运动控制产品

应用说明
使用 EtherCAT 的通用驱动器接口（GDI）

AN00234

Rev E (CN)

即用型 PLC 功能块，与预先编写的 Mint GDI 程序结合，可通过 EtherCAT 方便的控制 e150、e180 和 e190 驱动器

介绍

本应用说明详细介绍如何与支持 EtherCAT 的 ABB AC500 PLC 一起使用“通用驱动器接口”（GDI）的 Mint 程序（可以通过任何现场总线访问的简单驱动器控制协议。可以通过它访问 e100、e150、e180 或 e190 驱动器的 NETDATA 数组） 。请注意，在使用 EtherCAT 的情况下，不能使用 e100 驱动器。

有关 GDI Mint 程序的完整和全面说明，以及各种预先编写的 AC500 运动功能模块的运行方式，请参阅文档 AN00204。本应用说明假定正在使用的 Mint GDI 程序版本高于 2.17。

本应用说明中包含一个 ABB PLC 项目的示例（需要使用 Automation Builder v1.2 或更高版本）。该项目使用“PLCopen”方式的功能模块来控制驱动器的运动。

除通信层的变化外，这些功能块与 AN00204 附带的项目使用的功能块相同（AN00204 包含与 Modbus 相关的代码，本应用说明说明了 EtherCAT 的使用方式）。

因为是由驱动器本身（即分布式运动控制）对运动进行控制，因此 PLC 项目不需要包含 ABB PS552-MC-E 运动库，这使其成为仅需要点对点类型运动（例如转位输送机、简单的取放或龙门架系统），但仍需要 EtherCAT 网络的高性能（例如快速响应时间或在系统中包含 ABB 的 S500 EtherCAT IO 模块）的应用的经济高效的解决方案。

示例项目（一个用于 MicroFlex e150，一个用于 MotiFlex e180）使用 PM591 处理器和 CM579-ETHCAT EtherCAT 通信通讯模块（任何支持使用 EtherCAT 通讯模块的 AC500 都满足要求，用户应根据整体应用要求选择处理器型号）。

MicroFlex e150/e190 和 MotiFlex e180 伺服驱动器应运行 5863 或更高版本的固件，以通过 EtherCAT 使用 GDI。PLC 处理器应运行 2.4.2 或更高版本的固件，EtherCAT 通讯模块应运行 2.6.9 或更高版本的固件 - 如果您需要有关如何查看这些版本的信息或需要更新的固件，请联系您当地的 ABB 支持团队。

MicroFlex e190 的 ESI（XML）文件，可以导入 Automation Builder 以根据需要添加 e190 设备。应用说明还包括伺服驱动器包，其中包括用于 5863 和更早固件版本的 e150 和 e180 设备 - 有关伺服驱动器包的更多详细信息，请参阅 AN00204。，对任何版本的 Automation Builder，都可以从 e150/e190/e180 驱动器的网页中检索 ESI 文件，并根据需要将其导入 Automation Builder 的设备库中。

本应用说明的项目示例为 AC500 PLC 提供了一种机制来：

- 发送一条寻零命令
- 发送一条查找终点停车命令
- 发送一条相对定位命令
- 发送一条绝对定位命令
- 发送一条增量相对定位命令（可选择在经过“快速锁存”位置一段设定距离后停车）
- 发送一条增量绝对定位命令（可选择在经过“快速锁存”位置一段设定距离后停车）
- 为增量式运动设置一个偏移目标（即相对于记录的快速中断位置来定位轴）
- 点动轴
- 设置轴位置
- 发送一个速度给定值
- 发送一个转矩给定值
- 使能/停用轴
- 使能/停用硬限位
- 复位轴错误
- 在轴上执行一次受控停车或紧急停车
- 使轴与第二个编码器输入相关联
- 为所有运动设置速度、加速时间、减速时间和 S 曲线
- 控制旋转轴或非旋转轴

同时，PLC 还能监视来自驱动器的状态信息，包括：

- 使能状态
- 准备进入使能状态
- 空闲状态
- 到达目标位置状态
- 电机抱闸状态
- 是否完成寻零
- 正向限位状态
- 反向限位状态
- 故障状态
- 停车输入状态
- 错过快速锁存中断的指示
- 寻相状态
- 错误代码
- 测量位置
- 测量速度
- 跟随误差
- 轴运行模式
- 均方根电流

这一切都是通过 PLC 的 PDO 映射 Netdata 位置（驱动器内部存储器）来实现的。

其中还包括一个可选的看门狗机制。它允许驱动器在通信丢失时执行操作（默认情况下为急速停车和停用）。

驱动器设置/配置

只需要调试连接的电机（通过 Mint Workbench 调试向导），确保在该向导的一部分操作中将驱动器的 CONTROLREFSOURCESTARTUP 参数设置为“Direct”。所有其他配置都通过 GDI Mint 程序进行。

配置通用驱动器接口（GDI）Mint 程序

预先编写的 GDI Mint 程序只需要少量的定制，就能匹配用户的应用。以下部分详细说明了应根据需要编辑的程序部分。

Mint 启动块

Mint 启动块中有编码器跟随（联动）的配置参数。所有相关驱动器平台都有条件编译的代码段。编辑这些代码段可以匹配应用程序中使用的驱动器，例如：

```
'MicroFlex e150...
#If _platform = _nMicroFlexE150 Then
  MASTERSOURCE(0)  = _msENCODER
  MASTERCHANNEL(0) = 2    '1 = 快速输入编码器， 2 = 通用编码器输入通道上的辅助编码器
  ENCODERMODE(2)   = 0    ‘使用与方向匹配的 0 或 1
#End If
```

MicroFlex e150/e190 和 MotiFlex e180 驱动器支持多种跟随模式（有关详细信息，请参阅 Mint 帮助文件）。

Mint 启动块的编写与电机旋转/行进的方向有关。编辑 MOTORDIRECTION 值使用户能够设置被认为是正向移动的方向。

```
'设置电机的方向以匹配应用。
MOTORDIRECTION(0) = 0   ‘或 1
```

应用常量

在主程序开始部分（程序头之后）是一组与应用相关的常量 …

```
' ------------------------------------------------------------------------
' 全局常量
' ------------------------------------------------------------------------

' 应用专用（按照需要编辑）...
Const _bRemoteEnable As Integer    = _true   '能否启用 PLC 控制?
Const _bUseWatchdog As Integer     = _true
Const _nWatchdogTime As Integer    = 2000    'ms
Const _fScale As Float             = 8000    '每个用户单位的计数
Const _nHomeType As Integer        = _hmNEGATIVE_SWITCH
Const _nHomeInput As Integer       = 1       '
Const _nFwdLimit As Integer        = -1      '数字输入
Const _nRevLimit As Integer        = -1      '数字
Const _nLimitMode As Integer       = _emCRASH_STOP_DISABLE
Const _nStopInput As Integer       = -1      '
Const _nStopMode As Integer        = _smDECEL
Const _nInputLevel As Integer      = 0001    '0=低电平有效, 1=高电平有效
Const _nEncoderWrap As Integer     = 0       '对旋转轴，设置为 1
Const _fFolErrorFatal As Float     = 1       '用户单位 - 编辑以匹配应用
```

应该按要求编辑这些参数以匹配应用：

常量	用途
_bRemoteEnable	用户可决定 PLC 是否对驱动器的使能状态有控制能力
_bUseWatchdog	用户可决定驱动器是否使用看门狗机制来检测 PLC 通讯的丢失（如果设置为_true，则驱动器会在通讯丢失时紧急停车和停用）
nWatchdogTime	用户可设置合理的超时值（以 ms 为单位）。PLC 需要在本时间段内（默认情况下，样本 PLC 程序将每 500ms 切换这些寄存器中的一位-该操作可定制，在后文有说明）反复向 Modbus 寄存器写入 0/1（Netinteger0）
_fScale	用户可设置比例因数，代表一个用户行程单位中由电机产生的编码器计数数量。默认值为 8000，代表 BSM60R-240MG 伺服电机的用户单位“转”。如果使用 e150 或 e180 演示套件来尝试使用本应用说明，那么这些套件使用的是具有 SmartAbs 17 位反馈的电机，因此将_fScale 设置为 131072，代表用户单位“转”。

_nHomeType	用户可定义轴执行的寻零序列的类型（参考 Mint 帮助文件中的 HOME 关键字，以了解完整的可用寻零类型列表及其相关的 Mint 常量值）
_nHomeInput	允许用户指定把驱动器的哪一个本地数字输入用作寻零传感器输入。如果使用的寻零类型不需要输入（比如，_hmPOSITIVE_INDEX），则把这个值设置为-1。
_nFwdLimit/nRevLimit	允许用户指定把驱动器的哪些本地数字输入用作定向限位输入。如果应用不需要行程限位，则把这些常量设置为-1
_nLimitMode	允许用户指定驱动器对其中一个限值输入被激活的响应方式。默认情况下，驱动器将紧急停车并停用（参考 Minit 帮助文件中的 LIMITMODE 关键词，以了解其它选项及其相关的 Mint 常量值）
_nStopInput	允许用户指定把驱动器的哪些本地数字输入用作 Mint 停车输入（Mint 停车输入激活后，将处理 Mint 应用程序的停车事件）。如果没有对停车输入的要求，则把该常量设置为-1
_nStopMode	允许用户指定驱动器对停车输入激活的响应方式。默认情况下，驱动器将按照当前定义的减速速率减速直到静止（参考 Minit 帮助文件中的 STOPMODE 关键词，以了解其它选项及其相关的 Mint 常量值）
_nInputLevel	允许用户指定是把输入视为高电平有效还是低电平有效。默认值被指定为二进制值（输入 0 是最低有效位），把输入 0 设置为高电平有效，把输入 1 和 2 设置为低电平有效。根据使用的驱动器，可能需要扩展此二进制值以实现包括更多输入的配置。
_nEncoderWrap	如果应用使用旋转轴（比如，转盘要求在 0 到 360 度的范围内），则本常量应该被设置为一个完整的轴循环内的编码器计数。对非旋转轴，把本常量设置零。
_fFolErrorFatal	设置将导致跟随误差错误的跟随误差量级（需求和测量位置之间的差值）。这个值一般在精细参数整定之后，用户能够明确运动中可被视为“正常”跟随误差的条件时设置。然后，把致命值设置为略高于这个“正常”执行状态

PLC 项目示例

本应用说明包含两个控制单个驱动器的示例项目（一个用于 MicroFlex e150，一个用于 MotiFlex e180）。如果需要，可以很容易地调整其中任何一个项目，以匹配 MicroFlex e190。每个项目包括三个主要元素：

1. Mint 通用驱动器接口（GDI）中包括的每个运动控制类型的功能块。对本应用说明中包含的 EtherCAT 示例，以及应用说明 AN00204 中包含的 Modbus 示例，这些功能块都是相同的。有关这些功能块的详尽操作信息，请参阅 AN00204。
2. 数据接口功能块。本代码负责把应用层数据（比如，从功能块使用）发送到 EtherCAT 通信层。从程序（ECAT_INTERFACE.PRG）调用此功能块，该程序元素与 EtherCAT 报文的接收同步执行。
3. 通过 Mint GDI 程序使用的 NETDATA 寄存器的 PDO 映射，PLC 可读取/写入所需的数据。

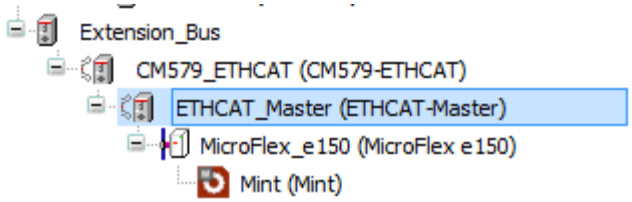
为了说明 PLC 代码和 Mint GDI 的使用方法，PLC 示例项目中包含了 Visu。它允许用户操作每个可用的 GDI 功能块并监视驱动器的状态信息。Visu（Test_Panel）由为每个 GDI 函数创建的各个 Visu 组成。这些 Visu 通过为其设置的替代占位符与主程序中的相关功能块实例链接。

为额外的轴添加附加控件非常简单，只需要增加另一个 Visu 对象，并相应地配置其替代占位符。如果需要，可以导出这些 Visu，并与 AN00204 的 PLC 项目一起使用。
有关使用 Visu、占位符和导出/导入项目元素的其他详细信息，请参阅 Codesys 帮助文件。

创建/定义轴

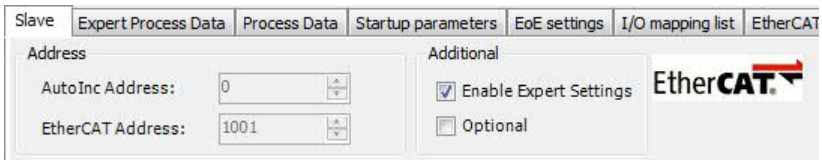
编写的 PLC 示例程序用于通过 EtherCAT 控制一台 e150/e180 驱动器（通过 GDI）。但是，可以方便地为它添加更多的轴。

使用 Automation Builder，右键单击硬件树的 ETHCAT_Master 元素。

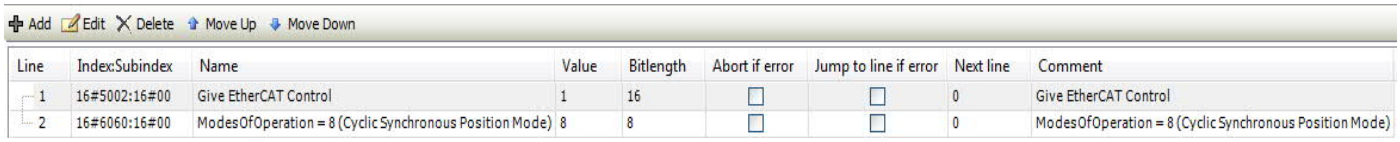


然后，选择“Add object”以添加额外的 e150 或 e180 伺服驱动器。根据所使用的驱动器的固件版本，从结果对话框中选择适当的设备（有关将驱动器添加到 Automation Builder 设备库的更多信息，请参阅应用笔记 AN00205）

双击新驱动器条目，然后从右侧窗格中选择“Slave”选项卡。在此选项卡上，然后选中“Enable Expert Settings”复选框。



现在选择“Startup parameters”选项卡。默认情况下，每当将新驱动器添加到设备树时，Automation Builder 将添加两个启动参数。

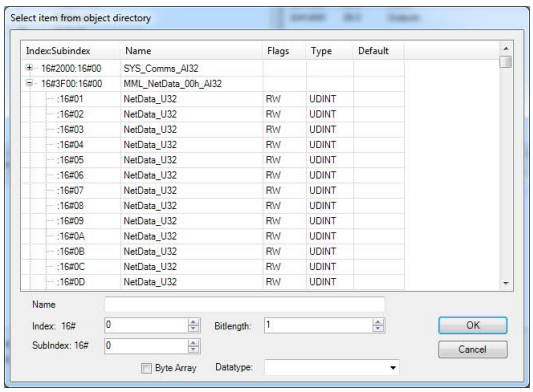


“Give EtherCAT control”将导致 PLC 在 PLC 应用程序启动时将驱动器切换到“实时以太网”模式（我们不希望发生这种情况 - 驱动器需要使用“直接”控制给定值信号源进行操作）。
当 PLC 应用程序启动时，“Modes of Operation”将使 PLC 通知驱动器切换到循环同步位置（模式 8）（我们根本不需要此命令）。

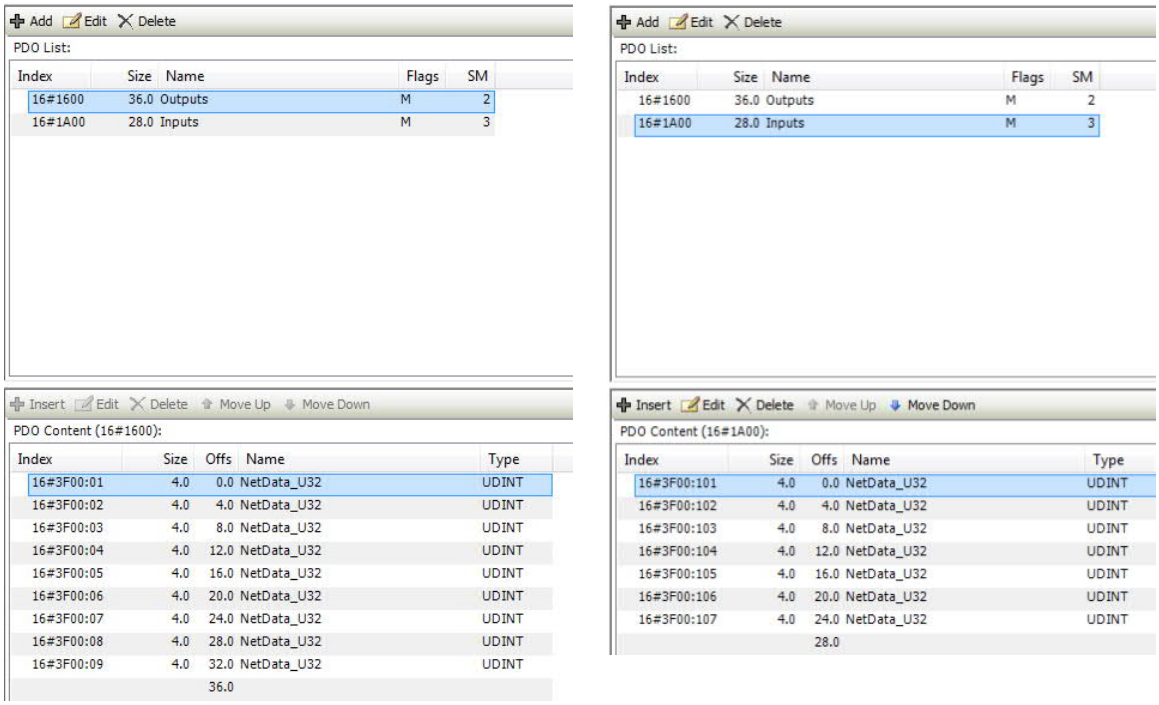
依次选择每条启动命令，然后单击“Delete”按钮将其删除。最后，您应该得到一个完全空白的启动命令列表。

现在选择“Expert Process Data”选项卡并删除为新添加的驱动器创建的默认 PDO 映射（默认情况下，Automation Builder 将添加用于通过 EtherCAT 进行循环同步位置控制的基本映射，但使用 GDI 时不需要这些映射）。

现在为 GDI 添加过程数据条目 - 输出映射是 Netdata 0 到 8，输入映射是 Netdata 100 到 106。要完成添加，突出显示“PDO List”框架中的 Outputs 或 Inputs，然后右键单击“PDO Content”框架并选择“Insert”以添加新的 PDO 映射。

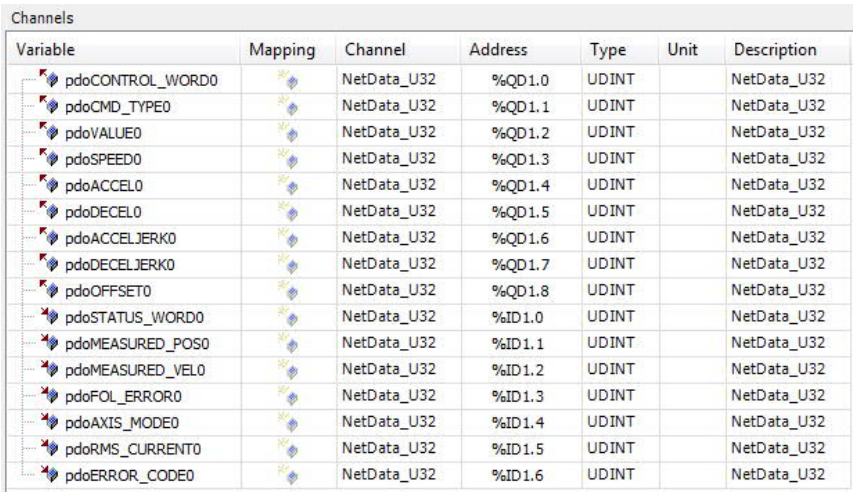


请记住，MML_Netdata 对象的子索引 16#00 用于可用的其他子索引的数量（即可以映射的 Netdata 位置的数量），因此 Netdata 位置 0 对应于子索引 16#01。在映射完输出和输入后，PDO Content 窗格的各个部分看起来应该与下图相似。

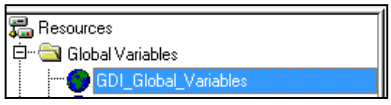


请注意，作为步骤 1 到 4 的替代方法，还可以右键单击硬件树中的现有驱动器，选择“复制”，然后右键单击并选择“粘贴”以创建相同驱动器类型的另一个实例。

现在为新轴/驱动器选择“EtherCAT I/O Mapping”选项卡。在这里，我们可以为刚为附加轴添加的 PDO 映射命名。下面的屏幕截图显示了为应用程序示例中使用的单个轴分配的默认名称。通常，好的做法是使用相同的命名规则（例如，只需将末尾的 0 更改为 1 以用于下一个轴等）。



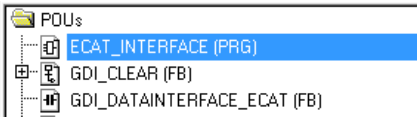
现在，双击 Automation Builder 中的 GDI（EtherCAT）图标以启动 Codesys 编程环境。在 Codesys 打开后，在 Codesys 应用程序的“Resources”选项卡中双击“GDI_Global_Variables”图标。



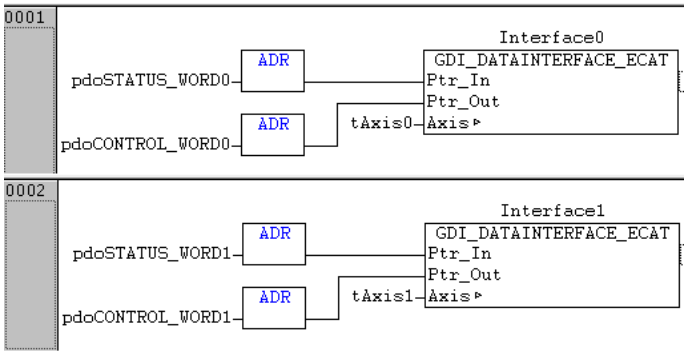
为每条额外的轴另外添加一条 TGDIAxisRef 型轴结构的声明。下文的示例显示了两个额外的轴...

```
        （*按照要求添加轴数据类型*）
    tAxis0：TGDIAxisRef；
    tAxis1：TGDIAxisRef；
    tAxis2：TGDIAxisRef；
```

现在双击 Codesys 中 POUs 选项卡上的 ECAT_INTERFACE POU 图标。



为了使 PLC 与驱动器交换控制和状态数据，为每个驱动器调用 GDI_DATAINTERFACE_ECAT 功能块的新实例（把相关的轴结构以及状态字和控制字 PDO 的地址作为参数传递给传递这个块）。



按照要求把应用代码添加到 PLC 程序，以控制额外的轴（比如，包括调用所需要的 GDI 功能模块的新实例） - 请参考应用说明 AN00204 了解 GDI 功能块操作的详尽信息。

联系我们

欲了解更多信息，请联系当地的 ABB 代表，
或以下任意一种方式：

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