will be cleared when the home operation begins and will be set when it is complete.

Use GetAllOmsDoneFlags function can be used to determine when the home operation is complete.

Note to minimize position error the velocity of the axis should be less than 1024 steps per second when the home switch is activated. If the axis is a large distance from home you might want to:

- Use the SetOmsAxisVelocity function to set the axis velocity to a higher value than 1024.
- Home the axis.
- Reverse the sense of the home switch. See the DefineOmsHomeAsSwitchOpen and DefineOmsHomeAsSwitchClosed functions.
- Use the SetOmsAxisVelocity function to set the axis velocity to less than 1024.
- Home the axis in the opposite direction.

Example

See HomeOmsAxisFwdUseEncoder for a similar example

7.10 HomeOmsAxisRevUseEncoderWait

Move selected axis reverse direction until it activates the home limit switch and the encoder index position is detected.

C / C++
```
long HomeOmsAxisRevUseEncoderWait (HANDLE hDevice, long AxisSelection,
long lPosition, long lTimeLimit);
```

VB
```
Declare Function HomeOmsAxisRevUseEncoderWait Lib "OmsMAXkMC" (ByVal
hDevice As Long, ByVal AxisSelection As Long, ByVal lPosition As Long,
ByVal lTimeLimit As Long) As Long
```

Return Values

SUCCESS	Request was sent
COMMAND_TIME_OUT	Command timed out
INVALID_AXIS_SELECTION	Invalid axis value
MOVE_TIME_OUT	Timeout limit reached before finding home position

Parameters

hDevice	[in] Handle to the controller
AxisSelection	[in] Selected Axis
Position	[in] Set home position after triggering the home logic
lTimeLimit	[in] Motion timeout limit in milliseconds.

Equivalent Command String

A*HEHR#;ID

***** - axis character corresponding to the value passed in *AxisSelection*.

- value passed in via *Position*.

Remarks

This is a **blocking** function and control returns to the application when the position as been reached or the timeout has expired. When the home position is detected the controller will set the axis position to its home value and stop the axis. The axis done flag will be cleared when the home operation begins and will be set when it is complete.

Note to minimize position error the velocity of the axis should be less than 1024 steps per second when the home switch is activated. If the axis is a large distance from home you might want to:

- Use the SetOmsAxisVelocity function to set the axis velocity to a higher value than 1024.

- Home the axis.
- Reverse the sense of the home switch. See the DefineOmsHomeAsSwitchOpen and DefineOmsHomeAsSwitchClosed functions.
- Use the SetOmsAxisVelocity function to set the axis velocity to less than 1024.
- Home the axis in the opposite direction.

Example

See HomeOmsAxisFwdUseEncoderWait for a similar example

7.11 HomeOmsAxisRevUseSwitch

Move selected axis forward direction until it activates the home limit switch.

C / C++
```
long HomeOmsAxisRevUseSwitch (HANDLE hDevice, long AxisSelection, long
lPosition);
```

VB
```
Declare Function HomeOmsAxisRevUseSwitch Lib "OmsMAXkMC" (ByVal hDevice
As Long, ByVal AxisSelection As Long, ByVal lPosition As Long) As Long
```

Return Values

SUCCESS	Request was sent
COMMAND_TIME_OUT	Command timed out
INVALID_AXIS_SELECTION	Invalid axis value

Parameters

hDevice	[in] Handle to the controller
AxisSelection	[in] Selected Axis
Position	[in] Set home position after triggering the home switch

Equivalent Command String

A*HSHR#;ID

***** - axis character corresponding to the value passed in *AxisSelection*.

- value passed in via *Position*.

Remarks

This is **not** a blocking function and control to the application will return immediately. When home limit switch is detected the controller sets the axis position to its home value and stops the axis. The axis done flag will be cleared when the home operation begins and will be set when it is complete.

Use GetAllOmsDoneFlags function can be used to determine when the home operation is complete.

Note to minimize position error the velocity of the axis should be less than 1024 steps per second when the home switch is activated. If the axis is a large distance from home you might want to:

- Use the SetOmsAxisVelocity function to set the axis velocity to a higher value than 1024.

- Home the axis.
- Reverse the sense of the home switch. See the DefineOmsHomeAsSwitchOpen and DefineOmsHomeAsSwitchClosed functions.
- Use the SetOmsAxisVelocity function to set the axis velocity to less than 1024.
- Home the axis in the opposite direction.

Example

See HomeOmsAxisFwdUseSwitch for a similar example

7.12 HomeOmsAxisRevUseSwitchWait

Move selected axis forward direction until it activates the home limit switch.

C / C++

```
long HomeOmsAxisRevUseSwitchWait (HANDLE hDevice, long AxisSelection,
long lPosition, long lTimeLimit);
```

VB

```
Declare Function HomeOmsAxisRevUseSwitchWait Lib "OmsMAXkMC" (ByVal
hDevice As Long, ByVal AxisSelection As Long, ByVal lPosition As Long,
ByVal lTimeLimit As Long) As Long
```

Return Values

SUCCESS	Request was sent
COMMAND_TIME_OUT	Command timed out
INVALID_AXIS_SELECTION	Invalid axis value
MOVE_TIME_OUT	Timeout limit reached before finding home position

Parameters

hDevice	[in] Handle to the controller
AxisSelection	[in] Selected Axis
Position	[in] Set home position after triggering the home logic
lTimeLimit	[in] Motion timeout limit in milliseconds.

Equivalent Command String

A*HSHR#;ID

***** - axis character corresponding to the value passed in *AxisSelection*.

- value passed in via *Position*.

Remarks

This is a **blocking** function and control returns to the application when the position as been reached or the timeout has expired. When home limit switch is detected the controller will set the axis position to its home value and stop the axis. The axis done flag will be cleared when the home operation begins and will be set when it is complete.

Note to minimize position error the velocity of the axis should be less than 1024 steps per second when the home switch is activated. If the axis is a large distance from home you might want to:

- Use the SetOmsAxisVelocity function to set the axis velocity to a higher value than 1024.
- Home the axis.

- Reverse the sense of the home switch. See the DefineOmsHomeAsSwitchOpen and DefineOmsHomeAsSwitchClosed functions.
- Use the SetOmsAxisVelocity function to set the axis velocity to less than 1024.
- Home the axis in the opposite direction.

Example

See HomeOmsAxisFwdUseSwitchWait for a similar example

7.13 SetOmsAxisPosition

Set motor position of an axis to the specified value.

C / C++
```
long SetOmsAxisPosition (HANDLE hDevice, long AxisSelection, long
lPosition);
```

VB
```
Declare Function SetOmsAxisPosition Lib "OmsMAXkMC" (ByVal hDevice As
Long, ByVal AxisSelection As Long, ByVal lPosition As Long) As Long
```

Return Values

SUCCESS	Request was sent
COMMAND_TIME_OUT	Command timed out
INVALID_AXIS_SELECTION	Invalid axis value

Parameters

hDevice	[in] Handle to the controller
AxisSelection	[in] Selected Axis
Position	[in] Set home position after triggering the home logic

Equivalent Command String

A*LP#;

***** - axis character corresponding to the value passed in *AxisSelection*.

- value passed in *Position*.

Remarks

None

Example

C / C++

```
//  Set Z axis position to 1500
lRetVal = SetOmsAxisPosition ( hDevice, OMS_Z_AXIS, 1500);
if ( lRetVal != SUCCESS )  //  Error has occurred
{
  //  Handle any errors
}
```

VB

```
‘ Set Z axis position to 1500
lRetVal = SetOmsAxisPosition ( hDevice, OMS_Z_AXIS, 1500)
If ( lRetVal <> SUCCESS ) Then  ‘ Error has occurred
  ‘ Handle any errors
End If
```

8 VELOCITY CONTROL and REPORTING FUNCTIONS

8.01 GetOmsAxisVelocity

Returns the current velocity at which the axis is moving.

C / C++

```
long GetOmsAxisVelocity (HANDLE hDevice, long AxisSelection, long
*Velocity);
```

VB

```
Declare Function GetOmsAxisVelocity Lib "OmsMAXkMC" (ByVal hDevice As
Long, ByVal AxisSelection As Long, ByRef Velocity As Long) As Long
```

Return Values

SUCCESS	Request was sent
RESPONSE_TIME_OUT	Timeout waiting for response from the controller
INVALID_AXIS_SELECTION	Invalid axis value

Parameters

hDevice	[in] Handle to the controller
AxisSelection	[in] Selected Axis
Velocity	[out] long pointer to hold velocity value

Equivalent Command String

A*****RV

***** - axis character corresponding to the value passed in *AxisSelection*.

Remarks

None

Example

C / C++

```
lRetVal = GetOmsAxisVelocity (hDevice, OMS_Y_AXIS, &lVelocity);
```

VB

```
lRetVal = GetOmsAxisVelocity (hDevice, OMS_Y_AXIS, lVelocity)
```

8.02 GetSelectedOmsVelocities

Return the current velocities of the selected axes in AXES_DATA structure.

C / C++

```
long GetSelectedOmsVelocities (HANDLE hDevice, long AxisSelection,
PAXES_DATA Velocities);
```

VB

```
Declare Function GetSelectedOmsVelocities Lib "OmsMAXkMC" (ByVal
hDevice As Long, ByVal AxisSelection As Long, ByRef Velocities As
AXES_DATA) As Long
```

Return Values

SUCCESS	Request was sent
RESPONSE_TIME_OUT	Timeout waiting for response from the controller
INVALID_AXIS_SELECTION	Invalid axis value

Parameters

hDevice	[in] Handle to the controller
AxisSelection	[in] Selected Axis
Velocities	[out] AXES_DATA pointer to hold velocities values

Equivalent Command String

AMRV

Remarks

None

Example

C / C++

```
lRetVal = GetSelectedOmsVelocities ( hDevice, OMS_X_AXIS + OMS_Y_AXIS,
                                     &axesVelocities);
```

VB

```
lRetVal = GetSelectedOmsVelocities ( hDevice, OMS_X_AXIS + OMS_Y_AXIS, _
                                     axesVelocities)
```

8.03 SetOmsAxisBaseVelocity

Set the base (starting) acceleration ramp of a specified axis.

C / C++
```
long SetOmsAxisBaseVelocity (HANDLE hDevice, long AxisSelection, long Velocity);
```

VB
```
Declare Function SetOmsAxisBaseVelocity Lib "OmsMAXkMC" (ByVal hDevice As Long, ByVal AxisSelection As Long, ByVal Velocity As Long) As Long
```

Return Values

SUCCESS	Request was sent
COMMAND_TIME_OUT	Command timed out
INVALID_AXIS_SELECTION	Invalid axis value
INVALID_PARAMETER	Invalid velocity parameter

Parameters

hDevice	[in] Handle to the controller
AxisSelection	[in] Selected Axis
Velocity	[in] Base Axis velocity

Equivalent Command String

A*VB#;

***** - axis character corresponding to the value passed in *AxisSelection*.

- value passed in via *Velocity*.

Remarks

Setting a nonzero base velocity provides a means to accelerate a stepper motor through its low velocity resonance region as fast as possible. The *Velocity* (base velocity) must be less then the maximum velocity value.

Example

C / C++

```
//  Set base velocity
lRetVal = SetOmsAxisBaseVelocity ( hDevice, OMS_X_AXIS, 500);

if ( lRetVal != SUCCESS )  //  Error has occurred
{
  //  Handle any errors
}

//  Set maximum velocity
lRetVal = SetOmsAxisVelocity ( hDevice, OMS_Z_AXIS, 25000);

if ( lRetVal != SUCCESS )  //  Error has occurred
{
  //  Handle any errors
}
```

VB

```
‘ Set base velocity
lRetVal = SetOmsAxisBaseVelocity ( hDevice, OMS_X_AXIS, 500)

If ( lRetVal <> SUCCESS ) Then  ‘ Error has occurred
  ‘ Handle any errors
End If

‘ Set maximum velocity
lRetVal = SetOmsAxisVelocity ( hDevice, OMS_Z_AXIS, 25000)

If ( lRetVal <> SUCCESS ) Then  ‘ Error has occurred
  ‘ Handle any errors
End If
```

8.04 SetOmsAxisVelocity

Set the maximum velocity of an axis.

C / C++
```
long SetOmsAxisVelocity (HANDLE hDevice, long AxisSelection, long Velocity);
```

VB
```
Declare Function SetOmsAxisVelocity Lib "OmsMAXkMC" (ByVal hDevice As Long, ByVal AxisSelection As Long, ByVal Velocity As Long) As Long
```

Return Values

SUCCESS	Request was sent
COMMAND_TIME_OUT	Command timed out
INVALID_AXIS_SELECTION	Invalid axis value
INVALID_PARAMETER	Invalid velocity parameter

Parameters

hDevice	[in] Handle to the controller
AxisSelection	[in] Selected Axis
Velocity	[in] Maximum Axis velocity

Equivalent Command String

A*VL#;

***** - axis character corresponding to the value passed in *AxisSelection*.

- value passed in via *Velocity*.

Remarks

None

Example

See SetOmsAxisBaseVelocity

9 ACCELERATION CONTROL and REPORTING FUNCTIONS

9.01 SetOmsAxisAcceleration

Set the maximum acceleration rate of an axis.

C / C++
```
long SetOmsAxisAcceleration (HANDLE hDevice, long AxisSelection, long
lAxisAccel);
```

VB
```
Declare Function SetOmsAxisAcceleration Lib "OmsMAXkMC" (ByVal hDevice
As Long, ByVal AxisSelection As Long, ByVal lAxisAccel As Long) As Long
```

Return Values

SUCCESS	Request was sent
COMMAND_TIME_OUT	Command timed out
INVALID_AXIS_SELECTION	Invalid axis value
INVALID_PARAMETER	Invalid acceleration parameter

Parameters

hDevice	[in] Handle to the controller
AxisSelection	[in] Selected Axis
lAxisAccel	[in] Axis acceleration value

Equivalent Command String

A*AC#;

***** - axis character corresponding to the value passed in *AxisSelection*.

- value passed in via *lAxisAccel*.

Remarks

Acceleration value can only be between 1 and 8,000,000.

Example

C / C++

```
//  Set X axis max acceleration to 100000
lRetVal = SetOmsAxisAcceleration ( hDevice, OMS_X_AXIS, 100000);
//  Handle any errors

lRetVal = GetOmsAxisAcceleration ( hDevice, OMS_X_AXIS, &lAccel);
//  Handle any errors
```

VB

```
‘ Set X axis max acceleration to 100000
lRetVal = SetOmsAxisAcceleration ( hDevice, OMS_X_AXIS, 100000)
‘ Handle any errors

lRetVal = GetOmsAxisAcceleration ( hDevice, OMS_X_AXIS, lAccel)
‘ Handle any errors
```

9.02 GetOmsAxisAcceleration

Retrieve the set acceleration of a specified axis.

C / C++
```
long GetOmsAxisAcceleration (HANDLE hDevice, long AxisSelection, long* lAcceleration);
```

VB
```
Declare Function GetOmsAxisAcceleration Lib "OmsMAXkMC" (ByVal hDevice As Long, ByVal AxisSelection As Long, ByRef lAcceleration As Long) As Long
```

Return Values

SUCCESS	Request was sent
COMMAND_TIME_OUT	Command timed out
INVALID_AXIS_SELECTION	Invalid axis value

Parameters

hDevice	[in] Handle to the controller
AxisSelection	[in] Selected Axis
lAcceleration	[out] long pointer for acceleration value

Equivalent Command String

A*?AC

***** - axis character corresponding to the value passed in *AxisSelection*.

Remarks

None

Example

See SetOmsAxisAcceleration

9.03 GetSelectedOmsAccelerations

Retrieve current accelerations of specified axes.

C / C++

```
long GetSelectedOmsAccelerations (HANDLE hDevice, long AxisSelections,
PAXES_DATA lAccelerations);
```

VB

```
Declare Function GetSelectedOmsAccelerations Lib "OmsMAXkMC" (ByVal
hDevice As Long, ByVal AxisSelections As Long, ByRef lAccelerations As
AXES_DATA) As Long
```

Return Values

SUCCESS	Request was sent
COMMAND_TIME_OUT	Command timed out
INVALID_AXIS_SELECTION	Invalid axis bit mask

Parameters

hDevice	[in] Handle to the controller
AxisSelection	[in] Selected Axis
lAccelerations	[out] AXES_DATA pointer for acceleration values

Equivalent Command String

AMRC

Remarks

Only the requested axes accelerations are populated in the AXES_DATA structure.

Example

None

9.04 SelectOmsCosineRamp

Set cosine acceleration ramp for a selected axis.

C / C++
```
long SelectOmsCosineRamp (HANDLE hDevice, long AxisSelection);
```

VB
```
Declare Function SelectOmsCosineRamp Lib "OmsMAXkMC" (ByVal hDevice As
Long, ByVal AxisSelection As Long) As Long
```

Return Values

SUCCESS	Request was sent
COMMAND_TIME_OUT	Command timed out
INVALID_AXIS_SELECTION	Invalid axis value

Parameters

hDevice	[in] Handle to the controller
AxisSelection	[in] Selected Axis

Equivalent Command String

A*SC

***** - axis character corresponding to the value passed in *AxisSelection*.

Remarks

None

Example

C / C++

```
lRetVal = SelectOmsCosineRamp ( hDevice, OMS_Y_AXIS);
```

VB

```
lRetVal = SelectOmsCosineRamp ( hDevice, OMS_Y_AXIS)
```

9.05 SelectOmsLinearRamp

Set linear acceleration ramp for a selected axis.

C / C++
```
long SelectOmsLinearRamp (HANDLE hDevice, long AxisSelection);
```

VB
```
Declare Function SelectOmsLinearRamp Lib "OmsMAXkMC" (ByVal hDevice As Long, ByVal AxisSelection As Long) As Long
```

Return Values

SUCCESS	Request was sent
COMMAND_TIME_OUT	Command timed out
INVALID_AXIS_SELECTION	Invalid axis value

Parameters

hDevice	[in] Handle to the controller
AxisSelection	[in] Selected Axis

Equivalent Command String

A*LA

***** - axis character corresponding to the value passed in *AxisSelection*.

Remarks

None

Example

C / C++

```
lRetVal = SelectOmsLinearRamp ( hDevice, OMS_Z_AXIS);
```

VB

```
lRetVal = SelectOmsLinearRamp ( hDevice, OMS_Z_AXIS)
```

9.06 SelectOmsParabolicRamp

Set parabolic acceleration ramp for a selected axis.

C / C++
```
long SelectOmsParabolicRamp (HANDLE hDevice, long AxisSelection, long Steps);
```

VB
```
Declare Function SelectOmsParabolicRamp Lib "OmsMAXkMC" (ByVal hDevice As Long, ByVal AxisSelection As Long, ByVal Steps As Long) As Long
```

Return Values

SUCCESS	Request was sent
COMMAND_TIME_OUT	Command timed out
INVALID_AXIS_SELECTION	Invalid axis value

Parameters

hDevice	[in] Handle to the controller
AxisSelection	[in] Selected Axis
Steps	[in] Number of linear segments in the ramp

Equivalent Command String

A*PR#;

***** - axis character corresponding to the value passed in *AxisSelection*.

- value passed in via *Steps*.

Remarks

If *Steps* is set equal to zero then a special three segment version of the parabolic ramp will be used. If *Steps* is between 3 and 10 then the number of segments determines at what segment the parabolic acceleration ramp will be truncated.

Example

C / C++

```
//  Set Z axis ramp to 8 parabolic ramp steps
lRetVal = SelectOmsParabolicRamp ( hDevice, OMS_Z_AXIS, 8);
```

VB

```
‘ set Z axis ramp to 8 parabolic ramp steps
lRetVal = SelectOmsParabolicRamp ( hDevice, OMS Z AXIS, 8)
```

9.07 SelectOmsSCurveRamp

Set S Curve ramp for a selected axis.

C / C++
```
long SelectOmsSCurveRamp (HANDLE hDevice, long AxisSelection, long RampSelection);
```

VB
```
Declare Function SelectOmsSCurveRamp Lib "OmsMAXkMC" (ByVal hDevice As Long, ByVal AxisSelection As Long, ByVal RampSelection As Long) As Long
```

Return Values

SUCCESS	Request was sent
COMMAND_TIME_OUT	Command timed out
INVALID_AXIS_SELECTION	Invalid axis value
INVALID_PARAMETER	Invalid ramp selection

Parameters

hDevice	[in] Handle to the controller
AxisSelection	[in] Selected Axis
RampSelection	[in] Number of S Curve ramps

Equivalent Command String

A*SS#;

***** - axis character corresponding to the value passed in *AxisSelection*.

- value passed in via *RampSelection*.

Remarks

Use the AJ command to define the custom S Curve for the axis. A maximum number of 8 ramp profiles are available.

Example

C / C++

```
//  Select ramp profile 1 for Z axis
lRetVal = SelectOmsSCurveRamp ( hDevice, OMS Z AXIS, 1);
```

VB

```
‘ Select ramp profile 1 for Z axis
lRetVal = SelectOmsSCurveRamp ( hDevice, OMS Z AXIS, 1)
```

10 AXIS/AXES EVENT FLAGS FUNCTIONS

10.01 GetOmsAxisDoneFlag

Return the done flag status of a selected axis.

C / C++
```
long GetOmsAxisDoneFlag (HANDLE hDevice, long AxisSelection, long *
pFlagStatus);
```

VB
```
Declare Function GetOmsAxisDoneFlag Lib "OmsMAXkMC" (ByVal hDevice As
Long, ByVal AxisSelection As Long, ByRef pFlagStatus As Long) As Long
```

Return Values

SUCCESS	Request was sent
INVALID_AXIS_SELECTION	Invalid axis value

Parameters

hDevice	[in] Handle to the controller
AxisSelection	[in] Selected Axis
pFlagStatus	[out] long pointer for axis done flag

Equivalent Command String

QA (Returns a similar result)

Remarks

TRUE if the done flag is set, FALSE if not set. Once a done flag is set it remains set until it is cleared via a call to the ClrOmsDoneFlags function.

Example

C / C++

```
//  Get T axis done flag
lRetVal = GetOmsAxisDoneFlag ( hDevice, OMS_T_AXIS, &lFlagStatus);
```

VB

```
‘ Get T axis done flag
lRetVal = GetOmsAxisDoneFlag ( hDevice, OMS T AXIS, lFlagStatus)
```

10.02 GetDoneFlags

Same as GetAllOmsDoneFlags

10.03 GetAllOmsDoneFlags

Returns the axis done flag status of all the axes.

C / C++
```
long GetAllOmsDoneFlags (HANDLE hDevice, long* FlagStatus);
```

VB
```
Declare Function GetAllOmsDoneFlags Lib "OmsMAXkMC" (ByVal hDevice As
Long, ByRef FlagStatus As Long) As Long
```

Return Value

SUCCESS	Request was sent

Parameters

hDevice	[in] Handle to the controller
FlagMask	[out] long pointer for done flags

Equivalent Command String

`QI` (Returns a similar result)

Remarks

Axis bit assignments:

bit 0	=	X axis	=	OMS_X_AXIS
bit 1	=	Y axis	=	OMS_Y_AXIS
bit 2	=	Z axis	=	OMS_Z_AXIS
bit 3	=	T axis	=	OMS_T_AXIS
bit 4	=	U axis	=	OMS_U_AXIS
bit 5	=	V axis	=	OMS_V_AXIS
bit 6	=	R axis	=	OMS_R_AXIS
bit 7	=	S axis	=	OMS_S_AXIS
bit 8	=	W axis	=	OMS_W_AXIS
bit 9	=	K axis	=	OMS_K_AXIS

Once a done flag is set it remains set until it is cleared via a call to the ClrOmsDoneFlags function.

Example

C / C++

```
//  Check for X and Z axes done flag status
lRetVal = GetAllOmsDoneFlags ( hDevice, &lFlags);
if ( lRetVal != SUCCESS )  //  Error has occurred
{
  //  Handle any errors
}

if( (lFlags & (OMS_X_AXIS + OMS_Z_AXIS)) == (OMS_X_AXIS + OMS_Z_AXIS))
{
  // Clear and handle X & Z axes done condition...
  lRetVal = ClrOmsDoneFlags ( hDevice, OMS_X_AXIS + OMS_Z_AXIS);
}
```

VB

```
‘ Check for X and Z axes done flag status
lRetVal = GetAllOmsDoneFlags ( hDevice, lFlags)

If ( lRetVal <> SUCCESS ) Then  ‘ Error has occurred
  ‘ Handle any errors
End If

If((lFlags And (OMS_X_AXIS + OMS_Z_AXIS)) = (OMS_X_AXIS + OMS_Z_AXIS)) Then
  ' Clear and handle X & Z axis done condition...
  lRetVal = ClrOmsDoneFlags ( hDevice, OMS_X_AXIS + OMS_Z_AXIS)
End If
```

10.04 ClrDoneFlags

Same as ClrOmsDoneFlags

10.05 ClrOmsDoneFlags

Clear selected axis done flags.

C / C++
```
long ClrOmsDoneFlags (HANDLE hDevice, long FlagMask);
```

VB
```
Declare Function ClrOmsDoneFlags Lib "OmsMAXkMC" (ByVal hDevice As
Long, ByVal FlagMask As Long) As Long
```

Return Values

SUCCESS	Request was sent
COMMAND_TIME_OUT	Command timed out
INVALID_AXIS_SELECTION	Invalid axis flag mask

Parameters

hDevice	[in] Handle to the controller
FlagMask	[in] flag mask

Equivalent Command String

None

Remarks

This function clears the copy of the axis done flags that is maintained by the driver and not the copy of the done flags that is maintained within the controller. Depending on what value is passed in *FlagMask*, all or selected number of axes done flags can be cleared.

Controller Axis selection constants:

1	= OMS_X_AXIS	2	= OMS_Y_AXIS
4	= OMS_Z_AXIS	8	= OMS_T_AXIS
16	= OMS_U_AXIS	32	= OMS_V_AXIS
64	= OMS_R_AXIS	128	= OMS_S_AXIS
256	= OMS_W_AXIS	512	= OMS_K_AXIS

Multiple axes are selected by adding individual axis selection values.

If your application obtains its done flag information exclusively from the GetOmsAxisDoneFlag or GetAllOmsDoneFlags functions then you do not need to be concerned about clearing the controller's copy of the done flags. However if your application acquires done flag information via the RA, QA, RI and QI controller commands then it must use the appropriate controller commands to clear the controller's copy of the done flags. See the controller manual for descriptions of the IC, CA, RA and RI commands.

Example

See GetOmsAxisDoneFlag or GetAllOmsDoneFlags

10.06 GetLimitFlags

Same as GetAllOmsLimitFlags

10.07GetAllOmsLimitFlags

Returns the overtravel status of all the axes.

C / C++
```
long GetAllOmsLimitFlags (HANDLE hDevice, long* FlagStatus);
```

VB
```
Declare Function GetAllOmsLimitFlags Lib "OmsMAXkMC" (ByVal hDevice As Long, ByRef FlagStatus As Long) As Long
```

Return Value

SUCCESS	Request was sent
COMMAND_TIME_OUT	Command timed out

Parameters

hDevice	[in] Handle to the controller
FlagStatus	[in] long pointer for Overtravel flag

Equivalent Command String

None

Remarks

Axis bit assignments:

bit 0	= X axis	= OMS_X_AXIS	bit 1	= Y axis	= OMS_Y_AXIS
bit 2	= Z axis	= OMS_Z_AXIS	bit 3	= T axis	= OMS_T_AXIS
bit 4	= U axis	= OMS_U_AXIS	bit 5	= V axis	= OMS_V_AXIS
bit 6	= R axis	= OMS_R_AXIS	bit 7	= S axis	= OMS_S_AXIS
bit 8	= W axis	= OMS_W_AXIS	bit 9	= K axis	= OMS_K_AXIS

TRUE if Overtravel flag is set, FALSE if not set. Once a done flag is set it remains set until it is cleared via a call to the ClrOmsLimitFlags function.

Example

C / C++

```
//  Check for X and Z axes overtravel status
lRetVal = GetAllOmsLimitFlags ( hDevice, &lFlags);

if ( lRetVal != SUCCESS )  //  Error has occurred
{
  //  Handle any errors
}

if( (lFlags & (OMS_X_AXIS + OMS_Z_AXIS)) == (OMS_X_AXIS + OMS_Z_AXIS))
{
  //  Clear X & Z axis done flags and handle overtravel condition...
  lRetVal = ClrOmsLimitFlags ( hDevice, OMS_X_AXIS + OMS_Z_AXIS);
}
```

VB

```
‘ Check for X and Z axes done flag status
lRetVal = GetAllOmsLimitFlags ( hDevice, lFlags)

If ( lRetVal <> SUCCESS ) Then  ‘ Error has occurred
  ‘ Handle any errors
End If

If((lFlags And (OMS_X_AXIS + OMS_Z_AXIS)) = (OMS_X_AXIS + OMS_Z_AXIS)) Then
  ‘ Clear X & Z axis done flags and handle overtravel condition...
  lRetVal = ClrOmsLimitFlags ( hDevice, OMS_X_AXIS + OMS_Z_AXIS);
End If
```

10.08 GetOmsAxisLimitFlag

Returns the overtravel status flag of a selected axis.

C / C++

```
long GetOmsAxisLimitFlag (HANDLE hDevice, long AxisSelection, long*
FlagStatus);
```

VB

```
Declare Function GetOmsAxisLimitFlag Lib "OmsMAXkMC" (ByVal hDevice As
Long, ByVal AxisSelection As Long, ByRef FlagStatus As Long) As Long
```

Return Value

SUCCESS	Request was sent
COMMAND_TIME_OUT	Command timed out
INVALID_AXIS_SELECTION	Invalid axis

Parameters

hDevice	[in] Handle to the controller
AxisSelection	[in] Selected Axis
FlagStatus	[in] long pointer for Overtravel flag

Equivalent Command String

None

Remarks

TRUE if overtravel flag is set, FALSE if not set. Once a done flag is set it remains set until it is cleared via a call to the ClrOmsLimitFlags function.

Example

C / C++

```
//  Check for Y axis overtravel status
lRetVal = GetOmsAxisLimitFlag ( hDevice, OMS_Y_AXIS, &lFlags);

if ( lRetVal != SUCCESS )  //  Error has occurred
{
  //  Handle any errors
}

if( lFlags != 0)
{
  //  Clear y axis done flags and handle overtravel condition...
  lRetVal = ClrOmsLimitFlags ( hDevice, OMS_Y_AXIS);
}
```

VB

```
‘ Check for Y axis overtravel status
lRetVal = GetOmsAxisLimitFlag ( hDevice, OMS_Y_AXIS, &lFlags)

If ( lRetVal <> SUCCESS ) Then  ‘ Error has occurred
  ‘ Handle any errors
End If

If( lFlags <> 0) Then
  ‘ Clear y axis done flags and handle overtravel condition...
  lRetVal = ClrOmsLimitFlags ( hDevice, OMS_Y_AXIS)
End If
```

10.09 ClrLimitFlags

Same as ClrOmsLimitFlags

10.10ClrOmsLimitFlags

Clear all overtravel flags.

C / C++
```
long ClrOmsLimitFlags (HANDLE hDevice, long FlagMask);
```

VB
```
Declare Function ClrOmsLimitFlags Lib "OmsMAXkMC" (ByVal hDevice As
Long, ByVal FlagMask As Long) As Long
```

Return Values

SUCCESS	Request was sent
COMMAND_TIME_OUT	Command timed out
INVALID_AXIS_SELECTION	Invalid flag mask

Parameters

hDevice	[in] Handle to the controller
FlagMask	[in] flag mask

Equivalent Command String

None

Remarks

This function clears the copy of the overtravel flags that is maintained by the driver, not the copy maintained within the controller. Depending on what value is passed in *FlagMask*, all or selected number of axes done flags can be cleared.

Controller Axis selection constants:

1	= OMS_X_AXIS	2	= OMS_Y_AXIS
4	= OMS_Z_AXIS	8	= OMS_T_AXIS
16	= OMS_U_AXIS	32	= OMS_V_AXIS
64	= OMS_R_AXIS	128	= OMS_S_AXIS
256	= OMS_W_AXIS	512	= OMS_K_AXIS

Multiple axes are selected by adding individual axis selection values.

If your application obtains its done flag information exclusively from theGetAllOmsDoneFlags, GetOmsAxisDoneFlag or GetOmsAxisFlags functions then you do not need to be concerned about clearing the controller's copy of the done flags. However if your application acquires done flag information via the RA, QA, RI and QI controller commands then it must use the appropriate controller commands to clear the controller's copy of the done flags. See the controller manual for descriptions of the IC, CA, RA and RI commands.

Example

See GetAllOmsLimitFlags

10.11 GetOmsControllerFlags

Return the state of the controller's error flag.

C / C++
```
long GetOmsControllerFlags (HANDLE hDevice, long* pFlags);
```

VB
```
Declare Function GetOmsControllerFlags Lib "OmsMAXkMC" (ByVal hDevice
As Long, ByRef pFlags As Long) As Long
```

Return Value

SUCCESS	Request was sent
COMMAND_TIME_OUT	Command timed out

Parameters

hDevice	[in] Handle to the controller
pFlags	[out] long pointer for error flag value

Equivalent Command String

None

Remarks

The controller command error is acquired from the flags that are latched by the device driver. When an error condition is detected the driver latches any flags that are on. The application must then clear that flag once the error condition has been handled. See ClrOmsControllerFlags

Example

C / C++

```
//  Get command error flag
lRetVal = GetOmsControllerFlags ( hDevice, &lCmdErrorFlag);
if ( lRetVal != SUCCESS )  //  Error has occurred
{
  //  Handle any errors
}

lRetVal = ClrOmsControllerFlags ( hDevice, 0);
if( lFlagStatus )
  //  Handle command error
```

VB

```
‘ Get command error flag
lRetVal = GetOmsControllerFlags ( hDevice, lCmdErrorFlag)

If ( lRetVal <> SUCCESS ) Then  ‘ Error has occurred
  ‘ Handle any errors
End If

If( lFlagStatus = 1 ) Then
  ' Handle encoder slip
  lRetVal = ClrOmsControllerFlags ( hDevice, 0)
End If
```

10.12 ClrStatusFlags

See ClrOmsControllerFlags

10.13ClrOmsControllerFlags

Clears the driver's copy of the command error bit.

C / C++
```
long ClrOmsControllerFlags (HANDLE hDevice, long FlagMask);
```

VB
```
Declare Function ClrOmsControllerFlags Lib "OmsMAXkMC" (ByVal hDevice
As Long, ByVal FlagMask As Long) As Long
```

Return Values

SUCCESS	Request was sent
COMMAND_TIME_OUT	Command timed out

Parameters

hDevice	[in] Handle to the controller
FlagMask	[in] flag mask

Equivalent Command String

None

Remarks

None

Example

See GetOmsControllerFlags

10.14 GetOmsAxisFlags

Return the status flags of a selected axis.

C / C++
```
long GetOmsAxisFlags (HANDLE hDevice, long AxisSelection, long*
FlagStatus);
```

VB
```
Declare Function GetOmsAxisFlags Lib "OmsMAXkMC" (ByVal hDevice As
Long, ByVal AxisSelection As Long, ByRef FlagStatus As Long) As Long
```

Return Values

SUCCESS	Request was sent
INVALID_AXIS_SELECTION	Invalid axis value

Parameters

hDevice	[in] Handle to the controller
AxisSelection	[in] Selected Axis
FlagStatus	[in] long pointer for flag status

Equivalent Command String

QA

Remarks

FlagStatus content is:

Bit 0 =	AXIS_DIRECTION	= {POSITIVE = 0, NEGATIVE = 1}
Bit 1 =	AXIS_DONE	= {FALSE = 0, TRUE = 1}
Bit 2 =	AXIS_OVERTRAVEL	= {FALSE = 0, TRUE = 1}
Bit 3 =	AXIS_HOME_SWITCH	= {FALSE = 0, TRUE = 1}

The axis done flag and overtravel status are acquired from the flags that are latched by the MAXk device driver.

Example

C / C++

```
//  Get T axis flags
lRetVal = GetOmsAxisFlags ( hDevice, OMS_T_AXIS, &lFlagStatus);

if ( lRetVal != SUCCESS )  //  Error has occurred
{
  //  Handle any errors
}

if( (lFlagStatus & AXIS_OVERTRAVEL) != 0 )
  //  Handle overtravel
```

VB

```
‘ Get T axis flags
lRetVal = GetOmsAxisFlags ( hDevice, OMS_T_AXIS, lFlagStatus)

If ( lRetVal <> SUCCESS ) Then  ‘ Error has occurred
  ‘ Handle any errors
End If

If( lFlagStatus And AXIS_OVERTRAVEL ) <> 0 ) Then
  ' Handle overtravel
End If
```

10.15 GetOmsLimitSensors

Return the status of the limit sensors for all axes.

C / C++
```
long GetOmsLimitSensors (HANDLE hDevice, long* FlagStatus);
```

VB
```
Declare Function GetOmsLimitSensors Lib "OmsMAXkMC" (ByVal hDevice As Long, ByRef FlagStatus As Long) As Long
```

Return Values

SUCCESS	Request was successful
COMMAND_TIME_OUT	Command timed out

Parameters

hDevice	[in] Handle to the controller
FlagStatus	[out] long pointer for limit sensor status

Equivalent Command String

None

Remarks

Axis bit assignments:

bit 00	=	X	axis negative travel limit sensor state
bit 01	=	Y	axis negative travel limit sensor state
bit 02	=	Z	axis negative travel limit sensor state
bit 03	=	T	axis negative travel limit sensor state
bit 04	=	U	axis negative travel limit sensor state
bit 05	=	V	axis negative travel limit sensor state
bit 06	=	R	axis negative travel limit sensor state
bit 07	=	S	axis negative travel limit sensor state
bit 08	=	W	axis negative travel limit sensor state
bit 09	=	K	axis negative travel limit sensor state
bit 10	=	X	axis positive travel limit sensor state
bit 11	=	Y	axis positive travel limit sensor state
bit 12	=	Z	axis positive travel limit sensor state
bit 13	=	T	axis positive travel limit sensor state
bit 14	=	U	axis positive travel limit sensor state
bit 15	=	V	axis positive travel limit sensor state
bit 16	=	R	axis positive travel limit sensor state
bit 17	=	S	axis positive travel limit sensor state
bit 18	=	W	axis positive travel limit sensor state

bit 19 = K axis positive travel limit sensor state
bit 20 to 31 – not used

Example

C / C++

```
lRetVal = GetOmsLimitSensors ( hDevice, &lLimitSensors);
if ( lRetVal != SUCCESS )  //  Error has occurred
{
  //  Handle any errors
}

lRetVal = GetOmsHomeSensors ( hDevice, &lHomeSensors);
if ( lRetVal != SUCCESS )  //  Error has occurred
{
  //  Handle any errors
}
```

VB

```
lRetVal = GetOmsLimitSensors ( hDevice, lLimitSensors)
If ( lRetVal <> SUCCESS ) Then  ‘ Error has occurred
  ‘ Handle any errors
End If

lRetVal = GetOmsHomeSensors ( hDevice, lHomeSensors)
If ( lRetVal <> SUCCESS ) Then  ‘ Error has occurred
  ‘ Handle any errors
End If
```

10.16 GetOmsHomeSensors

Return the status of the home sensors for all axes.

C / C++
```
long GetOmsHomeSensors (HANDLE hDevice, long* FlagStatus);
```

VB
```
Declare Function GetOmsHomeSensors Lib "OmsMAXkMC" (ByVal hDevice As
Long, ByRef FlagStatus As Long) As Long
```

Return Values

SUCCESS	Request was successful
COMMAND_TIME_OUT	Command timed out

Parameters

hDevice	[in] Handle to the controller
FlagStatus	[out] long pointer for home sensor status

Equivalent Command String

None

Remarks

Axis bit assignments:

bit 0	=	X axis	=	OMS_X_AXIS
bit 1	=	Y axis	=	OMS_Y_AXIS
bit 2	=	Z axis	=	OMS_Z_AXIS
bit 3	=	T axis	=	OMS_T_AXIS
bit 4	=	U axis	=	OMS_U_AXIS
bit 5	=	V axis	=	OMS_V_AXIS
bit 6	=	R axis	=	OMS_R_AXIS
bit 7	=	S axis	=	OMS_S_AXIS
bit 8	=	W axis	=	OMS_W_AXIS
bit 9	=	K axis	=	OMS_K_AXIS

Example

See GetOmsLimitSensors

11 ENCODER FEEDBACK SPECIFIC FUNCTIONS

11.01 EnableOmsSlipDetection

Enable slip detection for the selected encoder axis.

C / C++
```
long EnableOmsSlipDetection (HANDLE hDevice, long AxisSelection);
```

VB
```
Declare Function EnableOmsSlipDetection Lib "OmsMAXkMC" (ByVal hDevice
As Long, ByVal AxisSelection As Long) As Long
```

Return Values

SUCCESS	Request was sent
COMMAND_TIME_OUT	Command timed out
INVALID_AXIS_SELECTION	Invalid axis

Parameters

hDevice	[in] Handle to the controller
AxisSelection	[in] Selected Axis

Equivalent Command String

A*IS

***** - axis character corresponding to the value passed in *AxisSelection*.

Remarks

Slip detection is automatically disabled on any of the following conditions:

- On controller power up or reset.
- If any axis is driven into an overtravel limit.
- If slip is detected on any encoder axis.
- Any of the following functions are called:
  - HomeOmsAxisFwdUseEncoder / HomeOmsAxisFwdUseEncoderWait
  - HomeOmsAxisFwdUseSwitch / HomeOmsAxisFwdUseSwitchWait
  - HomeOmsAxisRevUseEncoder / HomeOmsAxisRevUseEncoderWait
  - HomeOmsAxisRevUseSwitch /HomeOmsAxisRevUseSwitchWait
  - SetOmsAxisPosition
  - FindOmsAxisFwdLimit / FindOmsAxisRevLimit
  - Any of the STOP MOTION FUNCTIONS

Example

C / C++

```
//  Enable slip detection on Z axis
lRetVal = EnableOmsSlipDectection ( hDevice, OMS_Z_AXIS);

if ( lRetVal != SUCCESS )  //  Error has occurred
{
  //  Handle any errors
}
```

VB

```
‘ Enable slip detection on Z axis
lRetVal = EnableOmsSlipDectection ( hDevice, OMS_Z_AXIS)

If ( lRetVal <> SUCCESS ) Then  ‘ Error has occurred
  ‘ Handle any errors
End If
```

11.02 GetOmsAxisSlipFlag

Return the state of the selected axis slip flag.

C / C++

```
long GetOmsAxisSlipFlag (HANDLE hDevice, long AxisSelection, long*
FlagStatus);
```

VB

```
Declare Function GetOmsAxisSlipFlag Lib "OmsMAXkMC" (ByVal hDevice As
Long, ByVal AxisSelection As Long, ByRef FlagStatus As Long) As Long
```

Return Values

SUCCESS	Request was sent
COMMAND_TIME_OUT	Command timed out
INVALID_AXIS_SELECTION	Invalid axis

Parameters

hDevice	[in] Handle to the controller
AxisSelection	[in] Selected Axis
FlagStatus	[out] long pointer for slip flag

Equivalent Command String

None

Remarks

None

Example

C / C++

```
lRetVal = GetOmsAxisSlipFlag ( hDevice, OMS_Y_AXIS, &lSlipFlags);
```

VB

```
lRetVal = GetOmsAxisSlipFlag ( hDevice, OMS_Y_AXIS, lSlipFlags)
```

11.03 GetSlipFlags

Same as GetAllOmsSlipFlags

11.04 GetAllOmsSlipFlags

Return the encoder slip flag status of all axes in a long integer.

C / C++

```
long GetAllOmsSlipFlags (HANDLE hDevice, long* FlagStatus);
```

VB

```
Declare Function GetAllOmsSlipFlags Lib "OmsMAXkMC" (ByVal hDevice As
Long, ByRef FlagStatus As Long) As Long
```

Return Values

SUCCESS	Request was sent
COMMAND_TIME_OUT	Command timed out

Parameters

hDevice	[in] Handle to the controller
FlagStatus	[out] long pointer for slip flags

Equivalent Command String

None

Remarks

None

Example

C / C++

```
lRetVal = GetAllOmsSlipFlags ( hDevice, &lSlipFlags);
```

VB

```
lRetVal = GetAllOmsSlipFlags ( hDevice, lSlipFlags)
```

11.05 ClrSlipFlags

Same as ClrOmsSlipFlags

11.06 ClrOmsSlipFlags

Clear the selected or all axes slip flags.

C / C++
```
long ClrOmsSlipFlags (HANDLE hDevice, long AxisSelection);
```

VB
```
Declare Function ClrOmsSlipFlags Lib "OmsMAXkMC" (ByVal hDevice As
Long, ByVal AxisSelection As Long) As Long
```

Return Values

SUCCESS	Command was sent
COMMAND_TIME_OUT	Command timed out
INVALID_AXIS_SELECTION	Invalid axis

Parameters

hDevice	[in] Handle to the controller
AxisSelection	[in] Selected Axis

Equivalent Command String

None

Remarks

None

Example

C / C++

```
// Clear Y and T slip flags
lRetVal = ClrOmsSlipFlags ( hDevice, OMS_Y_AXIS + OMS_T_AXIS);
```

VB

```
‘ Clear Y and T slip flags
lRetVal = ClrOmsSlipFlags ( hDevice, OMS Y AXIS + OMS T AXIS )
```

11.07 GetOmsEncoderFlags

Return the state of the encoder status flags for the selected axis.

C / C++
```
long GetOmsEncoderFlags (HANDLE hDevice, long AxisSelection, long
*FlagsStatus);
```

VB
```
Declare Function GetOmsEncoderFlags Lib "OmsMAXkMC" (ByVal hDevice As
Long, ByVal AxisSelection As Long, ByRef FlagsStatus As Long) As Long
```

Return Values

SUCCESS	Command was sent
RESPONSE_TIME_OUT	Timeout waiting for response from the controller
INVALID_AXIS_SELECTION	Invalid axis

Parameters

hDevice	[in] Handle to the controller
AxisSelection	[in] Selected Axis
FlagStatus	[out] long pointer for axis encoder flags

Equivalent Command String

A*****EA

***** - axis character corresponding to the value passed in *AxisSelection*.

Remarks

The table below describes the status byte configuration:

Bit 0	- Slip Detect Enable	0 – Disabled	1 – Enabled
Bit 1	- Position Maintenance Enable	0 – Disabled	1 – Enabled
Bit 2	- Slip or Stall Detected	0 – Disabled	1 – Enabled
Bit 3	- Within DEADBAND	0 – Disabled	1 – Enabled
Bit 4	- Axis home	0 – Disabled	1 – Enabled

Example

C / C++

```
//  Turn on hold mode to axis X
lRetVal = GetOmsEncoderFlags ( hDevice, OMS_X_AXIS, &lEncFlags);
```

VB

```
‘ Turn on hold mode to axis X
lRetVal = GetOmsEncoderFlags ( hDevice, OMS X AXIS, lEncFlags)
```

11.08 SetOmsAxisPidEnable

Same as SetOmsEncoderHoldMode

11.09 SetOmsEncoderHoldMode

Set the encoder hold mode ON or OFF of a selected axis.

C / C++

```
long SetOmsEncoderHoldMode (HANDLE hDevice, long AxisSelection, long HoldMode);
```

VB

```
Declare Function SetOmsEncoderHoldMode Lib "OmsMAXkMC" (ByVal hDevice As Long, ByVal AxisSelection As Long, ByVal HoldMode As Long) As Long
```

Return Values

SUCCESS	Command was sent
COMMAND_TIME_OUT	Command timed out
INVALID_AXIS_SELECTION	Invalid axis

Parameters

hDevice	[in] Handle to the controller
AxisSelection	[in] Selected Axis
HoldMode	[in] hold mode value

Equivalent Command String

[hold on] `A*HN` or [hold off] `A*HF`

`*` - axis character corresponding to the value passed in *AxisSelection*.

`HN` or `HF` – selected based on *HoldMode* value

Remarks

None

Example

C / C++

```
//  Turn on hold mode to axis X
lRetVal = SetOmsEncoderHoldMode ( hDevice, OMS_X_AXIS, MODE_ON);
```

VB

```
‘ Turn on hold mode to axis X
lRetVal = SetOmsEncoderHoldMode ( hDevice, OMS X AXIS, MODE ON)
```

11.10 SetOmsEncoderRatio

Set the encoder count / motor count ratio of a stepper motor axis that is using encoder feed back.

C / C++

```
long SetOmsEncoderRatio (HANDLE hDevice, long AxisSelection, long
EncoderCount, long MotorCount);
```

VB

```
Declare Function SetOmsEncoderRatio Lib "OmsMAXkMC" (ByVal hDevice As
Long, ByVal AxisSelection As Long, ByVal EncoderCount As Long, ByVal
MotorCount As Long) As Long
```

Return Values

SUCCESS	Request was sent
COMMAND_TIME_OUT	Command timed out
INVALID_AXIS_SELECTION	Invalid axis value
INVALID_PARAMETER	Invalid encoder and / or motor count

Parameters

hDevice	[in] Handle to the controller
AxisSelection	[in] Selected Axis
EncoderCount	[in] Encoder count of the ratio
MotorCount	[in] Motor count of the ratio

Equivalent Command String

A*ER#,##;

***** - axis character corresponding to the value passed in *AxisSelection*.

- *EncoderCount* value

- *MotorCount* value

Remarks

This command was designed for stepper motors with encoders only. The encoder/motor ratio is limited to counts from 1 to 32000.

Example

C / C++

```
//  Set encoder/motor count ratio to axis X
lEncCnt = 3000;
lMotCnt = 4000;

lRetVal = SetOmsEncoderRatio ( hDevice, OMS_X_AXIS, lEncCnt, lMotCnt);
//  Handle any errors

//  Set deadband to 10 on axis X
lDeadband = 10;

lRetVal = SetOmsHoldDeadBand ( hDevice, OMS_X_AXIS, lDeadband);
//  Handle any errors
```

VB

```
‘ Set encoder/motor count ratio to axis X
lEncCnt = 3000
lMotCnt = 4000

lRetVal = SetOmsEncoderRatio ( hDevice, OMS_X_AXIS, lEncCnt, lMotCnt)
‘ Handle any errors

‘ Set deadband to 10 on axis X
lDeadband = 10

lRetVal = SetOmsHoldDeadBand ( hDevice, OMS_X_AXIS, lDeadband)
‘ Handle any errors
```

11.11 SetOmsHoldDeadBand

Set the position hold dead band of a stepper motor axis that is using encoder feed back.

C / C++
```
long SetOmsHoldDeadBand (HANDLE hDevice, long AxisSelection, long DeadBand);
```

VB
```
Declare Function SetOmsHoldDeadBand Lib "OmsMAXkMC" (ByVal hDevice As Long, ByVal AxisSelection As Long, ByVal DeadBand As Long) As Long
```

Return Values

SUCCESS	Request was sent
COMMAND_TIME_OUT	Command timed out
INVALID_AXIS_SELECTION	Invalid axis value
INVALID_PARAMETER	Invalid deadband value

Parameters

hDevice	[in] Handle to the controller
AxisSelection	[in] Selected Axis
DeadBand	[in] Deadband

Equivalent Command String

A*HD#;

***** - axis character corresponding to the value passed in *AxisSelection*.

- *DeadBand* value

Remarks

Deadband is limited to 0 to 64000.

Example

See SetOmsEncoderRatio

11.12 SetOmsEncoderSlipTolerance

Set the encoder slip detection tolerance of a stepper motor axis that is using encoder feed back or to set the maximum allowable following error for a servo axis.

C / C++
```
long SetOmsEncoderSlipTolerance (HANDLE hDevice, long AxisSelection,
long SlipTol);
```

VB
```
Declare Function SetOmsEncoderSlipTolerance Lib "OmsMAXkMC" (ByVal
hDevice As Long, ByVal AxisSelection As Long, ByVal SlipTol As Long) As
Long
```

Return Values

SUCCESS	Request was sent
COMMAND_TIME_OUT	Command timed out
INVALID_AXIS_SELECTION	Invalid axis value
INVALID_PARAMETER	Invalid slip detection tolerance value

Parameters

hDevice	[in] Handle to the controller
AxisSelection	[in] Selected Axis
SlipTol	[in] Tolerance value

Equivalent Command String

A*ES#;

***** - axis character corresponding to the value passed in *AxisSelection*.

- *SlipTol* value

Remarks

The encoder slip tolerance range is limited from 0 to 65535 for stepper axes and from 0 to 327,670 for servo axes.

Example

C / C++

```
//  Set slip tolerance to 5 on axis X
lRetVal = SetOmsEncoderSlipTolerance ( hDevice, OMS_X_AXIS, 5);
```

VB

```
‘ Set slip tolerance to 5 on axis X
lRetVal = SetOmsEncoderSlipTolerance ( hDevice, OMS X AXIS, 5)
```

11.13 SetOmsHoldGain

Set the position hold gain of a stepper motor axis that is using encoder feed back.

C / C++
```
long SetOmsHoldGain (HANDLE hDevice, long AxisSelection, long HoldGain);
```

VB
```
Declare Function SetOmsHoldGain Lib "OmsMAXkMC" (ByVal hDevice As Long, ByVal AxisSelection As Long, ByVal HoldGain As Long) As Long
```

Return Values

SUCCESS	Request was sent
COMMAND_TIME_OUT	Command timed out
INVALID_AXIS_SELECTION	Invalid axis value
INVALID_PARAMETER	Invalid hold gain value

Parameters

hDevice	[in] Handle to the controller
AxisSelection	[in] Selected Axis
HoldGain	[in] Hold gain value

Equivalent Command String

A*HG#;

***** - axis character corresponding to the value passed in *AxisSelection*.

- *HoldGain* value

Remarks

Note the gain is multiplied by position error to determine the correction velocity for correction. The velocity used will not exceed the value set with SetOmsHoldVelocity or command "HV".

Example

C / C++

```
//  Set hold velocity to 10000 on axis Y
lRetVal = SetOmsHoldVelocity ( hDevice, OMS_Y_AXIS, 10000);
//  Handle any errors

//  Set hold gain to 50 on axis X
lRetVal = SetOmsHoldGain ( hDevice, OMS_X_AXIS, 50);
//  Handle any errors
```

VB

```
‘ Set hold velocity to 10000 on axis Y
lRetVal = SetOmsHoldVelocity ( hDevice, OMS_Y_AXIS, 10000)
‘ Handle any errors

‘ Set hold gain to 50 on axis X
lRetVal = SetOmsHoldGain ( hDevice, OMS X AXIS, 50)
```

11.14 SetOmsHoldVelocity

Set the position hold velocity limit of a stepper motor axis that is using encoder feed back.

C / C++
```
long SetOmsHoldVelocity (HANDLE hDevice, long AxisSelection, long HoldVelocity);
```

VB
```
Declare Function SetOmsHoldVelocity Lib "OmsMAXkMC" (ByVal hDevice As Long, ByVal AxisSelection As Long, ByVal HoldVelocity As Long) As Long
```

Return Values

SUCCESS	Request was sent
COMMAND_TIME_OUT	Command timed out
INVALID_AXIS_SELECTION	Invalid axis value
INVALID_PARAMETER	Invalid hold velocity value

Parameters

hDevice	[in] Handle to the controller
AxisSelection	[in] Selected Axis
HoldVelocity	[in] Hold velocity value

Equivalent Command String

A*HV#;

***** - axis character corresponding to the value passed in *AxisSelection*.

- *HoldVelocity* value

Remarks

Hold velocity is the maximum velocity that the controller will use to correct an axis position error.

Example

See SetOmsHoldGain

12 AXIS OVERTRAVEL HANDLING FUNCTIONS

12.01 SetOmsAxisOvertravelDetect

Turn over travel detection for an axis ON or OFF

C / C++
```
long SetOmsAxisOvertravelDetect(HANDLE hDevice, long AxisSelection,
long LimitMode);
```

VB
```
Declare Function SetOmsAxisOvertravelDetect Lib "OmsMAXkMC" (ByVal
hDevice As Long, ByVal AxisSelection As Long, ByVal LimitMode As Long)
As Long
```

Return Values

SUCCESS	Request was sent
COMMAND_TIME_OUT	Command timed out
INVALID_AXIS_SELECTION	Invalid axis value
INVALID_PARAMETER	Invalid limit mode

Parameters

hDevice	[in] Handle to the controller
AxisSelection	[in] Selected Axis
LimitMode	[in] Limit mode

Equivalent Command String

[Limits ON] A***LN;** or [Limits OFF] A***LF;**

***** - axis character corresponding to the value passed in *AxisSelection*.

LN or LF - based on the value passed in for *LimitMode*

Remarks

Turning off limits allows the motor to move beyond the limit switches and should be used with caution.

Example

C / C++

```
lRetVal = SetOmsAxisOvertravelDetect ( hDevice, OMS_X_AXIS, MODE_ON);
```

VB

```
lRetVal = SetOmsAxisOvertravelDetect ( hDevice, OMS_X_AXIS, MODE_ON)
```

12.02 ConfigureOmsAxisLimitInput

Set axis limit logic to active high or active low.

C / C++
```
long ConfigureOmsAxisLimitInput (HANDLE hDevice, long AxisSelection,
long LimitState);
```

VB
```
Declare Function ConfigureOmsAxisLimitInput Lib "OmsMAXkMC" (ByVal
hDevice As Long, ByVal AxisSelection As Long, ByVal LimitState As Long)
As Long
```

Return Values

SUCCESS	Command was sent
COMMAND_TIME_OUT	Command timed out
INVALID_AXIS_SELECTION	Invalid axis value
INVALID_PARAMETER	Invalid limit state

Parameters

hDevice	[in] Handle to the controller
AxisSelection	[in] Selected Axis
LimitState	[in] Limit state

Equivalent Command String

[Limits HIGH] A***LH;** or [Limits LOW] A***LL;**

***** - axis character corresponding to the value passed in *AxisSelection*.

LH or LL - based on the value passed in for *LimitState*

Remarks

None

Example

None

12.03 SetOmsSoftLimitsMode

Turns ON/OFF the soft limit handling mode of the selected axis.

C / C++
```
long SetOmsSoftLimitsMode (HANDLE hDevice, long AxisSelection, long SoftLimit);
```

VB
```
Declare Function SetOmsSoftLimitsMode Lib "OmsMAXkMC" (ByVal hDevice As Long, ByVal AxisSelection As Long, ByVal SoftLimit As Long) As Long
```

Return Values

SUCCESS	Command was sent
COMMAND_TIME_OUT	Command timed out
INVALID_AXIS_SELECTION	Invalid axis value
INVALID_PARAMETER	Invalid limit state

Parameters

hDevice	[in] Handle to the controller
AxisSelection	[in] Selected Axis
SoftLimit	[in] Soft limit state

Equivalent Command String

[Soft Limits ON] A***SL;** or [Soft Limits OFF] A***SF;**

***** - axis character corresponding to the value passed in *AxisSelection*.

SL or SF - based on the value passed in for *SoftLimit*

Remarks

When soft limit handling is **ON**, the axis will use the deceleration ramp to stop when an overtravel limit switch is activated. If soft limit mode is **OFF** then the axis axes will stop immediately with out using a deceleration ramp.

Example

C / C++

```
//  Set X axis soft limit on
lRetVal = SetOmsSoftLimitsMode ( hDevice, OMS_X_AXIS, MODE_ON);
```

VB

```
‘ Set X axis soft limit on
lRetVal = SetOmsSoftLimitsMode ( hDevice, OMS X AXIS, MODE ON)
```

13 GENERAL PURPOSE I/O BIT FUNCTIONS

13.01 GetOmsIOBitConfig

Get the configuration of all of the general purpose input and output bits.

C / C++
```
long GetOmsIOBitConfig (HANDLE hDevice, long* pBitConfig);
```

VB
```
Declare Function GetOmsIOBitConfig Lib "OmsMAXkMC" (ByVal hDevice As
Long, ByRef pBitConfig As Long) As Long
```

Return Values

SUCCESS	Query command was sent
COMMAND_TIME_OUT	Command timed out

Parameters

hDevice	[in] Handle to the controller
pBitConfig	[out] long pointer for bits configuration

Equivalent Command String

?BD

Remarks

Return value for each bit in *pBitConfig* are:
- 0 indicates the bit is input
- 1 indicates the bit is output

Example

None

13.02 ConfigureAllOmsIOBits

Configure all the general purpose input and output bits directions.

C / C++
```
long ConfigureAllOmsIOBits (HANDLE hDevice, long BitConfig);
```

VB
```
Declare Function ConfigureAllOmsIOBits Lib "OmsMAXkMC" (ByVal hDevice
As Long, ByVal BitConfig As Long) As Long
```

Return Values

SUCCESS	Command was sent
COMMAND_TIME_OUT	Command timed out
INVALID_PARAMETER	Invalid bit configuration

Parameters

hDevice	[in] Handle to the controller
BitConfig	[in] long pointer for bits configuration

Equivalent Command String

BD#;
- # - *BitConfig* value

Remarks

Return value for each bit in *BitConfig* are:
- 0 sets the bit to input
- 1 sets the bit to output

Example

None

13.03 SetAllOmsIOBits

Configuration of all of the GPIO output bits to a known state.

C / C++
```
long SetAllOmsIOBits (HANDLE hDevice, long BitStates);
```

VB
```
Declare Function SetAllOmsIOBits Lib "OmsMAXkMC" (ByVal hDevice As
Long, ByVal BitStates As Long) As Long
```

Return Values

SUCCESS	Command was sent
COMMAND_TIME_OUT	Command timed out

Parameters

hDevice	[in] Handle to the controller
BitState	[in] output bit states configuration

Equivalent Command String

BS#;
#-BitState value

Remarks

Return value for each bit in *BitState* are:
- 0 indicates the bit is input
- 1 indicates the bit is output

Example

C / C++

```
//  Set output bits 0-3 to HIGH
lBitsDirection = 0x007;
lRetVal = SetAllOmsIOBits ( hDevice, lBitsState);

//  Handle any errors

//  Check if bits direction was set
lRetVal = GetAllOmsIOBits ( hDevice, &lBits);

//  Handle any errors
```

VB

```
‘ Set output bits 0-3 to HIGH
lBitsDirection = 0x007
lRetVal = SetAllOmsIOBits ( hDevice, lBitsState)

‘ Handle any errors

‘ Check if bits direction was set
lRetVal = GetAllOmsIOBits ( hDevice, lBits)

‘ Handle any errors

If( lBits == lBitsState) Then
  ‘ Bits states configured is OK
End If
```

13.04 GetAllOmsIOBits

Get the TTL states of all the general purpose input / output bits.

C / C++
```
long GetAllOmsIOBits (HANDLE hDevice, long* pBitStates);
```

VB
```
Declare Function GetAllOmsIOBits Lib "OmsMAXkMC" (ByVal hDevice As
Long, ByRef pBitStates As Long) As Long
```

Return Values

SUCCESS	Query command was sent
COMMAND_TIME_OUT	Command timed out

Parameters

hDevice	[in] Handle to the controller
pBitStates	[out] long pointer for GPIO TTL bits state

Equivalent Command String

BX

Remarks

Return value for each bit in *pBitStates* are:
- 0 indicates the bit is at a TTL low level.
- 1 indicates the bit is at a TTL high level.

Example

See SetAllOmsIOBits

13.05 GetOmsIOBit

Get the TTL state of a general purpose input output bit.

C / C++
```
long GetOmsIOBit (HANDLE hDevice, long lBitNum, long* BitState);
```

VB
```
Declare Function GetOmsIOBit Lib "OmsMAXkMC" (ByVal hDevice As Long,
ByVal lBitNum As Long, ByRef BitState As Long) As Long
```

Return Values

SUCCESS	Query command was sent
COMMAND_TIME_OUT	Command timed out
INVALID_BIT_NUMBER	Invalid bit number

Parameters

hDevice	[in] Handle to the controller
lBitNum	[in] GPIO bit number
BitState	[out] long pointer for GPIO TTL bit state

Equivalent Command String

BX (The response is filtered to return the value for the bit specified.)

Remarks

Valid bit numbers for *lBitNum* are between 0 and 15. Return value for *BitState* are:
- 0 indicates the bit is at a TTL low level.
- 1 indicates the bit is at a TTL high level.

Example

C / C++

```
// Set bit I/O output bit 4 state to high
lRetVal = SetOmsIOBit ( hDevice, 4, HIGH);

//  Handle any errors

//  check if bit four was set to output
lRetVal = GetOmsIOBit ( hDevice, 4, &lBitConfig);

//  Handle any errors

if( lBitState == 1)
  //  GPIO bit 4 is high
else
  //  GPIO bit 4 is low
```

VB

```
‘ Set bit I/O output bit 4 state to high
lRetVal = SetOmsIOBit ( hDevice, 4, HIGH)

‘ Handle any errors

lRetVal = GetOmsIOBit ( hDevice, 4, lBitState)

‘ Handle any errors

If ( lBitState = 1 ) Then
  ‘ GPIO bit 4 is high
Else
  ‘ GPIO bit 4 is low
End If
```

13.06 SetOmsIOBit

Set the state of a general purpose output bit.

C / C++
```
long SetOmsIOBit (HANDLE hDevice, long lBitNum, long BitState);
```

VB
```
Declare Function SetOmsIOBit Lib "OmsMAXkMC" (ByVal hDevice As Long,
ByVal lBitNum As Long, ByVal BitState As Long) As Long
```

Return Values

SUCCESS	Command was sent
COMMAND_TIME_OUT	Command timed out

Parameters

hDevice	[in] Handle to the controller
lBitNum	[in] GPIO bit number
BitState	[in] bit state, 1 = HIGH, 0 = LOW

Equivalent Command String

[bit HIGH] `BH#;` or [bit LOW] `BL#;`
`#` - *`BitState`* value
`BH` or `BL` – based on the value passed in for *`BitState`*.

Remarks

The direction of the bit must be set as an output bit in order to change it's state. Valid bit numbers include all output bits in the range of 0 to 15.

Example

Set GetOmsIOBit

14 AXIS AUXILIARY OUTPUT BIT FUNCTIONS

14.01 EnableOmsAxisAuxOutAutoMode

Sets the axis auxiliary output on or off at the beginning of each move.

C / C++
```
long EnableOmsAxisAuxOutAutoMode (HANDLE hDevice, long AxisSelection,
long MotionState);
```

VB
```
Declare Function EnableOmsAxisAuxOutAutoMode Lib "OmsMAXkMC" (ByVal
hDevice As Long, ByVal AxisSelection As Long, ByVal MotionState As
Long) As Long
```

Return Values

SUCCESS	Command was sent
COMMAND_TIME_OUT	Command timed out
INVALID_AXIS_SELECTION	Invalid axis value
INVALID_PARAMETER	Invalid motion state value

Parameters

hDevice	[in] Handle to the controller
AxisSelection	[in] Selected Axis
MotionState	[in] Switch state , HIGH or LOW

Equivalent Command String

[power automatic ON] `A*PA0;` or [power automatic OFF] `A*PA1;`

`*` - axis character corresponding to the value passed in *AxisSelection*.

`PA0;` or `PA1;` – based on the value passed in for *MotionState*.

Remarks

The Auxiliary output is set HIGH or LOW at the beginning of a move command or GO/GD and complement output after the move has executed. This automatic mode is disabled if the board is reset or the state of the auxiliary output bit is set via SetOmsAxisAuxOutBit function.

Valid auxiliary modes:

- LOW indicates the auxiliary output is to be set to a TTL low level during the axis move or each GO or GD command execution.

- HIGH indicates the auxiliary output is to be set to a TTL high level during the axis move or each GO or GD command execution.

Example

C / C++

```
//  Set Z axis to active low, set aux. output to low during a move
lRetVal = EnableOmsAxisAuxOutAutoMode ( hDevice, OMS_Z_AXIS, LOW);
//  Handle any errors

//  Set Z axis settle time to 500 milliseconds
lSettleTime = 500;
lRetVal = SetOmsAxisAuxOutSettleTime ( hDevice, OMS_Z_AXIS, lSettleTime);
//  Handle any errors
```

VB

```
‘ Set Z axis to active low, set aux. output to low during a move
lRetVal = EnableOmsAxisAuxOutAutoMode ( hDevice, OMS_Z_AXIS, LOW)
‘ Handle any errors

 ‘ Set Z axis settle time to 500 milliseconds
lSettleTime = 500
lRetVal = SetOmsAxisAuxOutSettleTime ( hDevice, OMS_Z_AXIS, lSettleTime)
‘ Handle any errors
```

14.02 SetOmsAxisAuxOutSettleTime

Settling time of an axis auxiliary output bit after execution of an axis move.

C / C++
```
long SetOmsAxisAuxOutSettleTime (HANDLE hDevice, long AxisSelection,
long SettleTime));
```

VB
```
Declare Function SetOmsAxisAuxOutSettleTime Lib "OmsMAXkMC" (ByVal
hDevice As Long, ByVal AxisSelection As Long, ByVal SettleTime As Long)
As Long
```

Return Values

SUCCESS	Command was sent
COMMAND_TIME_OUT	Command timed out
INVALID_AXIS_SELECTION	Invalid axis value
INVALID_PARAMETER	Invalid settle time

Parameters

hDevice	[in] Handle to the controller
AxisSelection	[in] Selected Axis
SettleTime	[in] Auxiliary bit settling time in milliseconds

Equivalent Command String

A*SE#;

***** - axis character corresponding to the value passed in *AxisSelection*.

 # – SettleTime value

Remarks

Settling range is between 0 to 1000 milliseconds. This is the number of milliseconds that the bit is left active after the axis has finished a move.

Example

See EnableOmsAxisAuxOutAutoMode

14.03 SetOmsAxisAuxOutBit

Sets the selected axis' auxiliary output bit to HIGH or LOW.

C / C++
```
long SetOmsAxisAuxOutBit (HANDLE hDevice, long AxisSelection, long
BitState);
```

VB
```
Declare Function SetOmsAxisAuxOutBit Lib "OmsMAXkMC" (ByVal hDevice As
Long, ByVal AxisSelection As Long, ByVal BitState As Long) As Long
```

Return Values

SUCCESS	Command was sent
COMMAND_TIME_OUT	Command timed out
INVALID_AXIS_SELECTION	Invalid axis value
INVALID_PARAMETER	Invalid bit state

Parameters

hDevice	[in] Handle to the controller
AxisSelection	[in] Selected Axis
BitState	[in] Auxiliary state, MODE_ON or MODE_OFF

Equivalent Command String

[auxiliary ON] A***AN;** or [auxiliary OFF] A***AF;**

***** - axis character corresponding to the value passed in *AxisSelection*.

AN or AF – based on the value of *BitState*

Remarks

Note calling this function **disables** the automatic activation of the axis auxiliary bit when the axis is in motion (see EnableOmsAxisAuxOutAutoMode).

Valid bit states:

- MODE_OFF auxiliary output is to be set to a TTL low level.
- MODE_ON auxiliary output is to be set to a TTL high level.

Example

C / C++

```
//  Set Y axis auxiliary to high
lRetVal = SetOmsAxisAuxOutBit ( hDevice, OMS_Y_AXIS, MODE_ON);
```

VB

```
‘ Set Y axis auxiliary to high
lRetVal = SetOmsAxisAuxOutBit ( hDevice, OMS Y AXIS, MODE ON)
```

14.04 SetSelectedOmsAuxOutBits

Sets the selected auxiliary output bits to a known state.

C / C++
```
long SetSelectedOmsAuxOutBits (HANDLE hDevice, long AxisSelection, long BitState);
```

VB
```
Declare Function SetSelectedOmsAuxOutBits Lib "OmsMAXkMC" (ByVal hDevice As Long, ByVal AxisSelection As Long, ByVal BitState As Long) As Long
```

Return Values

SUCCESS	Command was sent
COMMAND_TIME_OUT	Command timed out
INVALID_AXIS_SELECTION	Invalid axis / axes selection
INVALID_PARAMETER	Invalid bit state

Parameters

hDevice	[in] Handle to the controller
AxisSelection	[in] Selected Axis
BitState	[in] Bit state, BIT_HIGH or BIT_LOW

Equivalent Command String

[auxiliary ON] A***AN#,#,…#;*** or [auxiliary OFF] A***AF#,#,…#;***

******* - axis character corresponding to the value passed in *AxisSelection*.

AN or AF – based on the value of *BitState*

#,#,…# – values from *BitState*

Remarks

None

Example

C / C++

```
//  Set X, Z, V axes to high
lSelAxes = OMS_X_AXIS + OMS_Z_AXIS + OMS_V_AXIS;
lRetVal = SetSelectedOmsAuxOutBits ( hDevice, lSelAxes, HIGH);
```

VB

```
‘  Set X, Z, V axes to high
lSelAxes = OMS_X_AXIS + OMS_Z_AXIS + OMS_V_AXIS
lRetVal = SetSelectedOmsAuxOutBit ( hDevice, lSelAxes, HIGH)
```

15 TIME DELAY FUNCTION

15.01 OmsWait

Block the thread for the specified number of milliseconds.

C / C++
```
long OmsWait(long WaitTime);
```

VB
```
Declare Function OmsWait Lib "OmsMAXkMC" (ByVal WaitTime As Long) As Long
```

Return Values

SUCCESS	Command was sent
INVALID_PARAMETER	Invalid wait time parameter

Parameter

WaitTime	[in] Wait time in milliseconds

Equivalent Command String

None

Remarks

This function calls the Microsoft SDK **`Sleep()`** function directly. Time resolution depends on the PC's timer tick resolution, typically about +/- 10 milliseconds.

Example

C / C++

```
lRetVal = OmsWait( 5000 ); //  wait for 5 seconds
//  handle any errors
```

VB

```
lRetVal = OmsWait( 5000 ) ‘ wait for 5 seconds
 ‘ Handle any errors
```

16 BIG BUFFER FUNCTIONS

Big buffer allows large buffer arrays of commands be sent to the controller in one call. Where as before normal buffer mode has the limit of 1024 characters big buffer is user defined up to 500,000 characters in size. Big buffer was created to be used with Variable Velocity Commands. Although it can be used like the normal command buffer it would be over kill and recommended that it only be used for Variable Velocity Commands.

16.01 OMS_SetCommandBuffer

Select between the Normal or size specified Big Command Buffer to send commands.

C / C++

```
long OMS_SetCommandBuffer(HANDLE hDevice, eOMSBuffMode eBuffSelection,
long lBuffSize);
```

VB

```
Declare Function OMS_SetCommandBuffer Lib "OmsMAXkMC" (ByVal hDevice As
Long, ByVal eBuffSelection As eOMSBuffMode, ByVal lBuffSize As Long) As
Long
```

Return Values

SUCCESS	Command was sent
COMMAND_TIME_OUT	Command timed out
INVALID_PARAMETER	Invalid buffer selection or buffer size

Parameters

hDevice	[in] Handle to the controller
eBuffSelection	[in] Buffer selection: OMS_BIG_CMD_BUFFER_MODE OMS_NORMAL_CMD_BUFFER_MODE
lBuffSize	[in] Size of command buffer to allocate

Equivalent Command String

[big buffer] `AX;#BB#;` or [normal buffer] `AX;#BN;`
`# (bold)` – buffer size in *lBuffSize*

Remarks

This setting is not preserved in non-volatile memory so if the controller experiences any loss of power or reset via ResetOmsController or "RS" command the setting will need to be re-sent.

Calling this function puts the controller in signal axis mode defaulting to the X axis. If specifying OMS_BIG_CMD_BUFFER_MODE, *lBuffSize* can be any value between 1024 and 500,000 or 0 for the default size of 500,000. Unlike OMS_SwitchCommandBuffer, this function will flush the buffer so any commands pooled in the Big Command Buffer will get cleared.

Example

C / C++

```
lRetVal = OMS_SetCommandBuffer ( hDevice, OMS_BIG_CMD_BUFFER_MODE, 35000 );
//  handle any errors
```

VB

```
lRetVal = OMS_SetCommandBuffer ( hDevice, OMS_BIG_CMD_BUFFER_MODE, 35000 )
 ‘ Handle any errors
```

16.02 OMS_GetCommandBufferSelection

Get the identity of the current command buffer

C / C++

```
long OMS_GetCommandBufferSelection(HANDLE hDevice, eOMSBuffMode
*pBuffSelection);
```

VB

```
Declare Function OMS_GetCommandBufferSelection Lib "OmsMAXkMC" (ByVal
hDevice As Long, ByRef pBuffSelection As eOMSBuffMode) As Long
```

Return Values

SUCCESS	Query command was successful
COMMAND_TIME_OUT	Command timed out

Parameters

hDevice	[in] Handle to the controller
pBuffSelection	[out] enum pointer for buffer mode

Equivalent Command String

None

Remarks

None

Example

C / C++

```
lRetVal = OMS_GetCommandBufferSelection ( hDevice, pBuffMode );

if ( lRetVal != SUCCESS )  //  Error has occurred
  //  Handle any errors
else if( pBuffMode == OMS_BIG_CMD_BUFFER_MODE)
  //  Axis is in big buffer mode
```

VB

```
lRetVal = OMS_GetCommandBufferSelection ( hDevice, pBuffMode )

If ( lRetVal <> SUCCESS ) Then  ‘ Error has occurred
  ‘ Handle any errors
Else If ( pBuffMode = OMS_BIG_CMD_BUFFER_MODE)
  ‘ Axis is in big buffer mode
End If
```

16.03 OMS_BigBufferSendBlock

Send a block of commands to the big command buffer.

C / C++

```
long OMS_BigBufferSendBlock(HANDLE hDevice, LPSTR pCmd, long
lBlockSize, long *lNumSent);
```

VB

```
Declare Function OMS_BigBufferSendBlock Lib "OmsMAXkMC" (ByVal hDevice
As Long, ByVal pCmd As String, ByVal lBlockSize As Long, ByRef lNumSent
As Long) As Long
```

Return Values

SUCCESS	Buffer was sent
COMMAND_TIME_OUT	Command timed out

Parameters

hDevice	[in] Handle to the controller
pCmd	[in] Pointer to character string array buffer
lBlockSize	[in] Size of the character string array buffer
lNumSent	[in] Number of characters in the big buffer

Equivalent Command String

None

Remarks

This command can be used while in normal command buffer mode. Commands sent will be pooled up before the switch over to big buffer mode.

OMS_COMMAND_TIME_OUT is returned if character buffer exceeds the amount of free space in big command buffer. Use OMS_GetBigBufferFreeSpace to check the total of free space available.

Example

None

16.04 OMS_GetBigBufferFreeSpace

Get the number of free character positions in the big command buffer.

C / C++

```
long OMS_GetBigBufferFreeSpace(HANDLE hDevice, long *pFree);
```

VB

```
Declare Function OMS_GetBigBufferFreeSpace Lib "OmsMAXkMC" (ByVal
hDevice As Long, ByRef pFree As Long) As Long
```

Return Values

SUCCESS	Query command was successful
COMMAND_TIME_OUT	Command timed out

Parameters

hDevice	[in] Handle to the controller
pFree	[out] long pointer for free space value

Equivalent Command String

#BQ

Remarks

The return value is the buffer size starting at zero. (i.e. free buffer size of 500,000 returns 499,999.) This function accesses shared memory on the controller and has no impact on the command queue.

Example

C / C++

```
lRetVal = OMS_GetBigBufferFreeSpace( hDevice, &lBuffFree );
//  Handle any errors

lRetVal = OMS_FlushBigBuffer( hDevice );
//  Handle any errors
```

VB

```
lRetVal = OMS_GetBigBufferFreeSpace( hDevice, lBuffFree )
‘ Handle any errors

lRetVal = OMS_FlushBigBuffer( hDevice )
‘ Handle any errors
```

16.05 OMS_FlushBigBuffer

Flush the big command buffer.

C / C++
```
long OMS_FlushBigBuffer(HANDLE hDevice);
```

VB
```
Declare Function OMS_FlushBigBuffer Lib "OmsMAXkMC" (ByVal hDevice As Long) As Long
```

Return Values

SUCCESS	Command was sent
COMMAND_TIME_OUT	Command timed out

Parameter

hDevice	[in] Handle to the controller

Equivalent Command String

#BF

Remarks

Calling this function puts the controller in single axis mode defaulting to the X axis

Example

See OMS_GetBigBufferFreeSpace

Appendix A Define Data Types

```
#define SUCCESS                0
#define COMMAND_TIME_OUT       1
#define RESPONSE_TIME_OUT      2
#define INVALID_AXIS_SELECTION 3
#define MOVE_TIME_OUT          4
#define INVALID_PARAMETER      5
#define INVALID_BIT_NUMBER     6

//  Vector mode constants.
#define INVALID_VECTOR_COMMAND         7
#define INVALID_VELOCITY_CONTROL_COUNT 8
#define INVALID_BIT_STATE_MASK         9
#define INVALID_VECTOR_QUEUE_SELECT    10
#define ALREADY_IN_VECTOR_MODE         11
#define QUEUE_FULL                     12
#define BUFFER_FULL                    13
#define INVALID_MOVE_TYPE              14

//  Axis status definitions
#define AXIS_DIRECTION   0x01
#define AXIS_DONE        0x02
#define AXIS_OVERTRAVEL  0x04
#define AXIS_HOME_SWITCH 0x08

//  Encoder status definitions
#define SLIP_DETECT_ENABLED           0x01
#define POSITION_MAINTENANCE_ENABLED  0x02
#define AXIS_SLIPPED                  0x04
#define AXIS_WITHIN_DEADBAND          0x08
#define ENCODER_AT_HOME               0x10

#define POSITIVE  0
#define NEGATIVE  1
#define MODE_OFF  0
#define MODE_ON   1
#define LOW       0
#define HIGH      1

//  Controller status definitions
#define COMMAND_ERROR           0x01

//  Axis selection codes
#define OMS_X_AXIS 0x01
#define OMS_Y_AXIS 0x02
#define OMS_Z_AXIS 0x04
#define OMS_T_AXIS 0x08
#define OMS_U_AXIS 0x10
#define OMS_V_AXIS 0x20
#define OMS_R_AXIS 0x40
#define OMS_S_AXIS 0x80
#define OMS_W_AXIS 0x100
#define OMS_K_AXIS 0x200
#define OMS_ALL_AXES 0x03FF
```

```
//  I/O bit selection codes
#define BIT0    0x0001
#define BIT1    0x0002
#define BIT2    0x0004
#define BIT3    0x0008
#define BIT4    0x0010
#define BIT5    0x0020
#define BIT6    0x0040
#define BIT7    0x0080
#define BIT8    0x0100
#define BIT9    0x0200
#define BIT10   0x0400
#define BIT11   0x0800
#define BIT12   0x1000
#define BIT13   0x2000
#define BIT14   0x4000
#define BIT15   0x8000

#define MIN_VELOCITY 1
#define MAX_VELOCITY 4000000

#define MIN_ACCELERATION 1
#define MAX_ACCELERATION 8000000
```

Appendix B Structure Data Types

```
typedef struct
{
  long X;
  long Y;
  long Z;
  long T;
  long U;
  long V;
  long R;
  long S;
  long W;
  long K;
} AXES_DATA, *PAXES_DATA;

typedef struct
{
   long Motor;
   long Encoder;
} AXIS_DATA;

typedef struct
{
  OMS_AXIS_DATA X;
  OMS_AXIS_DATA Y;
  OMS_AXIS_DATA Z;
  OMS_AXIS_DATA T;
  OMS_AXIS_DATA U;
  OMS_AXIS_DATA V;
  OMS_AXIS_DATA R;
  OMS_AXIS_DATA S;
  OMS_AXIS_DATA W;
  OMS_AXIS_DATA K;
} POSITION_DATA, *PPOSITION_DATA;
```

Appendix C Enumerated Data Types

```
typedef enum
{
  OMS_BIG_CMD_BUFFER_MODE = 0,
  OMS_NORMAL_CMD_BUFFER_MODE = 1
} eOMSBuffMode;
```