ACR-MotionMax API Users Guide

IMPORTANT

User Information

Warning!

ACR Series products are used to control electrical and mechanical components of motion control systems. You should test your motion system for safety under all potential conditions. Failure to do so can result in damage to equipment and/or serious injury to personnel.

Technical Assistance

Contact your local automation technology center (ATC) or distributor.

North America and Asia
Parker Hannifin
5500 Business Park Drive
Rohnert Park, CA 94928
Telephone: (800) 358-9070 or (707) 584-7558
Fax: (707) 584-3793
Email: emn_support@parker.com
Internet: http://www.parkermotion.com

Germany, Austria, Switzerland
Parker Hannifin
Postfach: 77607-1720
Robert-Bosch-Str. 22
D-77656 Offenburg
Telephone: +49 (0) 781 509-0
Fax: +49 (0) 781 509-176
Email: sales.hauser@parker.com

Europe (non-German speaking)
Parker Hannifin
21 Balena Close
Poole, Dorset
England BH17 7DX
Telephone: +44 (0) 1202 69 9000
Fax: +44 (0) 1202 69 5750
Email: EMDTech.Help@parker.com

Italy
Parker Hannifin
20092 Cinisello Balsamo
Milan, Italy via Gounod, 1
Telephone: +49 (0) 781 509-0
Fax: +49 (0) 781 509-176
Email: sales.sbc@parker.com

Parker Automation

Technical Support E-mail
emn_support@parker.com

Contents

Getting Started

Introduction

This Guide give basic instructions on the functionality and implementation of the MotionCore.DLL API. Only experience developers of motion control software should attempt to use this API. A working knowledge of the Parker/Acroloop AcroBasic language is also required to full implement a finished motion system.

Operating System

This 32 bit DLL is for use in MS Windows NT 4.0 (service pack 4.0 or above) , 2000 or XP Pro

Installation

Parker-Hannifin API Installation will prompt you to create a folder…C:\Program Files\Parker\ACR-MotionMax. All references in this document will assume the users accepted the default folder.

When the Installation is complete, a prompt to reboot the computer will be displayed. In order for all of the drivers to be properly activated, the computer must be rebooted before any calls to the MotionCore.DLL can be placed.

Installation Folders and Files

The Installation will generate the Axium Folder and several sub-folders.

Docs – this folder contains this document and others

ParFiles – this folder cantains all of the storage files incuding the configuration files

Samples – this folder contains a Visual C++ and a Visual Basic sample.

The MotionCore.DLL file was installed in the Windows\System32 folder. This is done so it is accessible from anywhere on the computer. Since the System32 folder is found in the Windows path, you do need to reference any drive and folder information to locate it.

The ACR-MOTIONMAX.INI files is alos located in the Windows folder. This file contains the location of any files you create as well as the file location of the configuration files. This file can be edited with any text editor and cahnged to accommodate you specific application. The files looks like the following…

[PATHS]

Root=C:\Program Files\Parker\ACR-MotionMax

Config=C:\Program Files\ Parker\ACR-MotionMax\ParFiles

API Command Groups

Command Groups

DLL Initialization

AX_InitializeControl	Start DLL … this must be the first call to the DLL
AX_IsCardPresent	Checks for Servo Controller Presence

Parameters

AX_UpdateParams	Force DLL to Reload All Parameters from Parameters.Cfg File
AX_DownloadParameters2Card	Force DLL to Update Params from Parameters.Cfg file to ServoCard

Feedback & Status

AX_GetEncoderCounts	Extract Encoder information from DLL in array
AX_GetDistanceToGoCoordinates	Extract DISTANCE TO GO information from DLL into array
AX_GetAxisVelocity	Extract Master velocity from DLL

Bit Functions

AX_BitOn	Turns on any bit (Except Inputs)
AX_BitOff	Turns on any bit

Bit Group Functions

AX_GetInputs	Extracts all 96 Inputs
AX_GetOutputs	Extracts all 96 Outputs
AX_GetControlBits	Extracts BITS [128-255] ACR-MOTIONMAX Control Signals
AX_GetMasterFlags	Extracts BITS [512-543] Master Flags
AX_GetMcodeBits	Extracts BITS [1920-2047] ACR-MOTIONMAX MCode Bits
AX_GetAdcInputs	Extracts ADC Channels (0-7) into Array

G-Code File Functions

AX_LoadGCodeFile	Send DLL a G-Code Filename to load in the Motion List
AX_GetGCodeLineNumber	Extract Current Gcode Line Number being executed in cycle

Jogging & Homing

AX_JogMinus	JOG SELECTED AXIS MINUS DIRECTION
AX_JogPlus	JOG SELECTED AXIS PLUS DIRECTION
AX_JogMinusFast	JOG SELECTED AXIS MINUS DIRECTION at Fast Rate
AX_JogPlusFast	JOG SELECTED AXIS PLUS DIRECTION at Fast Rate
AX_JogStop	STOP ALL JOGGING
AX_HomeAxis	HOME THE SELECTED AXIS

MDI Mode Commands

AX_StartMDI	Start MDI Mode Signal to DLL
AX_StopMDI	Stop MDI Mode Signal to DLL
AX_ExecuteMDI	Execute MDI command String to DLL

Virtual Overrides

AX_SetFov	Set Feedrate override manually with sliders
AX_SetRov	Set Rapid rate override manually with sliders
AX_SetSov	Set Spindle override manually with sliders

Graphics

AX_InitializeGPH	Initialize Graphics Engine & pass handle of the control area
AX_SetGphView	Tell Graphics which ViewPort to plot
AX_ZoomWindow	Pass Zoom Coordinates (in Pixies) to the Graphics Engine
AX_SetGraphics	Turn Graphics Generator on/off

Error Message

AX_SystemErr	Get the System errors from the DLL if any
AX_ClearSystemErr	Acknowledges & Clears the Error from the DLL

Cycle Functions

AX_CycleStart	Starts Loaded G Code File Running
Feedhold	Stop the Program running until Cleared
Reset	Stop Current programs and effects a M30
Auto/Step	Set this bit (133) to tell DLL to Cycle in Auto Clear this bit to run in Step

API Command Descriptions

(DLL Initialization)

AX_InitializeControl

Description Sends Handle (. hwnd) of the Applications Main Form to the DLL to spawn its threads.
Returns (Integer) 1 on Success, 0 on Failure
Arguments (Long) handle of the Applications Main Form

VB Example:

```
Private Sub frmForm_Initialize ()
 Call AX_InitializeControl (frmMain.hwnd)
End sub
```

Declaration: Public Declare Function AX_InitializeControl Lib "MotionCore.DLL" (ByVal ProcHwnd As Long) As Integer

AX_IsCardPresent

Description Checks if Servo Controller Card is present
Returns 1 if Motion card exists, 0 if not.
Arguments None

Declaration
Public Declare Function AX_IsCardPresent Lib "MotionCore.DLL" () As Integer

VB Example:

```
Private Sub frmMain_Initialize ()
 RetVal = Call AX_IsCardPresent ()
  If RetVal = 0 then
    Offline = True ‘MotionCard not found
  Else
    Offline = False ‘ MotionCard Found
  End if
End Sub
```

(Parameter Functions)

AX_UpdateParams

Description Force DLL to Reload All Parameters from Parameters.Cfg File
Returns (Integer) 1 on Success, 0 on Failure
Arguments None

VB Example:
Private Sub UpdateParamsBtn_Click ()
‘ Force DLL to Reload All Parameters from Parameters.Cfg File
Call AX_UpdateParams
Sleep (10)
‘Force DLL to Update Params from Parameters.Cfg file to ServoCard
Call AX_DownloadParameters2Card
Sleep (10)

Declaration: Public Declare Function AX_UpdateParams Lib _
"MotionCore.DLL" () As Integer

AX_DownloadParameters2Card

Description Force DLL to Update Params from Parameters.Cfg file to ServoCard
Returns (Integer) 1 on Success, 0 on Failure
Arguments None

VB Example:
Private Sub UpdateParamsBtn_Click ()
‘ Force DLL to Reload All Parameters from Parameters.Cfg File
Call AX_UpdateParams
Sleep (10)
‘Force DLL to Update Params from Parameters.Cfg file to ServoCard
Call AX_DownloadParameters2Card
Sleep (10)

Declaration: Declare Function AX_DownloadParameters2Card Lib _
"MotionCore.DLL" () As Integer

(Feedback & Status Functions)

AX_GetEncoderCounts

Description Extract Encoder information from DLL in array
Returns (Integer) 1 on Success, 0 on Failure
Arguments Global EncoderArray (0 To 8) As Long

VB Example:
Private Sub IOTIMER_Timer ()
Ret = AX_GetEncoderCounts(EncoderArray(0)) 'Get ABS Encoder Pos array

```
AbsReadout (0).Caption = Format(((EncoderArray(0) * Val(Resolution0)) - WorkOffsetsArray(0)), "00.0000")
AbsReadout(1).Caption = Format(((EncoderArray(1) * Val(Resolution1)) - WorkOffsetsArray(1)), "00.0000")
AbsReadout(2).Caption = Format(((EncoderArray(2) * Val(Resolution2)) –
WorkOffsetsArray(2) - CurLength), "00.0000")
AbsReadout(3).Caption = Format(((EncoderArray(3) * Val(Resolution3)) - WorkOffsetsArray(3)), "000.000")
End sub
```

Declaration: Public Declare Function AX_GetEncoderCounts Lib "MotionCore.DLL" _
(ByRef EncoderArray as Long) As Integer

AX_GetDistanceToGoCoordinates

Description Extract Distance to Go information from DLL in array
Returns (Integer) 1 on Success, 0 on Failure
Arguments Global DTGArray (0 To 8) As Long

VB Example: Ret = AX_GetDistanceToGoCoordinates (DTGArray(0))

Declaration: Public Declare Function AX_GetDistanceToGoCoordinates Lib _
"MotionCore.DLL" (ByRef lDTGArray As Long) As Integer

AX_GetAxisVelocity

Description Extract Master velocity from DLL
Returns (Integer) 1 on Success, 0 on Failure
Arguments Global MasterVel (0 To 8) As Double

VB Example:

```
Ret = AX_GetAxisVelocity(mastervel(0))          'Get Master velocity
‘ Display Current Profile Feedrate
FeedsReadout (3). Caption = Format (mastervel (0), "0000.00")
```

Declaration: Public Declare Function AX_GetAxisVelocity Lib _
"MotionCore.DLL" (ByRef DMasterVel as Double) As Integer

(Bit Functions)

AX_BitOn

Description Turns on any bit except the Inputs
Returns (Integer) 1 on Success, 0 on Failure
Arguments Global BitNum as Integer

VB Example: **Call Ax_BitOn (512) ‘set Kill Moves to Card**

Declaration: Public Declare Function AX_BitOn Lib _
"MotionCore.DLL" (ByVal BitNum as Integer) As Integer

AX_BitOff

Description Turns off any bit except the Inputs
Returns (Integer) 1 on Success, 0 on Failure
Arguments Global BitNum as Integer

VB Example: **Call Ax_BitOff (512) 'Clear Kill Moves to Card**

Declaration: Public Declare Function AX_BitOff Lib _
"MotionCore.DLL" (ByVal BitNum as Integer) As Integer

(Bit Group Functions)

AX_GetInputs

Description Extract all 96 Inputs as 0's or 1's to a 1 based string (See Control signals Map)
Returns (Integer) 1 on Success, 0 on Failure
Arguments Global Inputs As String: Inputs = Space (97)

VB Example: **Ret = AX_GetInputs (Inputs)**
If Mid (Inputs, 2, 1) = 1 Then Estop = True

Declaration: Public Declare Function AX_GetInputs Lib _
"MotionCore.DLL" (ByVal Inputs as String) As Integer

AX_GetOutputs

Description Extract all 96 Outputs as 0's or 1's to a 1 based string (See Control signals Map)
Returns (Integer) 1 on Success, 0 on Failure
Arguments Global Outputs As String: Outputs = Space (97)

VB Example: **Ret = AX_GetOutputs (Outputs)**
If Mid (Outputs, 2, 1) = 1 Then Output 2 is on

Declaration: Public Declare Function AX_GetInputs Lib _
"MotionCore.DLL" (ByVal Outputs as String) As Integer

AX_GetControlBits

Description GET INTERNAL CONTROL BITS [128 - 255] extracts all 128 BITS from DLL
Returns (Integer) 1 on Success, 0 on Failure
Arguments Global ControlSignals As String: ControlSignals = Space (129)

VB Example: Ret = AX_GetControlBits (ControlSignals)

Declaration: Public Declare Function AX_GetControlBits Lib _
"MotionCore.DLL" (ByVal ControlSignals As String) As Integer

AX_GetMasterFlags

Description Get all 32 MASTERFLAGS bits from DLL
Returns (Integer) 1 on Success, 0 on Failure
Arguments Global MasterFlags As String: MasterFlags = Space (33)

VB Example: Ret = AX_GetMasterFlags (MasterFlags)

Declaration: Public Declare Function AX_GetMasterFlags Lib _
"MotionCore.DLL" (ByVal MasterFlags As String) As Integer

AX_GetMcodeBits

Description GET M CODE BITS [1920 TO 2047] Extracts all BITS from DLL
Returns (Integer) 1 on Success, 0 on Failure
Arguments Global MCodes As String: MCodes = Space (129)

VB Example: Ret = AX_GetMcodeBits (MCodes)

Declaration: Public Declare Function AX_GetMcodeBits Lib _
"MotionCore.DLL" (ByVal MCodeBits as String) As Integer

AX_GetAdcInputs

Description Get All 8 Analog to Digital Channel Inputs in an Array
Returns (Integer) 1 on Success, 0 on Failure
Arguments Global ADCs (0 To 8) As Single

VB Example: Ret = AX_GetAdcInputs (ADCs (0))

Declaration: Public Declare Function AX_GetAdcInputs Lib _
"MotionCore.DLL" (ByRef ADCs as Single) As Integer

(Jog and Homing Functions)

AX_JogMinus

AX_JogPlus

AX_JogMinusFast

AX_JogPlusFast

AX_JogStop

Description Jogs the Selected Axis @ JogSpeed Defined in the Parameters.Cfg file

Returns (Integer) 1 on Success, 0 on Failure
Arguments Global CurrentAxis as Integer
0 = Axis0 1=Axis1 2=Axis2

VB Example:
‘ To Jog an Axis Place Code in “MouseDown” Event

```
Private Sub JogPB_MouseDown (Button As Integer, Shift As Integer, X As Single,  Y As Single)
 Ret = AX_JogMinus (CurrentAxis)          ‘ Jog Axis Minus @ Normal Speed
End Sub
```

‘ To Stop Jogging Place Code in “MouseUP” Event

```
Private Sub JogPB_MouseUp (Button As Integer, Shift As Integer, X As Single,  Y As Single)
 Ret = AX_JogStop                         ‘ Stops All Jogging/All Axis
End Sub
```

```
Ret = AX_JogMinus (CurrentAxis)          ‘ Jog Axis Minus @ Normal Speed
Ret = AX_JogPlus (CurrentAxis)           ‘ Jog Axis Minus @ Normal Speed
Ret = AX_JogMinusFast (CurrentAxis)      ‘ Jog Axis Minus @ Fast Speed
Ret = AX_JogPlusFast (CurrentAxis)       ‘ Jog Axis Minus @ Fast Speed
Ret = AX_JogStop                         ‘ Stops All Jogging
```

Declaration:

```
Public Declare Function AX_JogMinus Lib _
"MotionCore.DLL" (ByVal CurrentAxis as Integer) As Integer

Public Declare Function AX_JogPlus Lib _
"MotionCore.DLL" (ByVal CurrentAxis as Integer) As Integer

Public Declare Function AX_JogMinusFast Lib _
"MotionCore.DLL" (ByVal CurrentAxis as Integer) As Integer

Public Declare Function AX_JogPlusFast Lib _
"MotionCore.DLL" (ByVal CurrentAxis as Integer) As Integer
```

AX_HomeAxis

Description Calls the Homing Subroutines located in Prog0 of our AcroBasic Template
Axis0 is at prog0 line 100
Axis1 is at Prog0 line 200
Axis2 is at Prog0 line 300
Returns (Integer) 1 on Success, 0 on Failure
Arguments Global CurrentAxis as Integer
0 = Axis0 1=Axis1 2=Axis2

Declaration

```
Public Declare Function AX_HomeAxis Lib _
"MotionCore.DLL" (ByVal CurrentAxis as Integer) As Integer"
```

VB Example:

```
ret = AX_HomeAxis (CurrentAxis)          ‘Home the Selected Axis
```

Place VB Code in a button

```
'Home X Axis
        Ret = AX_HomeAxis (0)
        HomingActiveLblTXT.Caption = " Homing ... " & Axis0Letter
        Sleep (500)
WaitForX: DoEvents
        If Mid (ControlSignals, 2, 1) = 1 Or Mid (ControlSignals, 3, 1) = 1 Then Goto HomeErr 'if Estop exit
         'If X not home yet done yet loop
        If Mid (ControlSignals, 43, 1) = 0 Then Goto WaitForX
```

(G Code Functions)

AX_LoadGCodeFile

Description Send DLL a G-Code Filename to load in the Motion List
Returns (Integer) 1 on File Load Success, 0 on Failure
Arguments Global GCodeFilename as String

VB Example:

```
Ret = AX_LoadGCodeFile (GCodeFilename) 'send file to DLL
```

Declaration

```
Public Declare Function AX_LoadGCodeFile Lib _
"MotionCore.DLL" (ByVal filename As String) As Integer
```

Note: If File has Error the Error message System will pick it up to allow you to see what line caused the Error

AX_ GetGCodeLineNumber

Description Extract Current Gcode Line Number being executed in cycle
Returns (Integer) 1 on Success, 0 on Failure
Arguments Global gCodeLineNumber As Long

VB Example:

```
Ret = AX_GetGCodeLineNumber (gCodeLineNumber) 'Get cur Line from DLL
```

Declaration

```
Public Declare Function AX_GetGCodeLineNumber Lib _
 "MotionCore.DLL" (ByRef gCodeLineNumber As Long) As Integer
```

(MDI Mode Functions)

AX_StartMDI

Description Start MDI Mode Signal to DLL
Returns (Integer) 1 on Success, 0 on Failure
Arguments None

VB Example:

```
Call AX_StartMDI: Sleep (100)
```

Declaration: Public Declare Function AX_StartMDI Lib "MotionCore.DLL" _
() As Integer

AX_StopMDI

Description End MDI Mode Signal to DLL
Returns (Integer) 1 on Success, 0 on Failure
Arguments **None**

VB Example: Call AX_StopMDI: Sleep (100)

Declaration: Public Declare Function AX_StopMDI Lib "MotionCore.DLL" _
() As Integer

AX_ExecuteMDI

Description Send MDI Command String to DLL to Execute
Returns (Integer) 1 on Success, 0 on Failure
Arguments Global MDICmdStr as String

VB Example: Call AX_ExecuteMDI (MDICmdStr): Sleep (100)

Declaration Public Declare Function AX_ExecuteMDI Lib "MotionCore.DLL" _
(ByVal MDICmdStr as String) As Integer

(Virtual Overrides)

AX_SetFov

AX_SetRov

AX_SetSov

Description Sets the Feed/Rapid/Spindle Speed Overrides for machines not using External Override POTS or Switches
Returns (Integer) 1 on Success, 0 on Failure
Arguments Global OverrideVal as Double

VB Example: Call AX_SetFov (OVRSlider (0). value * 0.01) 'Set Feedrate OVR
Call AX_SetRov (OVRSlider (0). value * 0.01) 'Set Rapid OVR
Call AX_SetSov (OVRSlider (0). value * 0.01) 'Set Spindle Speed OVR

Declaration Public Declare Function AX_SetFov Lib "MotionCore.DLL" _
(ByVal FovVal as Double) As Integer

Public Declare Function AX_SetRov Lib "MotionCore.DLL" _
(ByVal RovVal as Double) As Integer

Public Declare Function AX_SetSov Lib "MotionCore.DLL" _
(ByVal SovVal as Double) As Integer

(Graphics Functions)

AX_InitializeGPH

Description Initialize Graphics Engine and give it the handle of the Control to plot to, this is typically a PictureBox control in VB. The DLL will extract the coordinates of the window in pixels, and use these coordinates form then on. It is important to not move the window once it has been initialized as incorrect display of graphics could occur.

Returns (Integer) 1 on Success, 0 on Failure
Arguments Long … **Handle (. hwnd) of the Graphics control**

VB Example: Ret = AX_InitializeGPH
Declaration: Declare Function AX_InitializeGPH Lib _
"MotionCore.DLL" (ByVal ProcHwnd As Long) As Integer

AX_SetGphView

Description Informs the Graphics Engine which View Port to Use.

ViewPort Layout:

0=Top	**4=NE**
1=Front	**5=NW**
2=Side	**6=SE**
3=NW	**7=SW**

ZoomAll -> Tells Engine to refresh ViewPort to outer boundaries. Depending on if Parameter "UseMachWindow=1" if not it uses the Saved ViewPort coordinates.

Returns (Integer) 1 on Success, 0 on Failure
Arguments viewpoint as Integer, ZoomAll As integer

VB Example: Ret = AX_ SetGphView (0) 'set to top view

Declaration: Public Declare Function AX_SetGphView Lib _
"MotionCore.DLL" (ByVal viewpoint As Integer, ByVal ZoomAll as Integer) As Integer

AX_ SetGraphics

Description Pass Screen Coordinates (in Pixels) of the ZoomBox to the API for Zoom operations
Returns (Integer) 1 on Success, 0 on Failure
Arguments: ByVal iGraphicsOn As Integer

VB Example:
```
'Turn Graphics On
Call AX_SetGraphics (1)
'Turn Graphics Off
Call AX_SetGraphics (0)
```

Declaration:
```
Public Declare Function AX_SetGraphics Lib "MotionCore.DLL" _
 (ByVal iGraphicsOn As Integer) As Integer
```

(Error Message System)

AX_SystemErr

Description 'Get the System errors from the DLL if any
Returns (Integer) 1 on Success, 0 on Failure
Arguments: ByVal ErrMsg as String, ByVal ErrSource as String, ByRef ErrLineNum As Long

VB Example:
```
ErrMsg = Space (255) : ErrSource = Space(128)
Call AX_SystemErr (ErrMsg, ErrSource, ErrLineNum)
ErrMsg = Mid (ErrMsg, 1, lstrlen (ErrMsg))
ErrSource = Mid (ErrSource, 1, lstrlen (ErrSource))

frmErrMsg.ErrMsgTxt.Text = Trim(ErrMsg) & vbCrLf & "Source = " & Trim(ErrSource) & vbCrLf
& "Line = " & ErrLineNum
ErrDisplayOn = True
```

Declaration:
```
Public Declare Function AX_SystemErr Lib _
"MotionCore.DLL" (ByVal ErrMsg As String, ByVal ErrSource As String, ByRef ErrLineNum As
Long) As Integer
```

AX_ClearSystemErr

Description Acknowledgement from Vb to DLL side from user that he got the error MSG.
This function also Clear the Error in the DLL Handler.If more errors are in the message Handler the Above Call will be Valid again.
Returns (Integer) 1 on Success, 0 on Failure
Arguments: None

VB Example:
```
Call AX_ClearSystemErr
```

Declaration:
```
Public Declare Function AX_SystemErr Lib _
"MotionCore.DLL" (ByVal ErrMsg as String, ByVal ErrSource As String, ByRef ErrLineNum As
Long) As Integer
```

(Cycle Functions)

AX_CycleStart

Description Cause DLL to Start running the current Loaded G code file from the Specified Line

Returns (Integer) 1 on Success, 0 on Failure
Arguments: Global Linenum as Long

VB Example: Call AX_CycleStart (0) ‘ Start Program from first Line

Declaration: Public Declare Function AX_CycleStart Lib _
"MotionCore.DLL" (ByVal Linenum as Long) As Integer

Note: **DLL Motion List is zero based.**
The System can run in 2 modes Auto/Step

To Set control System in Auto Mode
Call AX_BitOn (133) 'Set Auto/Step Mode Bit=1
Call AX_BitOn (134) ' RunMode bit on
To Set control System in Step Mode
Call AX_BitOff (133) 'Set Auto/Step Mode Bit=0
Call AX_BitOn (134) ' RunMode bit on

FeedHold

Description Setting these bits in an Option Button will effect a feed hold to the Card
This is not an API function but how we effect Feed hold
Returns Nothing
Arguments: None

VB Example: Private Sub FHoldBtn_Click ()

```
If FHoldBtn.value = 1 Then
 Call AX_BitOn (520) 'set feed hold to card
 Call AX_BitOn (131) 'set feed hold Control Signal bit
 CmdStart.Enabled = False 'Cycle Start Button
  INCYCLE = False
Else
 Call AX_BitOff (518) 'Clear Feed hold to card
 Call AX_BitOff (519) 'Clear Feed hold to card
 Call AX_BitOff (520) 'Clear Feed hold to card
 Call AX_BitOff (131) 'Clear Feed hold Control Signal bit
      ' program running bit
  If Mid (ControlSignals, 75, 1) = 1 Then
   CmdStart.Enabled = True 'Cycle Start Button
  End If
End If
End Sub
```

Reset

Description Setting these bits in an Option Button will effect a feed hold to the Card
This is not an API function but how we effect a Reset
Returns Nothing
Arguments: None

VB Example: Private Sub ResetBtn_Click ()

Call AX_BitOn (130) 'set reset bit …Stop Program and reset to line 0

Auto/Step Mode

Description Setting these bits in an Option Button will place control in Auto/Step mode
This is not an API function but how we effect a Reset
Returns Nothing
Arguments: None

VB Example: **To Set control System in Auto Mode**
Call AX_BitOn (133) 'Set Auto/Step Mode Bit=1
Call AX_BitOn (134) ' RunMode bit on
To Set control System in Step Mode
Call AX_BitOff (133) 'Set Auto/Step Mode Bit=0
Call AX_BitOn (134) ' RunMode bit on

I/O Map

Acroloop I/O Layout

Control Signals

These bits [**128-255**] used by the DLL for communication with the VB and AcroBasic.
The **MotionCore.DLL** depends on these bits so **don't use them in your Vb programs** for any other purpose than as defined below. If you need to use some bits for your program logic, we have set aside some **User Scratch Pad bits at bit locations (224-248)** that you may use as desired.

Bit		Description
128	Cycle Start	Set by AcroBasic to Signal that Cycle Start has been pressed.
129	Estop	Signals that Estop is On or Off.
130	Reset	Signals ACR-MOTIONMAX System to Do a Reset.
131	Feedhold	Signals ACR-MOTIONMAX that Feedhold is on.
132	Dry Run	Signals ACR-MOTIONMAX to run in Dry Run Mode
133	Auto/Step	Signals control to Run in Auto or Step mode Auto=1 Step =0
134	OK to Run	If Not in Feed Hold, Estop, MDI Mode or Jog Mode (This Signal Is Set by ACR-MOTIONMAX)
135	Jog Mode	Set by ACR-MOTIONMAX when In Jog Mode
136	MDI Mode	Set by ACR-MOTIONMAX when in MDI Mode
137	Offsets Mode	Set By ACR-MOTIONMAX when Control is in Offsets Screen
138	In Cycle	Set by ACR-MOTIONMAX when Control Goes in Cycle
139	Edit Mode	Set by ACR-MOTIONMAX when Control Goes in edit Mode
140	JoggedInCycle	Set by ACR-MOTIONMAX when the Control has jogged in cycle
141	ReturnToPreJogPos	Used by Mill max after a InCycle jog
142	OptionalStopActive	When=1 Optional Stop will be acted on else they will be ignored
143	Control Ready	Set In ACR-MOTIONMAX when Software Starts and cleared When Program End
144	Homing Active	Set In AcroBasic to Signal that a Homing operation is in progress
145	Control Initialized	Signal set by AcroBasic that Initialization succeeded to ACR-MOTIONMAX
146	Spindle Encoder Installed	Used by ACR-MOTIONMAX for Rigid Tapping
147	M Done	Set by AcroBasic to Signal that a M code has finished successfully
148	S Done	Set By AcroBasic to Signal that a S code has finished successfully
149	T Done	Set By AcroBasic to Signal that a T code has finished successfully
150	M Strobe	Set By ACR-MOTIONMAX when a M Code has been Sent to be serviced by AcroBasic
151	S Strobe	Set By ACR-MOTIONMAX when a S Code has been Sent to be serviced by AcroBasic

152	T Strobe	Set By ACR-MOTIONMAX when a T Code has been Sent to be serviced by AcroBasic
153	Tool Change Active	Set by ACR-MOTIONMAX DLL to Signal M6 Sub needs to do a change
154	Tool Change Done	Set By AcroBasic in M6 Code to signal a tool change has finished ok
155	Z @ T change Position	Set By AcroBasic to Signal that Z is at the Tool change Position
156	Tool Seek Active	Set in AcroBasic T Strobe Code to Signal that Magazine is Moving
157	Tool Seek Done	Set by AcroBasic to Signal that the seek operation is complete
158	Tool Seek Direction	Set By ACR-MOTIONMAX to Signal bi-directional Magazine direction to Move
159	Tool Seek Speed	Set By AcroBasic for Magazines that have Slow Down Capabilities
160	MPG Enabled	Set By AcroBasic to Signal ACR-MOTIONMAX that Pendant is Enabled
161	MPG X Select	Set by AcroBasic to Signal ACR-MOTIONMAX that Axis is under Pendant Control
162	MPG Y Select	Set by AcroBasic to Signal ACR-MOTIONMAX that Axis is under Pendant Control
163	MPG Z Select	Set by AcroBasic to Signal ACR-MOTIONMAX that Axis is under Pendant Control
164	MPG 4 Select	Set by AcroBasic to Signal ACR-MOTIONMAX that Axis is under Pendant Control
165	MPG 5 Select	Set by AcroBasic to Signal ACR-MOTIONMAX that Axis is under Pendant Control
166	X.1 Mode Select	Set by AcroBasic to Signal ACR-MOTIONMAX that X.1 Mode is selected for Pendant Moves
167	X1 Mode Select	Set by AcroBasic to Signal ACR-MOTIONMAX that X1 Mode is selected for Pendant Moves
168	X10 Mode Select	Set by AcroBasic to Signal ACR-MOTIONMAX that X10 Mode is selected for Pendant Moves
169	X100 Mode Select	Set by AcroBasic to Signal ACR-MOTIONMAX that X100 Mode is selected for Pendant Moves
170	X Home Done	Set by AcroBasic to Signal ACR-MOTIONMAX that Axis has been Referenced
171	Y Home Done	Set by AcroBasic to Signal ACR-MOTIONMAX that Axis has been Referenced
172	Z Home Done	Set by AcroBasic to Signal ACR-MOTIONMAX that Axis has been Referenced
173	4 Axis Home Done	Set by AcroBasic to Signal ACR-MOTIONMAX that Axis has been Referenced
174	5th Axis Home Done	Set by AcroBasic to Signal ACR-MOTIONMAX that Axis has been Referenced
175	Magazine Ref Done	Set by AcroBasic M18 to Signal ACR-MOTIONMAX that Magazine has been Referenced
176	Gear Change Active	Set by AcroBasic when effecting a Gear change
177	Gear Change bit 0	Set by ACR-MOTIONMAX to signal which Gear has been Requested (is a binary code)
178	Gear Change bit 1	Set by ACR-MOTIONMAX to signal which Gear has been Requested (is a binary code)
179	Gear Change bit 2	Set by ACR-MOTIONMAX to signal which Gear has been Requested (is a binary code)
180	Spindle Direction Bit	Set by AcroBasic to signal Spindle direction is for M3/M4 outputs 1 fwd 0 rev
181	0 Speed Arrival bit	Set by AcroBasic to Signal that Spindle is at rest
182	Speed Arrival bit	Set by AcroBasic to signal that spindle is running at the commanded speed
183	Orient Complete	Set By AcroBasic to signal that Spindle is at the orient Position
184	Spindle Enable	Set by ACR-MOTIONMAX to signal that the spindle is Enabled
185	Rigid Tapping Active	Signal will be 1 when the ACR-MOTIONMAX Control is Rigid Tapping
186	Tapping Active	Signal will be 1 when ACR-MOTIONMAX is running a Tapping cycle
187	4th Axis Present	Set by AcroBasic to ACR-MOTIONMAX informing control of 4th axis presence
188	4th Axis Clamped	Set by AcroBasic to ACR-MOTIONMAX signaling that 4th Axis is locked
189	Contouring	Set by ACR-MOTIONMAX to signal that a contouring mode is active
190	SpindleRunning	Set in AcroBasic to form logic to turn Spindle back on if paused
191	Reserved	
192	Over Travel Active	Set by AcroBasic to ACR-MOTIONMAX of an existing OverTravel
193	EnableVirtualEstop	Set to Allow Virtual Estop Button on GUI
194	V Estop Status	Status Signal used by ACR-MOTIONMAX of VEstop
195	Reserved for future use	
196	Reserved for future use	
197	Reserved for future use	
198	Reserved for future use	

199	Reserved for future use	
200	Reserved for future use	
202	Program Running	Set by ACR-MOTIONMAX signals that a G Code program is running InCycle
203	Program Modified	Set by ACR-MOTIONMAX when the current program has been altered
204 to 212		Reserved for future use (Don't Use)
213	Digital FOV Bit0	4 bit Code for External Feedrate Override Switches to
214	Digital FOV Bit1	Create logic 0 – 100 %
215	Digital FOV Bit2	
216	Digital FOV Bit3	
217	Digital ROV Bit0	4 bit Code for External Rapid Override Switches
218	Digital ROV Bit1	
219	Digital ROV Bit2	
220	Digital ROV Bit3	
221	Digital SOV Bit0	4 bit Code for External Rapid Override Switches
222	Digital SOV Bit1	
223	Digital ROV Bit2	
224	Digital ROV Bit3	
225 to 248		Scratch Pad bits (Bits for the User to use as status bits)
249	Reserved	Used by ACR-MOTIONMAX for internal Error Messages display
250	User Error bit 0	Bits 250 to 255 form a binary code for Error Messages to send to ACR-MOTIONMAX
251	User Error bit 1	
252	User Error bit 2	
253	User Error bit 3	
254	User Error bit 4	
255	User Error Bit 5	See: Setting Up User Defined Error Messages

Mcode Bits

These bits **(1920-2047)** these bits are reserved and **should not be used** other than to set an Mcode Function M0-M126 that the Mcode Processor will process.