Motion Control Products

Application note
Generic drive interface: Add-On Instructions for Studio 5000 ™ / RSLogix5000™

AN00222

Rev F (EN)

Ready to use PLC function blocks, combine with a pre-written Mint application for simple control of MicroFlex e150, MicroFlex e190 and MotiFlex e180 drives via Ethernet/IP

Introduction

This application note details how to import and configure the ABB Generic Drive Interface (GDI) Add-on Instructions (AOI) using RSLogix 5000™ version 19 (or later). The same principles can be applied for users of Logix Designer running under Studio 5000 ® so if using this later version of the Rockwell software please assume Logix Designer wherever RSLogix5000 is mentioned. These AOI provide pre-written data structures and function blocks that integrate seamlessly with the Mint based GDI and allow suitable Allen Bradley PLCs to control ABB drives running Mint programs that support Ethernet/IP (MicroFlex e150/e190 and MotiFlex e180). Note that MicroFlex e190 and MotiFlex e180 drives must be provided with the Mint memory card (option code +N8020).

The instructions promote consistency in all projects and greatly simplify the development of Allen Bradley PLC motion control applications where simple point to point motion is required.

This document assumes that the reader has basic knowledge of Allen Bradley PLCs, RSLogix5000 or Studio 5000, Ethernet/IP configuration, Mint Workbench and the Mint GDI. It is recommended that the reader refers to application note AN00204 for details on the Mint GDI operation and configuration.

The AOI with this application note provide mechanisms for an Allen Bradley Ethernet/IP equipped PLC to:

- Issue a home command
- Issue a command to detect a physical axis end stop and use this as a datum position (firmware version 5863 onwards required if using e150 or e180 drives)
- Issue a relative move
- Issue an absolute move
- Issue an incremental relative move (and optionally stop a programmed distance past a “fast-capture” position)
- Issue an incremental absolute move (and optionally stop a programmed distance past a “fast-capture” position)
- Setup an offset target for an incremental move (i.e. position the axis relative to a captured fast interrupt)
- Jog the axis
- Set the axis position
- Issue a speed reference
- Issue a torque reference
- Enable/disable the axis
- Enable/disable hardware limits
- Reset axis errors
- Perform a controlled stop or crash stop on the axis
- Gear the axis to a secondary encoder input
- Set speed, acceleration times, deceleration times and jerk times for all motion

- Control modulo or non-modulo axes

At the same time the PLC is able to monitor status information from the drive including:

- Enabled state
- Ready to be enabled state
- Idle state
- In Position state
- Motor brake state
- Homed state
- Forward limit state
- Reverse limit state
- Fault state
- Stop input state
- Indication of missing fast latch interrupt
- Phase search status
- Error code
- Measured position
- Measured velocity
- Following error
- Axis mode of operation
- RMS current

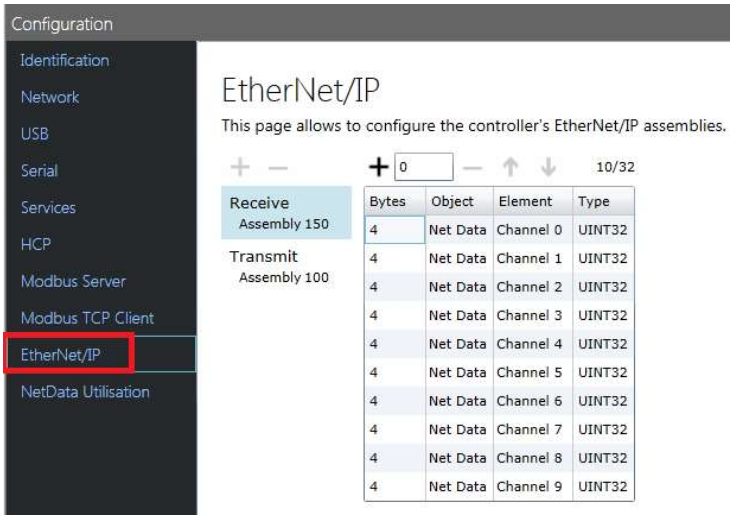
This is all achieved via, what appears to the PLC as, input and output registers. Because we have used 32 bit data (DINT data tyoe) for the interface each value is mapped onto a single 32-bit NETINTEGER or NETFLOAT location in the drive.
An optional watchdog mechanism is also included, allowing the drive to take action (crash stop and disable by default) in the event of communication loss.

Configuring the Generic Drive Interface (GDI) Mint program

The pre-written GDI Mint program only requires only a small amount of customisation to suit the user's application. Please refer to application note AN00204 for details.

Configuring Ethernet/IP on the Mint based drive

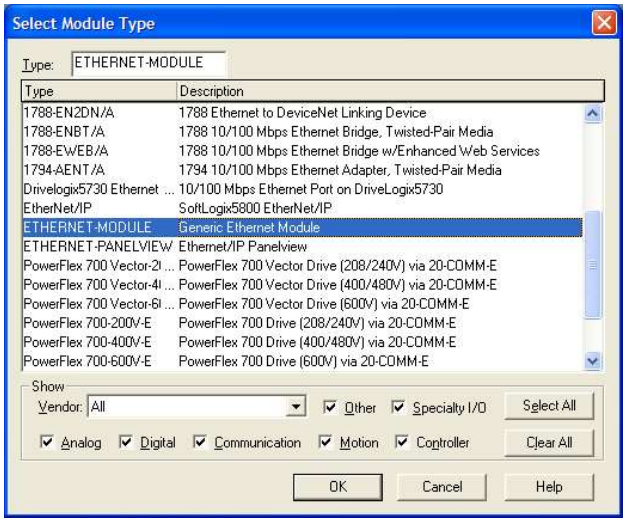
MicroFlex e150 and Mint-enabled MicroFlex e190/MotiFlex e180 drives are delivered "pre-configured" for operation of the GDI via Ethernet/IP. Netdata mappings for 10 input and 10 output words of data are already configured (the GDI only actually requires 9 input words and 7 output words but 10 of each are provided for ease of expansion by the user if required).
This configuration can be seen by entering the "Configuration" screen in Mint Workbench when connected to the drive. Upload the existing configuration from the drive and select the 'Ethernet/IP' section…



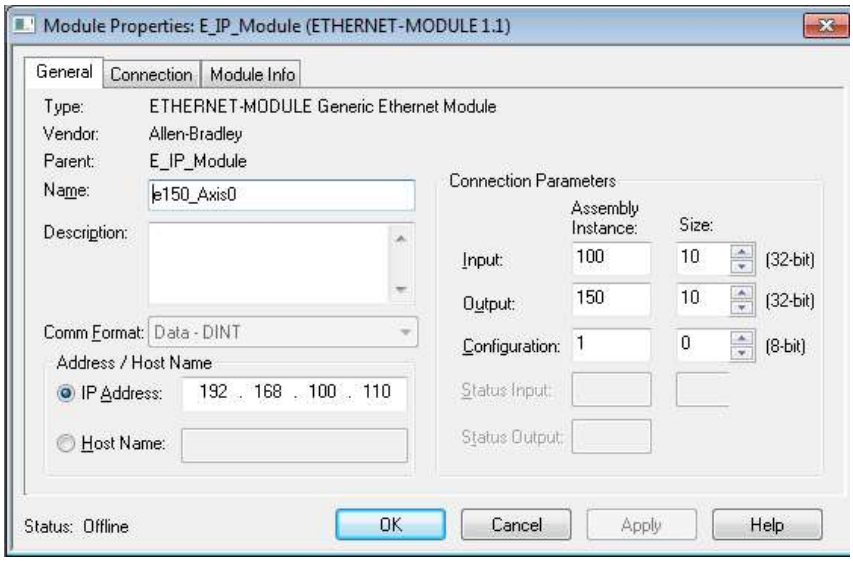
You will see that there are 10 mapped Netdata elements (0 to 9) for Assembly 150 (data received by the drive) and 10 mapped Netdata elements (100 to 109) for Assembly 100 (data transmitted by the drive). There is no need to change any of these mappings for standard operation of the GDI.

Adding the drive to the PLC Ethernet/IP configuration

This process may vary slightly according to PLC model, for this application note we used a Logix5561TM processor (1756-L61) with a 1756-ENET/B Ethernet/IP module. In RSLogix5000 right-click the Ethernet/IP module and select “New Module…”. From the list of possible Ethernet devices select Generic Ethernet Module as shown below:

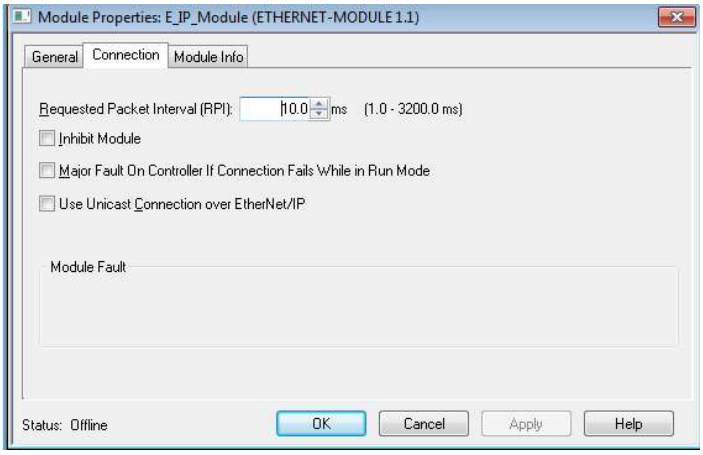


You will now be asked to enter some basic information about the drive…



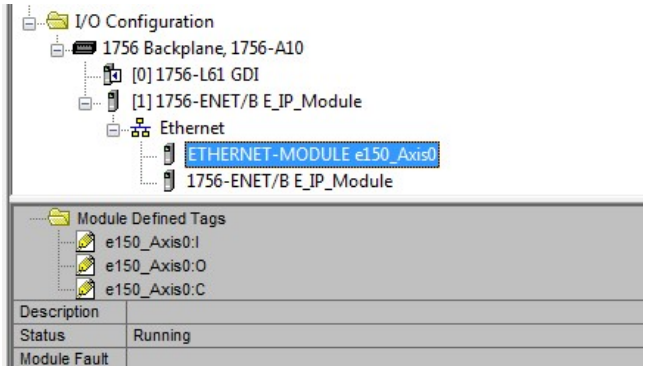
Give the drive a recognisable name. Set the input assembly instance to 100 with a size of 10 (i.e. to match the drive settings). Set the output assembly instance to 150 with a size of 10 also (again to match the drive settings). If you have modified the amount of mapped data on the drive (e.g. to add more PDO values) it is important that these sizes match the amount of mapped data exactly. Enter 1 for the configuration assembly instance and set the size of this to 0 (i.e. it is not used).
Now enter the IP address for the drive (192.168.0.1 is the default IP address for the drive but we’ve used 192.168.100.110 in our example – the drive’s IP address can be modified via the ‘Network’ screen within the Mint Workbench ‘Configuration’ section). Any address is possible providing it matches the drive’s settings and the first three octets (i.e. 192.168.100) match the setting for the Ethernet/IP module in the PLC rack (it is assumed the user knows how to change the Ethernet/IP module IP address if needed – refer to your Rockwell documentation if unsure). We changed our drive’s IP address to suit the PLC’s default subnet setting as this was easiest.

Now select the “Connection” tab and ensure the “Use Unicast Connection over Ethernet/IP” option is deselected as shown below if using drive firmware prior to 5853 (if the drive is running later firmware you can select the Unicast option if necessary). You can adjust the RPI interval if necessary, but we found that 10ms works well for most applications.



Repeat this process for additional drives as required (remember to give each a unique name and IP address). For this application note we have just added a single MicroFlex e150 drive.

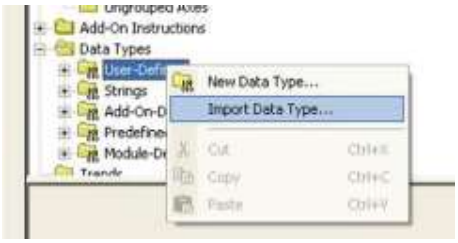
At this point you may want to download the PLC project and go “Online” to check communication is working correctly. If it is you should see something like this in your RSLogix5000 project tree…



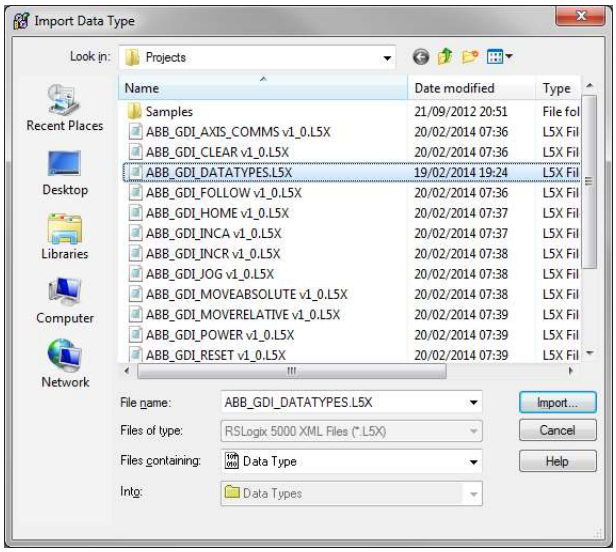
The module status should indicate “Running” with no faults. If you cannot see this status (e.g. it shows “I/O Faulted”) then check your Ethernet/IP module and Generic Ethernet Module settings.

Importing the User Defined Data Types and Add-On Instructions

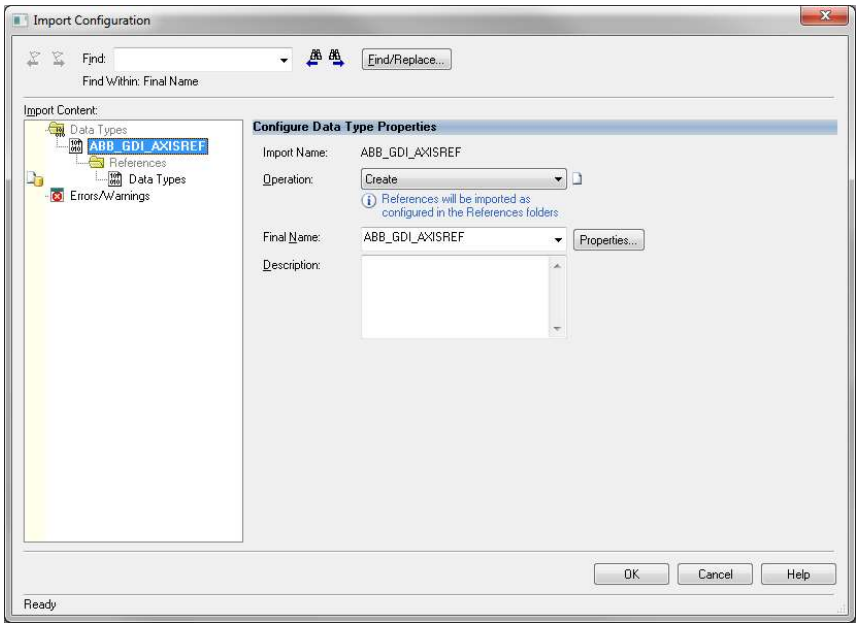
Open the “Data Types” folder in the RSLogix project and right click the “User-Defined” folder…



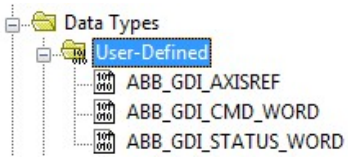
Select “Import Data Type…” and then navigate to and select the “ABB_GDI_DATATYPES.L5X” file that was included with this application note as shown below…



Click on the Import… button. If the GDI data types do not already exist in the project you will be asked to create these as shown below:

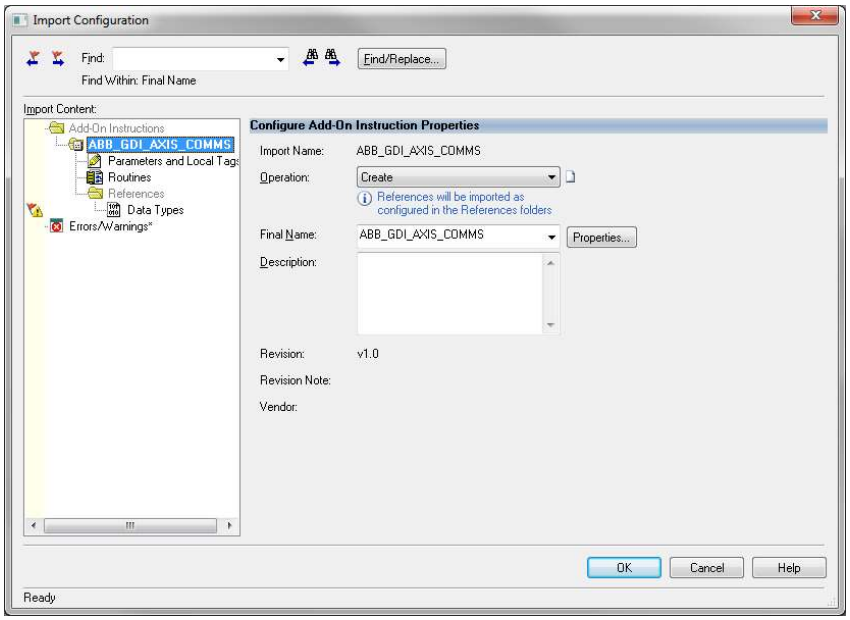


Click on OK. RSLogix will import all the user-defined data types required for operation of the GDI. At the end of the import your User-Defined folder should look like this…



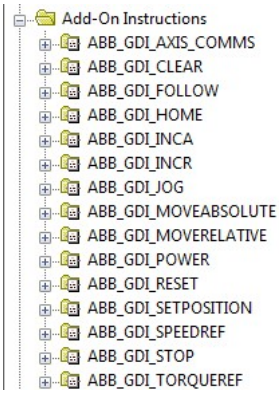
The project now contains data types for the GDI axis (ABB_GDI_AXISREF) as well as data types for the command and status words used by this axis.
Now right click the “Add On Instructions” folder and select “Import Add-On Instruction…” as shown below…

Now navigate to and select an add-on instruction from those supplied with this application note. The ABB_GDI_AXIS_COMMS instruction is essential as this handles the transfer of data from the PLC to the drive. For the remaining motion functions the user may wish to only import those required for a particular application but for this example we selected every single function. There is no way to select multiple functions at the same time so the import process must be repeated for each function. After clicking "Import…" click on the OK button on the next dialog to create the function…



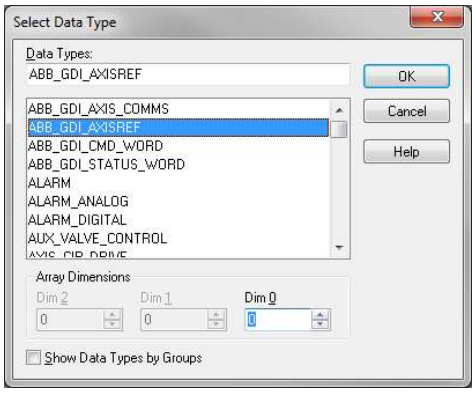
You may see a warning after importing the function. This can be ignored, it occurs because the functions themselves refer to the ABB_GDI_AXISREF data type which wasn't included in the function import. This is because we imported this earlier/separately – this has been done so the data type can be modified independently from the motion functions if required in the future.

After importing all the motion functions your Add On Instructions folder should look something like this (depending on which motion function blocks you decided to import)…



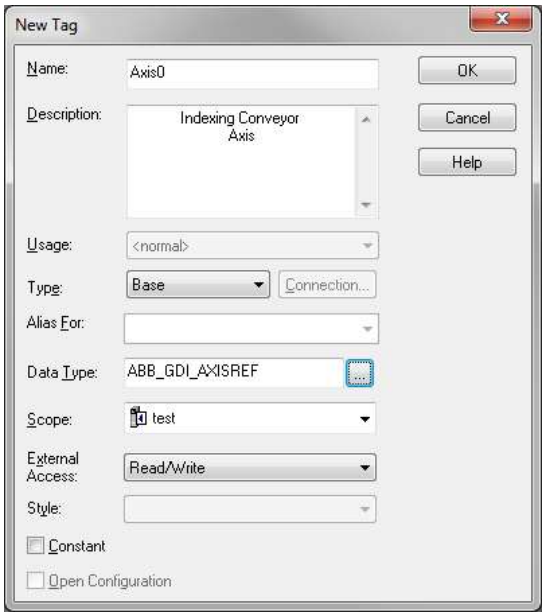
We are now ready to start using these instructions to create some motion in our PLC application. The first step is to create a tag in our project for each axis. For this application note we only have a single MicroFlex e150. We will create this tag in our "Controller Tags" folder in RSLogix5000 (so it has global scope for the project code).

Right click the Controller Tags icon and select “New Tag…”. A dialog appears allowing you to name this tag and, most importantly, define a data type for it. We need to select our ABB_GDI_AXISREF as the data type as shown below…

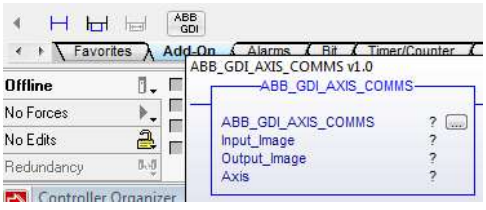


We can either create unique tags for each axis or we can create an array of axes using the Dim 0 up/down control (where each element of the array is then of our ABB_GDI_AXISREF data type). As we only have a single axis we’ll just create a single tag.

After clicking OK and entering some other information about our axis tag we ended up with this dialog…

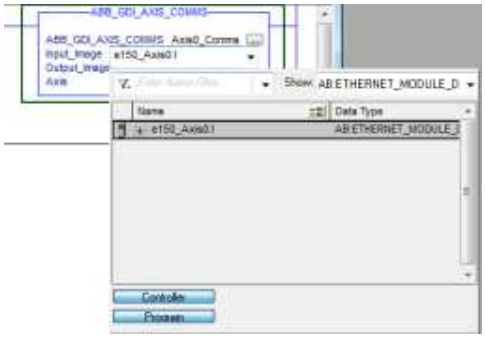


Click OK to create this new tag. The name we give this tag is how we will reference the axis/drive from now on.
We now need to include an instance of the ABB_GDI_AXIS_COMMS function for each axis in our application….
For this application note we will just use the Main Routine for our program logic. Select the first rung and from the instruction toolbox select the ABB_GDI function relating to COMMS (as you hover over each instruction button an image of the function block appears as shown below)



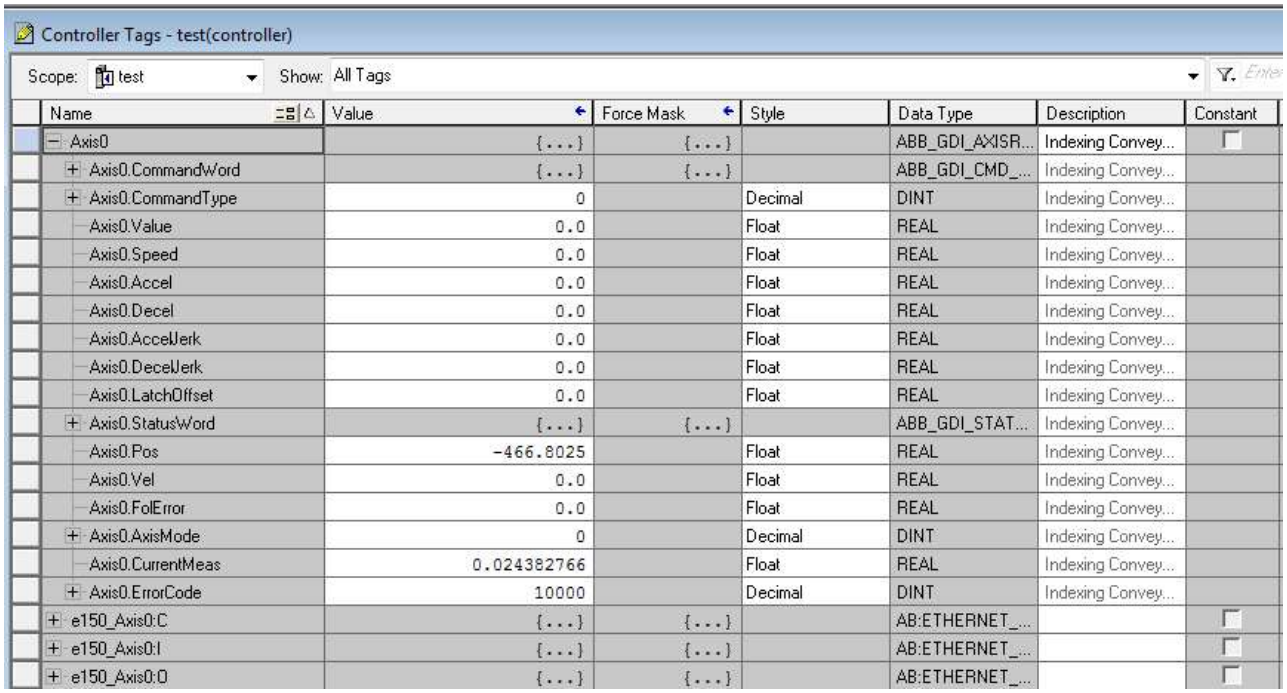
Once this block has been added to the rung you need to create a tag for the instance of this function block (tags must be created for every function block instance used in the program). Click on the ? next to ABB_GDI_AXIS_COMMS and enter a name (we called ours Axis0_Comms to indicate it is the communication function for Axis 0). After entering the name, right click it and select “New Axis0_Comms” (or whatever you’ve called your tag). A dialog will appear allowing you to define this new tag…

Click OK to accept the default tag settings. We now need to link the communication block's input and output images to the input and output assemblies for our Ethernet/IP drive. Double-click on the ? next to Input_Image and then click on the drop down arrow. RSLogix will show a list of possible input assemblies from the project (as we have only added a single drive to the Ethernet network we can only select from one possible item as shown below)…



Repeat this process for the Output_Image. Finally we need to link this function to our axis tag, so enter Axis0 (or whatever you called your tag of type ABB_GDI_AXISREF earlier) as the value for the Axis parameter. If you have followed these instructions correctly your first program rung should look something like this…

We can now download this to the PLC and test that we have basic communications operating correctly.
Double-click the Controller Tags icon in RSLogix5000 to open the Monitor Tags window. Expand the Axis0 tag and you will see the outgoing and incoming data…



Name	Value	Force Mask	Style	Data Type	Description	Constant
Axis0	{...}	{...}		ABB_GDI_AXISR...	Indexing Convey...	
Axis0.CommandWord	{...}	{...}		ABB_GDI_CMD_...	Indexing Convey...	
Axis0.CommandType	0		Decimal	DINT	Indexing Convey...	
Axis0.Value	0.0		Float	REAL	Indexing Convey...	
Axis0.Speed	0.0		Float	REAL	Indexing Convey...	
Axis0.Accel	0.0		Float	REAL	Indexing Convey...	
Axis0.Decel	0.0		Float	REAL	Indexing Convey...	
Axis0.AccelJerk	0.0		Float	REAL	Indexing Convey...	
Axis0.DecelJerk	0.0		Float	REAL	Indexing Convey...	
Axis0.LatchOffset	0.0		Float	REAL	Indexing Convey...	
Axis0.StatusWord	{...}	{...}		ABB_GDI_STAT...	Indexing Convey...	
Axis0.Pos	-466.8025		Float	REAL	Indexing Convey...	
Axis0.Vel	0.0		Float	REAL	Indexing Convey...	
Axis0.FolError	0.0		Float	REAL	Indexing Convey...	
Axis0.AxisMode	0		Decimal	DINT	Indexing Convey...	
Axis0.CurrentMeas	0.024382766		Float	REAL	Indexing Convey...	
Axis0.ErrorCode	10000		Decimal	DINT	Indexing Convey...	
e150_Axis0:C	{...}	{...}		AB:ETHERNET_...		
e150_Axis0:I	{...}	{...}		AB:ETHERNET_...		
e150_Axis0:O	{...}	{...}		AB:ETHERNET_...		

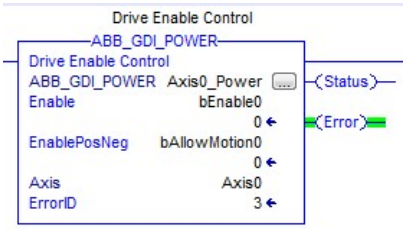
If the Mint GDI application is running on the drive then the drive's position, measured velocity, measured current etc… will all be updating according to the drive's actual values – this will confirm that everything is working correctly and you are ready to continue to program your motion application on the PLC.

AOI GDI Function Blocks

The following sections detail the use of the AOI GDI function blocks:

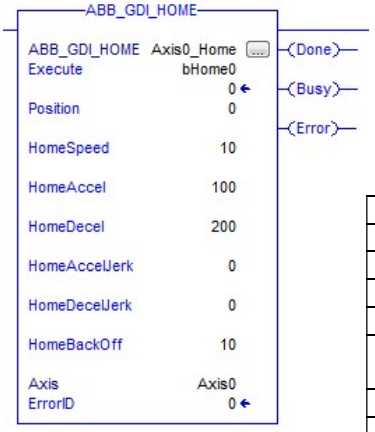
ABB_GDI_POWER

This function block is used to enable / disable an axis. The enable input enables the power stage in the drive and not the function block itself.



	Type	Description
VAR_IN_OUT		
Axis	ABB_GDI_AXISREF	Reference to the axis structure
VAR_INPUT		
Enable	BOOL	Whilst true the PLC will request the axis to be enabled
EnablePosNeg	BOOL	Whist true motion in both directions is permitted. If false motion is prevented (or a stop is performed if motion is already in progress)
VAR_OUTPUT		
Status	BOOL	Indicates whether the axis is enabled (1) or not (0)
Error	BOOL	Set to true if the axis is in error
ErrorID	DINT	Indicates the Mint error code reported by the axis

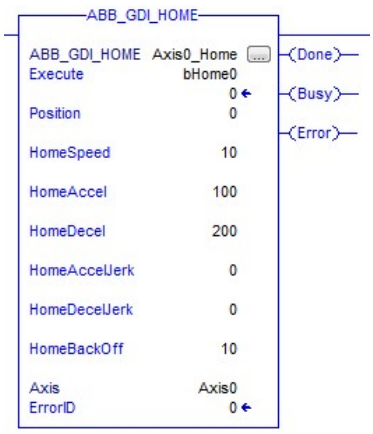
ABB_GDI_HOME



This function block is used to datum an axis. The details of the datum sequence are dependent on the Home type set in the Mint GDI program. The Position input is used to set the axis position at the end of a successful datum sequence.

	Type	Description
VAR_IN_OUT		
Axis	ABB_GDI_AXISREF	Reference to the axis structure
VAR_INPUT		
Execute	BOOL	Start the datum sequence on a rising edge
Position	REAL	Absolute position to be set at the end of a successful datum sequence
HomeSpeed	REAL	Homing speed in user units/sec
HomeAccel	REAL	Homing accel rate in user units/sec^2
HomeDecel	REAL	Homing decel rate in user units/sec^2
HomeAccelJerk	REAL	Homing accel jerk rate in user units/sec^3 (set to 0 for trapezoidal motion)
HomeDecelJerk	REAL	Homing decel jerk rate in user units/sec^3 (set to 0 for trapezoidal motion)
HomeBackOff	REAL	Ratio of Home speed to backoff speed
VAR_OUTPUT		
Done	BOOL	Indicates that the axis has homed successfully. If the Execute input is removed during homing and the axis completes the home sequence the Done output will be set for one PLC scan. If the Execute input remains 1 then the Done output will also remain set (providing the home was successful)
Busy	BOOL	Set true whilst the homing sequence is in progress
Error	BOOL	Set true if the axis is in error
ErrorID	DINT	Indicates the Mint error code reported by the axis

ABB_GDI_FIND_END_STOP

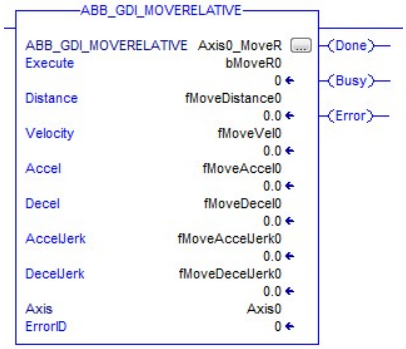


This function block is used as an alternative way to datum an axis in the absence of a home sensor. The axis will run at a commanded velocity with a programmed torque limit until this torque limit is reached and the speed of the axis is less than the programmed idle velocity. The Position input is used to set the axis position at the end of a successful datum sequence.

	Type	Description
VAR_IN_OUT		
Axis	ABB_GDI_AXISREF	Reference to the axis structure
VAR_INPUT		
Execute	BOOL	Start the datum sequence on a rising edge
Position	REAL	Absolute position to be set at the end of a successful datum sequence
FindSpeed	REAL	Speed in user units/sec (the sign of this value determines the seek direction)
FindAccel	REAL	Accel rate in user units/sec^2
FindDecel	REAL	Decel rate in user units/sec^2
FindAccelJerk	REAL	Accel jerk rate in user units/sec^3 (set to 0 for trapezoidal motion)
FindDecelJerk	REAL	Decel jerk rate in user units/sec^3 (set to 0 for trapezoidal motion)
TorqueLimit	REAL	Torque limit to apply during sequence (% of drive rated current)
VAR_OUTPUT		
Done	BOOL	Indicates that the axis has found the end stop successfully. If the Execute input is removed during the sequence and the axis finds the end stop the Done output will be set for one PLC scan. If the Execute input remains 1 then the Done output will also remain set (providing the sequence was successful)

ABB_GDI_MOVERELATIVE

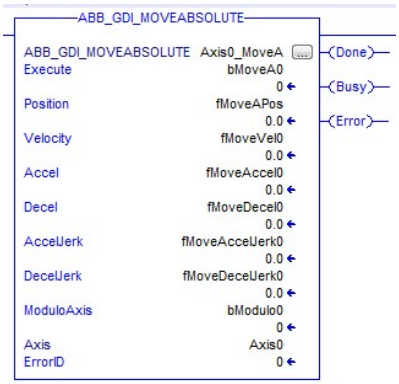
This function block is used to command a controlled motion of a specified distance relative to the start position.



	Type	Description
VAR_IN_OUT		
Axis	ABB_GDI_AXISREF	Reference to the axis structure
VAR_INPUT		
Execute	BOOL	Start the motion on a rising edge
Distance	REAL	Relative distance for the move (in user units)
Velocity	REAL	Maximum speed (not necessarily reached) in user units/sec
Accel	REAL	Accel rate in user units/sec^2
Decel	REAL	Decel rate in user units/sec^2
AccelJerk	REAL	Accel jerk rate in user units/sec^3 (0 for trapezoidal motion)
DecelJerk	REAL	Decel jerk rate in user units/sec^3 (0 for trapezoidal motion)
VAR_OUTPUT		
Done	BOOL	Indicates that the axis has reached the target position successfully. If the Execute input is removed during motion and the relative move completes the Done output will be set 1 for one PLC scan. If the Execute input remains True then the Done output will also remain set (providing the target position was successfully achieved)
Busy	BOOL	Set True whilst the relative move is in progress
Error	BOOL	Set True if the axis is in error
ErrorID	DINT	Indicates the Mint error code reported by the axis

ABB_GDI_MOVEABSOLUTE

This function block is used to command a controlled motion to a specified absolute position. This function can be used with Modulo axes (in which case the shortest route to the specified position will be taken).

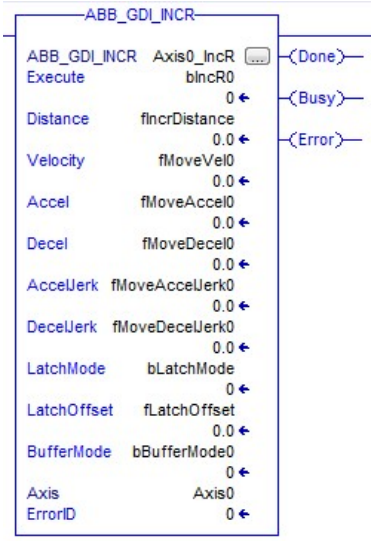


	Type	Description
VAR_IN_OUT		
Axis	ABB_GDI_AXISREF	Reference to the axis structure
VAR_INPUT		
Execute	BOOL	Start the motion on a rising edge
Position	REAL	Target position for the move (in user units)
Velocity	REAL	Maximum speed (not necessarily reached) in user units/sec
Accel	REAL	Accel rate in user units/sec^2
Decel	REAL	Decel rate in user units/sec^2
AccelJerk	REAL	Accel jerk rate in user units/sec^3 (0 for trapezoidal motion)
DecelJerk	REAL	Decel jerk rate in user units/sec^3 (0 for trapezoidal motion)
ModuloAxis	BOOL	Defines whether the axis is a modulo axis (i.e. using an ENCODERWRAP to define travel within one cycle). Absolute moves when using modulo axes are always implemented via the shortest path (e.g. an absolute move to 20 degrees from 350 degrees on a 0-360 degree modulo axis will result in forward travel of 30 degrees)
VAR_OUTPUT		
Done	BOOL	Indicates that the axis has reached the target position successfully. If the Execute input is removed during motion and the absolute move completes the Done output will be set True for one PLC scan. If the Execute input remains True then the Done output will also remain set (providing the target position was successfully achieved)

ABB_GDI_INCR

This function block is used to command a controlled motion of a specified distance relative to the target position at the time of the execution. The target position resulting from a call to this function block can be modified whilst motion is still in progress by any of the following methods:

a. By issuing another ABB_GDI_INCR or ABB_GDI_INCA function (providing input parameter BufferMode is True)
b. By setting the input parameter Latchmode to True and specifying a value for the input parameter LatchOffset. Mint code on the drive will then automatically modify the axis target position such that it stops the LatchOffset distance past the axis position captured by the defined fast interrupt. A bit within the Axis status word (btLatchMissed) is available to indicate failure to detect this fast interrupt (this condition may then be used to alert the operator to a system failure for example). Using Latchmode and LatchOffset allows simple implementation of indexing conveyor applications.



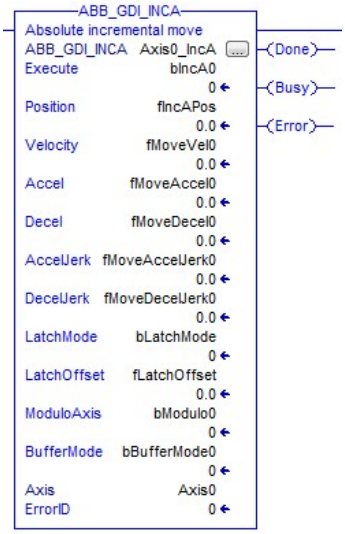
	Type	Description
VAR_IN_OUT		
Axis	ABB_GDI_AXISREF	Reference to the axis structure
VAR_INPUT		
Execute	BOOL	Start the motion on a rising edge
Distance	REAL	Relative distance for the move (in user units)
Velocity	REAL	Maximum speed (not necessarily reached) in user units/sec
Accel	REAL	Accel rate in user units/sec^2
Decel	REAL	Decel rate in user units/sec^2
AccelJerk	REAL	Accel jerk rate in user units/sec^3 (0 for trapezoidal motion)
DecelJerk	REAL	Decel jerk rate in user units/sec^3 (0 for trapezoidal motion)
LatchMode	BOOL	Sets whether the axis should utilise the configured fast latch interrupt and set a new target position 'LatchOffset' user units past the captured position
LatchOffset	REAL	Defines the distance past the captured fast position (in user units) the target for GDI_INCR should be modified by (when input parameter LatchMode is set True)
BufferMode	BOOL	Defines whether the function block should set the Done output and complete as soon as the move has been loaded. Setting BufferMode True allows the application to trigger further incremental moves whilst existing moves are in progress
VAR_OUTPUT		
Done	BOOL	When BufferMode is set False this indicates that the axis has reached the target position successfully. If the Execute input is removed during motion and the relative move completes the Done output will be set True for one PLC scan. If the Execute input remains True then the Done output will also remain set (providing the target position was successfully achieved). When BufferMode is set True the Done output is set for one PLC scan to indicate successful loading of the move
Busy	BOOL	Set True whilst the function block is in progress
Error	BOOL	Set True if the axis is in error
ErrorID	DINT	Indicates the Mint error code reported by the axis

ABB_GDI_INCR is also useful if the application needs to modify SPEED/ACCEL/DECEL of a relative move already in progress. Moves loaded using ABB_GDI_MOVERELATIVE are profiled using the SPEED/ACCEL/DECEL loaded at the time and these cannot be changed once the move has started. By using ABB_GDI_INCR with the input parameter BufferMode set True then it is possible to modify the profile parameters by loading another ABB_GDI_INCR (with new SPEED/ACCEL/DECEL) with input parameter Distance set to zero.

ABB_GDI_INCA

This function block is used to command a controlled motion to a specified absolute position. This function differs from ABB_GDI_MOVEABSOLUTE in that the target position can be modified whilst motion is in progress by any of the following methods:

a. By issuing another ABB_GDI_INCR or ABB_GDI_INCA function (providing input parameter BufferMode is True)
b. By setting the input parameter Latchmode to True and specifying a value for the input parameter LatchOffset. Mint code on the drive will then automatically modify the axis target position such that it stops the LatchOffset distance past the axis position captured by the defined fast interrupt. A bit within the Axis status word (btLatchMissed) is available to indicate failure to detect this fast interrupt (the example programs show how missing 3 latches in a row can be detected – this condition may then be used to alert the operator to a system failure for example).

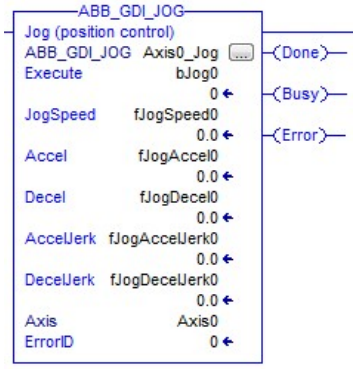


	Type	Description
VAR_IN_OUT		
Axis	ABB_GDI_AXISREF	Reference to the axis structure
VAR_INPUT		
Execute	BOOL	Start the motion on a rising edge
Position	REAL	Absolute position target for the move (in user units)
Velocity	REAL	Maximum speed (not necessarily reached) in user units/sec
Accel	REAL	Accel rate in user units/sec^2
Decel	REAL	Decel rate in user units/sec^2
AccelJerk	REAL	Accel jerk rate in user units/sec^3 (0 for trapezoidal motion)
DecelJerk	REAL	Decel jerk rate in user units/sec^3 (0 for trapezoidal motion)
LatchMode	BOOL	Sets whether the axis should utilise the configured fast latch interrupt and set a new target position 'LatchOffset' user units past the captured position
ModuloAxis	BOOL	Defines whether the axis is a modulo axis (i.e. using an ENCODERWRAP to define travel within one cycle). Absolute moves when using modulo axes are always implemented via the shortest path (e.g. an absolute move to 20 degrees from 350 degrees on a 0-360 degree modulo axis will result in forward travel of 30 degrees)
LatchOffset	REAL	Defines the distance past the captured fast position (in user units) the target for ABB_GDI_INCA should be modified by (when input parameter LatchMode is set True)
BufferMode	BOOL	Defines whether the function block should set the Done output and complete as soon as the move has been loaded. Setting BufferMode True allows the application to trigger further incremental moves whilst existing moves are in progress
VAR_OUTPUT		
Done	BOOL	When BufferMode is set False this indicates that the axis has reached the target position successfully. If the Execute input is removed during motion and the relative move completes the Done output will be set True for one PLC scan. If the Execute input remains True then the Done output will also remain set (providing the target position was successfully achieved). When BufferMode is set True the Done output is set for one PLC scan to indicate successful loading of the move
Busy	BOOL	Set True whilst the function block is in progress
Error	BOOL	Set True if the axis is in error
ErrorID	DINT	Indicates the Mint error code reported by the axis

ABB_GDI_INCA is also useful if the application needs to modify SPEED/ACCEL/DECEL of an absolute move already in progress. Moves loaded using ABB_GDI_MOVEABSOLUTE are profiled using the SPEED/ACCEL/DECEL loaded at the time and these cannot be changed once the move has started. By using ABB_GDI_INCA with the input parameter BufferMode set True then it is possible to modify the profile parameters by first loading a ABB_GDI_INCA move and then loading a ABB_GDI_INCR (with new SPEED/ACCEL/DECEL) with input parameter Distance set to zero.

ABB_GDI_JOG

This function block is used to command a constant speed move on the axis (using the position loop controller in the drive). Motion is performed as long as the Execute input remains True.



	Type	Description
VAR_IN_OUT		
Axis	ABB_GDI_AXISREF	Reference to the axis structure
VAR_INPUT		
Execute	BOOL	Start the motion on a rising edge and maintain motion as long as the input remains True. Motion ramps to zero speed at the configured Decel rate when Execute becomes False
JogSpeed	REAL	Value for the speed the axis will reach in user units/sec
Accel	REAL	Accel rate in user units/sec^2
Decel	REAL	Decel rate in user units/sec^2
AccelJerk	REAL	Accel jerk rate in user units/sec^3 (0 for trapezoidal motion)
DecelJerk	REAL	Decel jerk rate in user units/sec^3 (0 for trapezoidal motion)
VAR_OUTPUT		
Done	BOOL	Set True as soon as the Jog command has been successfully issued and remains set until Execute becomes False or an axis error occurs
Busy	BOOL	Set True whilst the function block is in progress

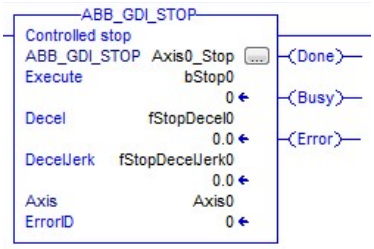
ABB_GDI_SETPOSITION

This function block is used to set the axis position (encoder and position values on the drive) to a programmed value. The axis must be idle when this function is called, otherwise the axis will return an “action not possible - motion in progress” error (Error code 10). If the axis is using an absolute encoder this will set/teach a new absolute position (GDI Mint program v2.17 onwards).

	Type	Description
VAR_IN_OUT		
Axis	ABB_GDI_AXISREF	Reference to the axis structure
VAR_INPUT		
Execute	BOOL	Set the new position on a rising edge
Position	REAL	Value for the axis position to be set (in user units)
VAR_OUTPUT		
Done	BOOL	Set True as soon as the command has been issued (regardless of whether it was successful or not – use the Error output to determine whether the command was successful). Remains True until the Execute input is removed. If the Execute input is removed before the Done bit is set then the Done bit will be set for a single PLC cycle.
Busy	BOOL	Set True whilst the function block is in progress (cleared once the Done bit is set)
Error	BOOL	Set True if the axis is in error
ErrorID	DINT	Indicates the Mint error code reported by the axis

ABB_GDI_STOP

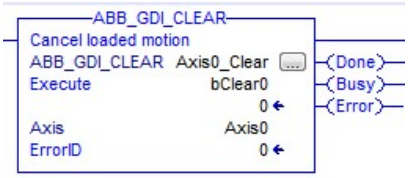
This function block is used to perform a controlled stop on the axis at the programmed deceleration rate.



	Type	Description
VAR_IN_OUT		
Axis	ABB_GDI_AXISREF	Reference to the axis structure
VAR_INPUT		
Execute	BOOL	Start the controlled stop on a rising edge
Decel	REAL	Decel rate in user units/sec^2
DecelJerk	REAL	Decel jerk rate in user units/sec^3 (0 for trapezoidal motion)
VAR_OUTPUT		
Done	BOOL	Set True when the axis becomes idle after completing the controlled stop or if an error occurs when the stop command is issued. Remains True until the Execute input is removed. If the Execute input is removed before the Done bit is set then the Done bit will be set for a single PLC cycle.
Busy	BOOL	Set True whilst the stop is in progress – cleared once the Done bit is set
Error	BOOL	Set True if the axis is in error
ErrorID	DINT	Indicates the Mint error code reported by the axis

ABB_GDI_CLEAR

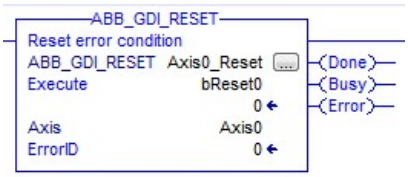
This function block is used to crash stop the axis and interrupt any motion that is in progress. The axis will remain enabled (providing ABB_GDI_POWER is requesting the enabled state and the axis is not in error).



	Type	Description
VAR_IN_OUT		
Axis	ABB_GDI_AXISREF	Reference to the axis structure
VAR_INPUT		
Execute	BOOL	Start the crash stop on a rising edge
VAR_OUTPUT		
Done	BOOL	Set True when the axis becomes idle after completing the crash stop or if an error occurs when the crash stop command is issued. Remains True until the Execute input is removed. If the Execute input is removed before the Done bit is set then the Done bit will be set for a single PLC cycle.
Busy	BOOL	Set True whilst the stop is in progress – cleared once the Done bit is set
Error	BOOL	Set True if the axis is in error
ErrorID	DINT	Indicates the Mint error code reported by the axis

ABB_GDI_RESET

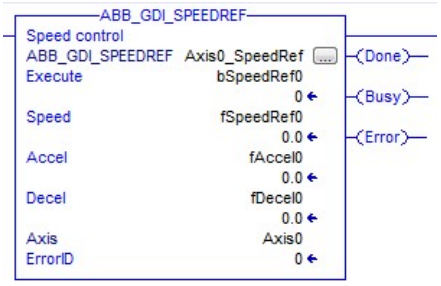
This function block is used to reset any axis error that is present.



	Type	Description
VAR_IN_OUT		
Axis	ABB_GDI_AXISREF	Reference to the axis structure
VAR_INPUT		
Execute	BOOL	Start the fault reset on a rising edge
VAR_OUTPUT		
Done	BOOL	Set True when the axis no longer has an error present. Remains True until the Execute input is removed. If the Execute input is removed before the Done bit is set then the Done bit will be set for a single PLC cycle. The Done bit will not be set if the error could not be cleared (use the Busy output to detect when the fault reset has been attempted)
Busy	BOOL	Set True whilst the function block is attempting to clear any axis error
Error	BOOL	Set True if the axis is in error
ErrorID	DINT	Indicates the Mint error code reported by the axis

ABB_GDI_SPEEDREF

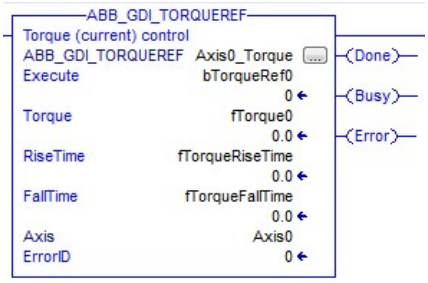
This function block is used to command a speed/velocity reference on the axis. In this mode of operation the position loop is not used on the drive (so no following error is recorded or acted upon). The axis will remain in Speed control mode (as indicated by the Statusword bits for Controlmode) until motion of another control mode type is issued (e.g. a position controlled move). To switch from zero speed operation (in speed control mode) to holding position (in position control mode) an ABB_GDI_MOVERELATIVE could be issued, for example, with a relative move distance of zero user units.



	Type	Description
VAR_IN_OUT		
Axis	ABB_GDI_AXISREF	Reference to the axis structure
VAR_INPUT		
Execute	BOOL	Start the axis on a rising edge and maintain motion as long as the input remains True. Motion ramps to zero speed at the configured Decel rate when Execute becomes False
Speed	REAL	Value for the speed the axis will reach in user units/sec. Can be modified whilst Execute is True to change the axis speed
Accel	REAL	Accel rate in user units/sec^2
Decel	REAL	Decel rate in user units/sec^2
VAR_OUTPUT		
Done	BOOL	Set True as soon as the speed reference has been issued (regardless of whether it was successful or not). The Done output remains set until Execute becomes False
Busy	BOOL	Set True whilst the function block is in progress (i.e. whilst Execute is True)
Error	BOOL	Set True if the axis is in error
ErrorID	DINT	Indicates the Mint error code reported by the axis

ABB_GDI_TORQUEREF

This function block is used to command a torque (current) reference on the axis. In this mode of operation the position loop is not used on the drive (so no following error is recorded or acted upon). The axis will remain in torque control mode (as indicated by the Statusword bits for Controlmode) until motion of another control mode type is issued (e.g. a position controlled move). To switch from zero torque operation (in torque control mode) to holding position (in position control mode) an ABB_GDI_MOVERELATIVE could be issued, for example, with a relative move distance of zero user units.



	Type	Description
VAR_IN_OUT		
Axis	ABB_GDI_AXISREF	Reference to the axis structure
VAR_INPUT		
Execute	BOOL	Start the torque reference on a rising edge and maintain torque as long as the input remains True. Torque ramps to zero at the configured FallTime rate when Execute becomes False
Torque	REAL	Value for the torque reference the axis will use (in % of DRIVERATEDCURRENT – see Mint Help file). Can be modified whilst Execute is True to change the torque produced
RiseTime	REAL	Sets the time taken (in ms) for current to rise from zero to DRIVEPEAKCURRENT (see Mint Help file)
FallTime	REAL	Sets the time taken (in ms) for current to fall from DRIVEPEAKCURRENT to zero (see Mint Help file)
VAR_OUTPUT		
Done	BOOL	Set True as soon as the torque reference has been issued (regardless of whether it was successful or not). The Done output remains set until Execute becomes False
Busy	BOOL	Set True whilst the function block is in progress (i.e. whilst Execute is True)
Error	BOOL	Set True if the axis is in error
ErrorID	DINT	Indicates the Mint error code reported by the axis

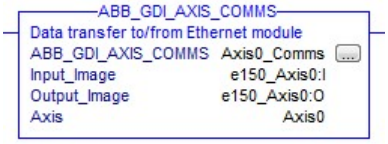
ABB_GDI_FOLLOW

This function block is used to command the axis to start following the configured master encoder reference at the programmed follow ratio (only position locked gearing is possible when using e100 drives, e150 drives offer additional following modes).

	Type	Description
VAR_IN_OUT		
Axis	ABB_GDI_AXISREF	Reference to the axis structure
VAR_INPUT		
Execute	BOOL	Start the follow on a rising edge. The axis will remain in follow mode when the Execute input becomes False (to stop the follow issue another motion command or clear motion using ABB_GDI_CLEAR)
Ratio	REAL	Value for the follow (gear) ratio between the axis and the master encoder reference (the value will affected by the scaling of the axis and the scaling of the master encoder – see the Mint Help file topic for FOLLOW). To set a new ratio whilst following it is necessary to issue a new ABB_GDI_FOLLOW command
VAR_OUTPUT		
Done	BOOL	Set True as soon as the follow has been issued (regardless of whether it was successful or not). The Done output remains set until Execute becomes False
Busy	BOOL	Set True whilst the function block is in progress (i.e. whilst Execute is True)
Error	BOOL	Set True if the axis is in error
ErrorID	DINT	Indicates the Mint error code reported by the axis

ABB_GDI_AXIS_COMMS

This function block is used to transfer command/status data between the application layer and the communication layer of the PLC program. An instance of the relevant function block must exist for each axis in the application.



	Type	Description
VAR_IN_OUT		
Input_Image	AB:ETHERNET_MODULE_DINT_40Bytes:I:0	Reference to the input assembly instance for the axis
Output_Image	AB:ETHERNET_MODULE_DINT_40Bytes:O:0	Reference to the output assembly instance for the axis
Axis	ABB_GDI_AXISREF	Reference to the axis structure

Using the Axis Structure

Most of the functionality of the GDI is encapsulated by the various GDI functions provided as Add On Instructions. However, in some cases the application logic may find access to the axis structure data useful. The ABB_GDI_AXISREF data type deceleration is shown below:

Name	Data Type	Style	Description	External Access
CommandWord	ABB_GDI_CMD_WO			Read/Write
CommandType	DINT	Decimal		Read/Write
Value	REAL	Float		Read/Write
Speed	REAL	Float		Read/Write
Accel	REAL	Float		Read/Write
Decel	REAL	Float		Read/Write
AccelJerk	REAL	Float		Read/Write
DecelJerk	REAL	Float		Read/Write
LatchOffset	REAL	Float		Read/Write
StatusWord	ABB_GDI_STATUS_			Read/Write
Pos	REAL	Float		Read/Write
Vel	REAL	Float		Read/Write
FolError	REAL	Float		Read/Write
AxisMode	DINT	Decimal		Read/Write
CurrentMeas	REAL	Float		Read/Write
ErrorCode	DINT	Decimal		Read/Write

This data structure in turn contains two further data structures (ABB_GDI_CMD_WORD and ABB_GDI_STATUS_WORD). The declarations for these are shown below:

Command word

Name	Data Type	Style	Description	External Access
btEnable	BOOL	Decimal	Enable	Read/Write
btMotionAllowed	BOOL	Decimal	Allow Motion	Read/Write
btPosLatchEnable	BOOL	Decimal	Enable Position Latch	Read/Write
btDisFwdLimit	BOOL	Decimal	Disable Fwd Limit	Read/Write
btDisRevLimit	BOOL	Decimal	Disable Reverse Limit	Read/Write
btModulo	BOOL	Decimal	Modulo Axis	Read/Write
btFaultReset	BOOL	Decimal	Reset Fault	Read/Write
btTriggerCmd	BOOL	Decimal	Trigger Motion Cmd	Read/Write
btWatchDog	BOOL	Decimal	Watchdog	Read/Write
btIgnoreFE	BOOL	Decimal	FolErrorMode	Read/Write

Status word

Name	Data Type	Style	Description	External Access
btEnabled	BOOL	Decimal		Read/Write
btIdle	BOOL	Decimal		Read/Write
btInPos	BOOL	Decimal		Read/Write
btBrakeEngaged	BOOL	Decimal		Read/Write
btHomed	BOOL	Decimal		Read/Write
btFwdLimit	BOOL	Decimal		Read/Write
btRevLimit	BOOL	Decimal		Read/Write
btFault	BOOL	Decimal		Read/Write
btStopInput	BOOL	Decimal		Read/Write
btReadyToEnable	BOOL	Decimal		Read/Write
btControlMode0	BOOL	Decimal		Read/Write
btControlMode1	BOOL	Decimal		Read/Write
btTriggerDone	BOOL	Decimal		Read/Write
btPermitted	BOOL	Decimal		Read/Write
btLatchMissed	BOOL	Decimal		Read/Write
btFaultReset	BOOL	Decimal		Read/Write
btPhaseSearchDone	BOOL	Decimal		Read/Write

The PLC code can therefore access any of this data via these structures.

Example:
Two rungs accessing the axis data structure directly, one reading the status of the Forward Limit Input on the drive and the other storing the measured axis velocity…

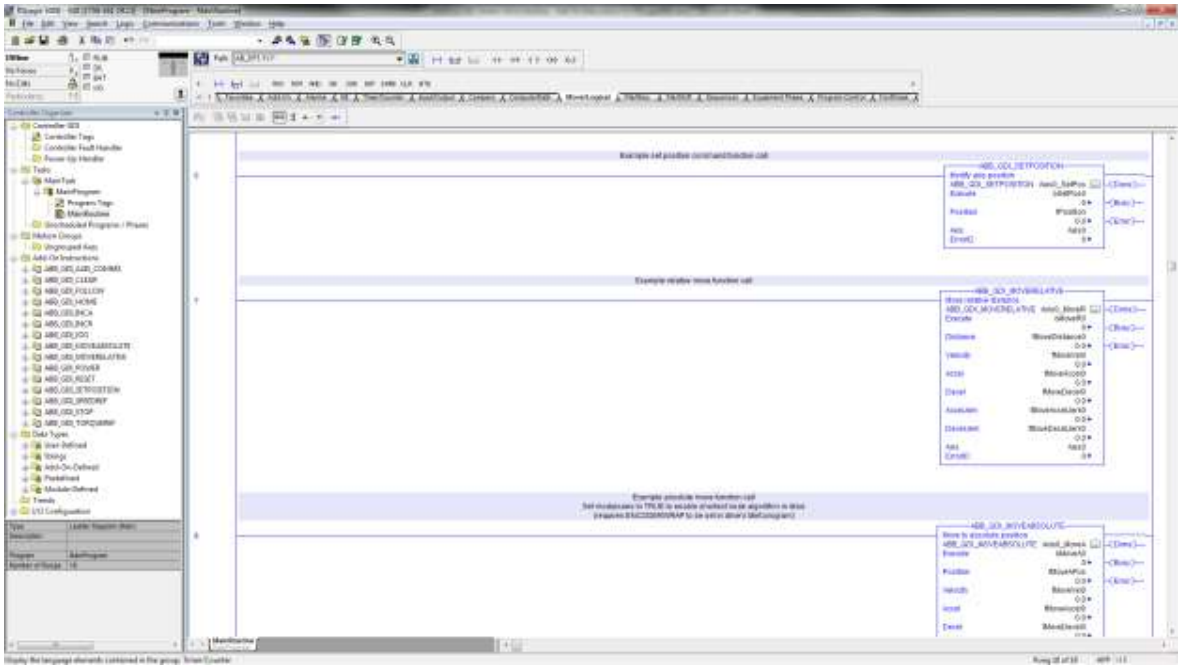
Being able to access this data directly allows great flexibility in the PLC application code (e.g. for an indexing conveyor application the PLC application can access the latch missed status bit (btLatchMissed) and use this to drive a counter that stops motion if a certain number of latches (fast interrupts) are missed in a row).

Communication Watchdog

By default the Mint GDI is configured to use a watchdog mechanism. From receipt of the first message from the PLC the Mint program checks that communication is still active. If this is lost the axis will stop and no further moves will be possible until the error is cleared. It is possible to disable the watchdog (see AN00204 for details), but for completeness an example watchdog for the PLC is shown below (this is included in the example PLC application included with this application note).

Example Application

In addition to the AOI files included with this application note the download also includes an example PLC application. The example PLC application contains the watchdog example, the direct axis structure access example rungs and instances of every single GDI function block (the RSLogix5000 data watch can be used to poke data into the relevant input parameters to exercise these functions).



Contact us

For more information please contact your local ABB representative or one of the following:

new.abb.com/motion
new.abb.com/drives
new.abb.com/drivespartners
new.abb.com/PLC

EtherNet/IP is a trademark of Open DeviceNet Vendor Association.
RSLogix5000, Logix Designer and Studio 5000 are trademarks of Rockwell Software Inc.