Motion Control Products

Application note
Generic drive interface via EtherCAT

AN00234

Rev F (EN)

Ready to use PLC function blocks, combine with pre-written Mint application for simple control of e150, e180 and e190 drives via EtherCAT

Introduction

This application note details how the ‘Generic drive interface’ (GDI) Mint program – a simple drive control profile that can be accessed via any fieldbus that provides access to an e100, e150, e180 or e190 drive’s NETDATA array – can be used in combination with an ABB AC500 PLC running EtherCAT. Note that as EtherCAT is being used this precludes the use of e100 drives in this case.

Please refer to AN00204, available from the support area of the ABB motion website, for a full and comprehensive description of the GDI Mint program and how the various pre-written AC500 motion function blocks operate. This application note assumes version 2.17 or later of the Mint GDI code is being used.

For this application note an example ABB PLC project (for which Automation Builder v1.2 or later is required) is included which utilises the ‘PLCopen motion control’ style function blocks to command motion on the drive. These function blocks are identical to those used by the projects included with AN00204, only the communication layer changes (AN00204 includes code relating to Modbus, this application note illustrates how to utilise EtherCAT).

As the motion is profiled by the drive itself (i.e. distributed motion control) there is no requirement for the PLC project to include the ABB PS552-MC-E motion library, making this a cost-effective solution for applications that only require point to point type motion (e.g. Indexing conveyors, simple pick and place or gantry systems) but still require the high performance of the EtherCAT network (e.g. fast response times or inclusion of ABB’s S500 EtherCAT IO modules in the system).

The example projects (one for MicroFlex e150, one for MotiFlex e180) use a PM591 processor together with a CM579-ETHCAT EtherCAT communication coupler (any AC500 that supports use of the EtherCAT coupler is sufficient, the user should select the processor model based on the overall application requirements).
MicroFlex e150/e190 and MotiFlex e180 servo drives should be running firmware version 5863 or later to use the GDI via EtherCAT.
The PLC processor should be running firmware version 2.4.2 or later and the EtherCAT coupler should be running firmware version 2.6.9 or later – please contact your local ABB support team if you require information on how to check these versions or you require later firmware.

An ESI (XML) file for MicroFlex e190 is included that can be imported into Automation Builder to add the e190 device type if needed. The application note also includes the servo drives package which includes e150 and e180 devices for firmware versions up to and include 5863 – please refer to AN00204 for further details about the servo drives package. In any event, and with any version of Automation Builder, the ESI file can be retrieved from the any e150/e190/e180 drive’s web page and used to import into the Automation Builder device repository if needed.

The sample projects with this application note provides a mechanism for the AC500 PLC to:

- Issue a home command
- Issue a find end stop command
- 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, PDO mapped Netdata locations (drive internal memory). An optional watchdog mechanism is also included, allowing the drive to take action in the event of communication loss (crash stop and disable by default).

Drive setup/configuration

It is only necessary to commission the connected motor (via the Mint Workbench commissioning wizard), being sure to set the drive’s CONTROLREFSOURCESTARTUP parameter to “Direct” as part of this wizard. All other configuration is performed via the GDI Mint program.

Configuring the Generic Drive Interface (GDI) Mint program

The pre-written GDI Mint program only requires a small amount of customisation to suit the user's application. The following sections detail the program sections that should be edited as required.

Mint Startup block

Within the Mint Startup block are the configuration parameters for encoder following (gearing). There are conditionally compiled sections of code for all relevant drive platforms. Edit the section to suit the drive being used in the application…for example,

```
'MicroFlex e150...
#If _platform = _nMicroFlexE150 Then
  MASTERSOURCE(0)  = _msENCODER
  MASTERCHANNEL(0) = 2   '1 = fast inputs encoder, 2 = aux encoder on Universal encoder input
  ENCODERMODE(2)   = 0   'Use 0 or 1 to suit direction
#End If
```

MicroFlex e150/e190 and MotiFlex e180 drives support a wide range of follow modes (refer to the Mint help file for further details).

The final customisation of the Mint Startup block relates to the direction of motor rotation/travel. Editing the MOTORDIRECTION value allows the user to set which direction is considered positive movement…

```
'Setup direction of motor to suit application...
MOTORDIRECTION(0) = 0  'or 1
```

Application Constants

At the beginning of the main program (after the program header) are a set of application related constants…

```
' -------------------------------------------------------------------
' Global Constants
' -------------------------------------------------------------------

' Application specific (edit as required)...
Const _bRemoteEnable As Integer   = _true  'can PLC control enable?
Const _bUseWatchdog As Integer    = _true
Const _nWatchdogTime As Integer   = 2000   'ms
Const _fScale As Float            = 8000   'counts per user unit
Const _nHomeType As Integer       = _hmNEGATIVE_SWITCH
Const _nHomeInput As Integer      = 1      '
Const _nFwdLimit As Integer       = -1     'digital input
Const _nRevLimit As Integer       = -1     'number
Const _nLimitMode As Integer      = _emCRASH_STOP_DISABLE
Const _nStopInput As Integer      = -1     '
Const _nStopMode As Integer       = _smDECEL
Const _nInputLevel As Integer     = 0001   '0=Active low, 1=Active high
Const _nEncoderWrap As Integer    = 0      'Set to 1 for modulo axis
Const _fFolErrorFatal As Float    = 1      'User units – edit to suit application
```

These should be edited as required to suit the application:

Constant	Purpose
_bRemoteEnable	The user can decide whether the PLC has any control of the drive's enabled status or not
_bUseWatchdog	The user can decide whether the drive uses a watchdog mechanism to detect loss of communication from the PLC or not (if set _true then the drive will crash stop and disable in the event of loss of communication)
nWatchdogTime	The user can set an appropriate timeout value (in ms). The PLC needs to repeatedly write to Modbus registers 0/1 (Netinteger0) within this time period (by default the sample PLC programs will toggle a bit within these registers every 500ms – this is also customisable and is described later on)
_fScale	The user can set a scalefactor representing the number of encoder counts produced by the motor for one user unit of travel. The default value of 8000 represents a user unit of "revs" for a BSM60R-240MG servo motor. If using an e150 or e180 demokit to try this application note then these kits use a motor with SmartAbs 17-bit feedback, so set _fScale to 131072 to represent a user unit of 'revs'
_nHomeType	The user can define what type of home sequence the axis performs when asked to datum (refer to the HOME keyword in the Mint help file for a full list of available home types and their associated Mint constant values)
_nHomeInput	Allows the user to specify which of the drive's local digital inputs is to be used as the home sensor input. If a home type is used that does not require an input (e.g. _hmPOSITIVE_INDEX) then set this value to -1.
_nFwdLimit / _nRevLimit	Allows the user to specify which of the drive's local digital inputs are to be used as directional limit inputs. If the application does not require travel limits then set these constants to -1
_nLimitMode	Allows the user to specify how the drive reacts to activation of one of the limit inputs. By default the drive will crash stop and disable (refer to the LIMITMODE keyword in the Mint help file for other options and their associated Mint constant values)
_nStopInput	Allows the user to specify which of the drive's local digital inputs is to be used as the Mint stop input (activation of the Mint stop input results in the Mint application's stop event being processed). If there is no requirement for a stop input then set this constant to -1
_nStopMode	Allows the user to specify how the drive reacts to activation of the stop input. By default the drive will decelerate to standstill at the currently defined deceleration rate (refer to the STOPMODE keyword in the Mint help file for other options and their associated Mint constant values)
_nInputLevel	Allows the user to specify whether inputs are considered to be active high or active low. The default value is specified as a binary value (input 0 is the least significant bit) setting input 0 as active high and inputs 1 and 2 as active low. Depending on which drive is used it may be necessary to extend this binary value to include configuration for further inputs
_nEncoderWrap	If the application is using a modulo axis (e.g. a turntable that requires the axis position to be wrapped in the range 0 to 360 degrees) then this constant should be set to the number of encoder counts in one full cycle of the axis. For non-modulo axes set this constant to zero
_fFolErrorFatal	Sets the magnitude of following error (difference between demand and measured position) that will result in a following error trip. This value is typically set after fine tuning when the user can recognise what may be construed as a "normal" following error during motion. The fatal value is then set somewhere above this "normal" performance

PLC Example Projects

Two example projects to control a single drive (one for MicroFlex e150, one for MotiFlex e180) are included with this application note. Either of these can easily be adapted to suit a MicroFlex e190 if needed. Each project comprises three main elements:

1. Function blocks for each motion command type included in the Mint Generic Drive Interface (GDI). These function blocks are identical for both the EtherCAT example included with this application note and the Modbus examples included with application note AN00204. Please refer to AN00204 for full details on the operation of these function blocks
2. A data interface function block. This code is responsible for routing the application level data (e.g. from the function block usage) to the EtherCAT communication level. This function block is called from a program element (ECAT_INTERFACE.PRG) that executes in synchronism with reception of EtherCAT telegrams
3. PDO mapping of the NETDATA registers used by the Mint GDI program to allow the PLC to read/write the required data

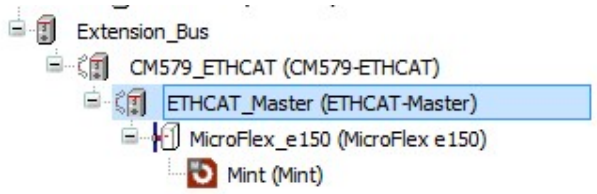
To illustrate the use of the PLC code and the Mint GDI, a visualisation is included with the PLC example projects that allows the user to operate each of the available GDI function blocks and monitor drive status information. This visualisation (Test_Panel) is made up from individual visualisations that have been created for each GDI function. These visualisations are linked to the relevant function block instance in the main program via the replacement placeholder setting for the visualisation.

Adding additional controls for extra axes is as simple as including another visualisation object and configuring its replacement placeholder accordingly. These visualisations can be exported if required and re-used with the PLC project from AN00204. Please refer to the Codesys help file for additional details about the use of the visualisation object, placeholders and exporting / importing project elements.

Creating/Defining Axes

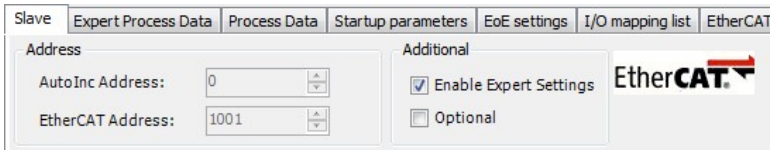
The example PLC program is written to allow one e150/e180 drive to be controlled (via the GDI) over EtherCAT. However, it is very simple to add additional axes.

Using Automation Builder, right click the ETHCAT_Master element of the hardware tree…

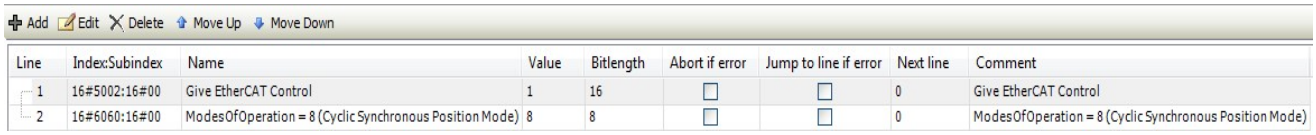


…and then select 'Add object' to add an additional e150 or e180 servo drive. Select the appropriate device from the resulting dialog depending on the firmware version of the drive being used (please refer to application note AN00205 for further details on adding drives to the Automation Builder device repository)

Double-click the new drive entry and from the right hand pane select the 'Slave' tab. On this tab then select the 'Enable Expert Settings' checkbox…



Now select the 'Startup parameters' tab. By default, whenever a new drive is added to the device tree, Automation Builder will add two Startup commands…

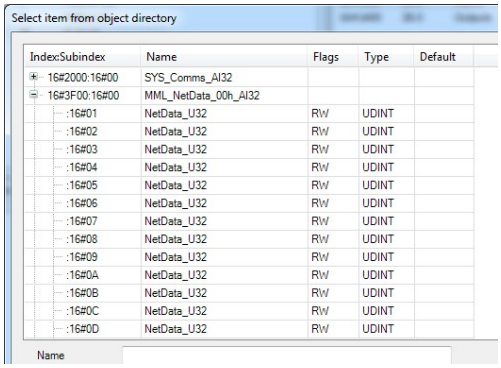


"Give EtherCAT control" will cause the PLC to switch the drive into "Real time Ethernet" mode when the PLC application starts (which we don't want to happen – the drive needs to operate using the "Direct" control reference source).
"Modes of Operation" will cause the PLC to tell the drive to switch to Cyclic Synchronous Position (mode 8) when the PLC application starts (we don't need this command at all).

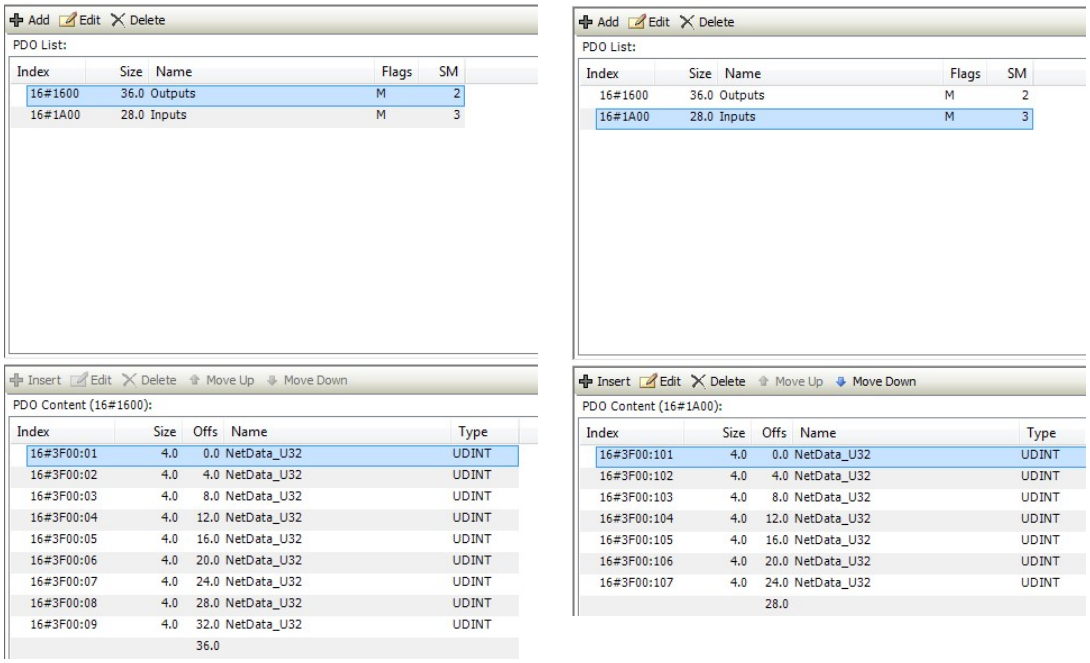
Select each Startup command in turn and click the 'Delete' button to remove it. You should end up with a completely blank list of Startup commands.

Now select the 'Expert Process Data' tab and delete the default PDO mappings that have been made for the newly added drive (by default Automation Builder will add the basic mappings used for Cyclic Synchronous Position control via EtherCAT – these are not required when using the GDI).

Now add process data entries for the GDI – the Output mappings are Netdata 0 to 8, the Input mappings are Netdata 100 to 106. This is done by highlighting Outputs or Inputs in the 'PDO List' frame and then right clicking in the 'PDO Content' frame and selecting 'Insert' to add a new PDO mapping….



Remember that subindex 16#00 for the MML_Netdata object is used for the number of further subindexes available (i.e. the number of Netdata locations that can be mapped), so Netdata location 0 corresponds with subindex 16#01. When the outputs and inputs have been mapped the PDO Content pane should look something like this for each section…



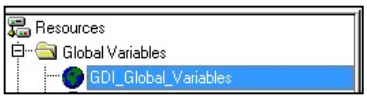
Note that as an alternative to steps 1 to 4 it is also possible to right click the existing drive in the hardware tree, select ‘Copy’ and then right click and select ‘Paste’ to create another instance of the same drive type.

Now select the ‘EtherCAT I/O Mapping’ tab for the new axis/drive. This is where we can give names to the PDO mappings we’ve just added for the additional axis. The screenshot below shows the default names assigned to the single axis used in the example application. It is usually a good idea to use the same naming convention (e.g. just change the 0 on the end to 1 for the next axis etc…)…

Channels

Variable	Mapping	Channel	Address	Type	Unit	Description
pdoCONTROL_WORD0		NetData_U32	%QD1.0	UDINT		NetData_U32
pdoCMD_TYPE0		NetData_U32	%QD1.1	UDINT		NetData_U32
pdoVALUE0		NetData_U32	%QD1.2	UDINT		NetData_U32
pdoSPEED0		NetData_U32	%QD1.3	UDINT		NetData_U32
pdoACCEL0		NetData_U32	%QD1.4	UDINT		NetData_U32
pdoDECEL0		NetData_U32	%QD1.5	UDINT		NetData_U32
pdoACCELJERK0		NetData_U32	%QD1.6	UDINT		NetData_U32
pdoDECELJERK0		NetData_U32	%QD1.7	UDINT		NetData_U32
pdoOFFSET0		NetData_U32	%QD1.8	UDINT		NetData_U32
pdoSTATUS_WORD0		NetData_U32	%ID1.0	UDINT		NetData_U32
pdoMEASURED_POS0		NetData_U32	%ID1.1	UDINT		NetData_U32
pdoMEASURED_VEL0		NetData_U32	%ID1.2	UDINT		NetData_U32
pdoFOL_ERROR0		NetData_U32	%ID1.3	UDINT		NetData_U32
pdoAXIS_MODE0		NetData_U32	%ID1.4	UDINT		NetData_U32
pdoRMS_CURRENT0		NetData_U32	%ID1.5	UDINT		NetData_U32
pdoERROR_CODE0		NetData_U32	%ID1.6	UDINT		NetData_U32

Now double click the GDI (EtherCAT) icon in Automation Builder to launch the Codesys programming environment. Once Codesys has opened, from the “Resources” tab of the Codesys application double-click the “GDI_Global_Variables” icon…

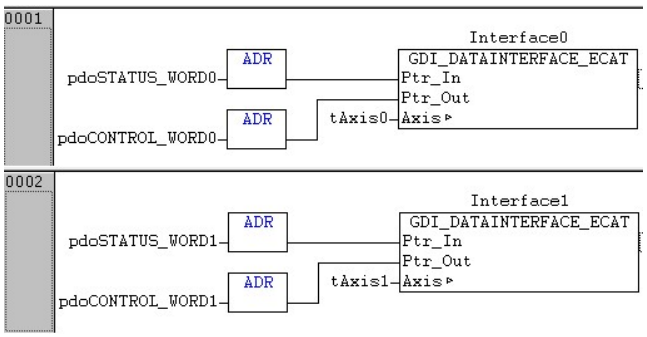


For every additional axis required add an additional declaration for an axis structure of type TGDIAxisRef. The example below shows two additional axes…

```
(*  Add axis data types as required *)
tAxis0 : TGDIAxisRef;
tAxis1 ; TGDIAxisRef;
tAxis2 ; TGDIAxisRef;
```

Now double-click the ECAT_INTERFACE POU icon on the POUs tab within Codesys…

To allow the PLC to exchange control and status data with an additional drive include a call to a new instance of the GDI_DATAINTERFACE_ECAT function block for each additional drive that is required (passing the relevant axis structure and the addresses of the status word and control word PDOs as parameters to this block)….



Add application code, as required, to the PLC program for control of the additional axis (e.g. include new instances of calls to the required GDI function blocks) – please refer to application note AN00204 for full details on the operation of the GDI function blocks

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

© Copyright 2016 ABB. All rights reserved.
Specifications subject to change without notice.