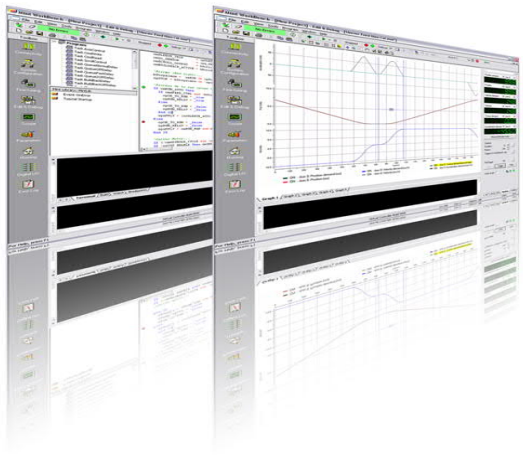
运动控制产品

应用说明
PID 控制器

AN00208
Rev B (CN)

利用 Mint 的灵活性来设计和编写您的自定义 PID 算法，用于放卷、复卷和温度控制等应用



引言

ABB 伺服产品在处理运动轴的转矩、速度和位置控制时采用 PID 控制环。这些算法使用驱动电流或电机位置/速度作为反馈。如果只想控制基本轴参数，这些就足够了。

在某些 ABB 伺服产品上，可以使用称为“保持模拟”或“HTA”的附加运动功能。HTA 实际上是允许自动调整轴*位置*，使其试图将模拟输入维持在预编程的设定点/水平的 PID（比例，积分和微分）控制器。例如，它可用于控制等离子或激光切割机上 Z 轴的高度/位置。其中，实际高度通过模拟输入反馈。

有时候，可能需要控制一些与轴位置无关的其他过程变量。我们可能希望根据模拟输入的值控制模拟输出或控制轴的速度/转矩，以便尝试将模拟输入维持在预设值。或者，我们可能正在控制 HTA 形式的轴，但需要比标准集成 HTA 算法更多的控制/灵活性。出于这些目的，本文档将介绍如何在 Mint 中实现自定义 PID 控制算法。

PID 控制是如何工作的

本节是对 PID 控制器功能的简要介绍。对如何设置 PID 控制器的详细说明不在本文档的范围。有很多免费提供的文献可供参考。

PID 控制器计算测量（输入）过程变量和预期设定点之间的“误差”值。控制器通过调整过程控制输出来尝试尽量减小此“误差”值，而过程控制输出又会影响到测量（输入）过程变量值。

PID 控制器算法涉及三个独立的增益参数：比例、积分和微分值，分别表示为 Kp，Ki 和 Kd。这些增益跟踪了误差随时间变化的方式。Kp 跟踪*当前*误差，Ki 跟踪累积误差，Kd 是基于当前变化率对*未来*误差的预测。这三个部分综合作用，调整 PID 输出值。

*比例*项产生的输出值与当前误差值成比例。对于给定的误差变化，高比例增益会导致输出发生大的变化。如果比例增益太高，系统可能会变得不稳定。比例增益对输出变化的影响最大。

*积分*项是随时间变化的误差之和。积分项消除了纯比例控制器产生的残余稳态误差。它也适用于把过程变量加速到设定点。如果设置得太高，可能会导致过程过冲或围绕设定值震荡。

*微分*项抑制误差的快速变化。它可减少积分项产生的过冲。高设置会降低控制器的响应灵敏度。应注意反馈信号是否有噪声，因为微分项将比其他增益项更大程度地放大噪声。

在 Mint 中为 PID 控制器编码

示例 1：即使在误差为零时，需要过程控制的系统也有输出（例如，卷轴的开卷复卷，恒定张力控制）

为了使代码更易于重复使用，PID 控制器的数据保存在一个结构体类型的变量中。每次在使用时，Structure 变量都会传递给 doPID 函数。以下结构体定义应放在程序的顶部。

```
Structure TPID
‘设置参数
PIDMaxOut As Float
PIDMinOut As Float
PIDKP As Float
PIDKI  As Float
PIDKILimit As Float
PIDKD As Float
PIDLoopTime As Float
‘内部变量
tTime As Time
fError As Float
fDerivative As Float
fErrorLast As Float
fIntegral As Float
fUnlimitedPidOut As Float
End Structure
```

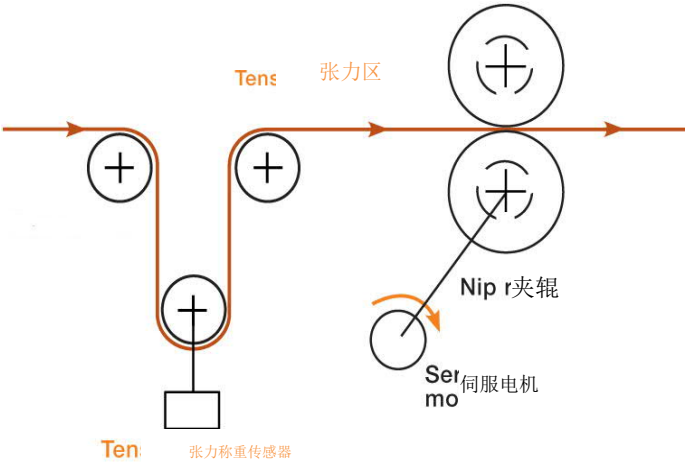
现在应该将 doPID 函数放在程序中：

```
Function doPID(ByRef PIDin As TPID, ByVal fTarget, ByVal fFeedback, ByVal fCurrentVal) As Float
Dim fKIntLimit

‘等待伺服循环计时器到期
Pause PIDin.tTime > PIDin.PIDLoopTime
PIDin.tTime = 0

‘计算 PID 误差
PIDin.fError =  fTarget - fFeedback

‘计算新的积分值并限制积分影响
If Sgn(PIDIn.fError) <> Sgn(PIDIn.fErrorLast) Then PIDin.fIntegral = 0
PIDin.fIntegral = PIDin.fIntegral + PIDin.fError
fKintLimit = ((hmiPID_KILIMIT / 100) * PIDin.PIDMaxOut)
If (PIDin.fError > 0) And (PIDin.fIntegral > fKIntLimit) Then PIDin.fIntegral = fKIntLimit
If (PIDin.fError < 0) And (PIDin.fIntegral < -fKIntLimit) Then PIDin.fIntegral = -fKIntLimit
```

```
‘计算微分值
PIDin.fDerivative = PIDin.fError - PIDin.fErrorLast
PIDin.fErrorLast = PIDin.fError

‘计算新的 PID 输出
PIDin.fUnlimitedPidOut = (PIDin.PIDKP * PIDin.fError) + _
(PIDin.PIDKI * PIDin.fIntegral) + _
(PIDin.PIDKD * PIDin.fDerivative) + _
fCurrentVal  ‘对于位置以外的 PID 模式，请包含此行

‘范围检查 PID 输出
If (PIDin.fUnlimitedPidOut > PIDin.PIDMaxOut ) Then
doPID = PIDin.PIDMaxOut
ElseIf (PIDin.fUnlimitedPidOut < PIDin.PIDMinOut) Then
doPID = PIDin.PIDMinOut
Else
doPID = PIDin.fUnlimitedPidOut
EndIf

End Function
```

示例函数将 PID 增益的影响添加到现有的过程输出值中。这种形式的代码将用于，例如，轴需要保持施加*速度*（或*转矩*），以将过程输入保持在预期值的系统中。对通过改变轴*位置*来控制过程的系统，一旦达到所需位置，PID 输出可能会降低到零，因此在这些情况下不需要包括当前过程输出值。

如右图所示，我们通过将转矩给定值施加到运行夹辊的伺服电机上，来控制幅材中的张力。通过伺服电机自身的电流环，我们就可以实现良好的开环张力控制。但如果我们能够得到实际张力的反馈值，我们会取得更好的结果。



闭环控制将使用张力臂或张力传感器来测量幅材中的张力。夹辊上的张力等于电机转矩和夹辊半径的乘积，在这种情况下，夹辊半径是固定的，并且电机具有固定的转矩常数。通在考虑电机和夹辊之间的传动装置后，我们可以得出一个换算系数，以便将电机转矩换算出材料张力（反之亦然）。

我们通过运动控制器（或智能驱动器）的模拟输入读取张力传感器的张力反馈，获得的读数在 0 到 10Vdc（Mint 中 0 到 100%）之间。

我们需要定义用于读取张力传感器模拟输入通道，和用于将模拟量转换成牛顿力/张力的比例因子。我们需要定义一个固定夹辊半径的常量，和另一个用于电机转矩常量系数（需考虑中间机械传动环节）。还需定义张力设定点和过程夹棍张力。

```
Const _fLoadCellScale As Float = 0.5 ‘0-100% = 0-50N
Const _fNipRollRadius As Float = 0.1 ‘m
Const _fMotorTorqueConstant As Float = 3.5 ‘Nm/A

Define aiLoadCell = (ADC(0) * _fLoadCellScale)

Dim fTensionSP As Float = 30 ‘N
Dim fNipTensionDemand As Float
Dim fPIDOutput As Float
```

PID 数据结构允许为积分项设置一些增益（可以在应用程序开发过程中按需调节）和限值。

```
TensionPID.PidKProp     = 1.0
TensionPID.PidKInt      = 0.05
TensionPID.PidKIntLimit = 15
TensionPID.PidKDeriv    = 0.05
```

然后在循环中重复调用 PID 控制器。PID 回路的输出给出扭矩参考值：

```
Loop
fNipTensionDemand = (DRIVERATEDCURRENT * _fMotorTorqueConstant * TorqueRef(_axNip) )/ (100 * _fNipRollRadius))
fPIDOutput = doPID( TensionPID, fTensionSP, aiLoadCell, fNipTensionDemand )
TorqueRef( _axNip ) = 100 * (fPIDOutput * _fNipRollRadius) / (_fMotorTorqueConstant * DRIVERATEDCURRENT)
End Loop
```

请注意，Mint 关键字 DRIVERATEDCURRENT 仅适用于智能驱动器。因此，对使用运动控制器的张力控制系统，此关键字将被替换为另一个硬编码常量。

本文档（Tension PID Control.mnt）配套 Mint 示例程序。为简单起见，其中使用了单个常量。它包含上面详述的所有系统变量（转矩常量，夹辊半径，驱动器额定电流）。如上文所示，如果需要弄清楚转矩给定值是如何算出的，用户可能希望对其进行扩展。还可以强化示例代码的功能，以包含补偿更复杂的因素，例如：

- 电缆盘直径补偿或锥度张力方案
- 惯性补偿
- 摩擦补偿

示例 2：一旦误差为零，不再需要过程控制输出的系统（例如，定位轴 - 一旦轴到位，速度给定值变为零）

该示例在操作上类似于 Mint HTA 功能，但是 HTA 需要模拟输入作为反馈，该示例使用轴位置本身作为过程反馈。在该示例中，系统通过 NETFLOAT（0）接收“请求位置”。代码对此请求位置（在 0-255 范围内）进行换算，以便实现的实际位置在 0 到 90 个用户单位范围内。然后，它使用 PID 回路生成速度/速度给定值，将轴移动到所需位置。

通过 PID 结构的成员变量设置的最大和最小速度给定值的限值 ...

```
PositionPID.PIDMaxOut   = 50  ‘速度 单位/秒
PositionPID.PIDMinOut   = -50 ‘速度 单位/秒
```

与示例 1 一样，为 PID 控制器采用“Structure”变量类型。然后，每次在使用 structure 变量时，都将其传递给 doPID 函数。因为一旦误差为零，我们就不再需要保持过程输出，所以不需要将当前进程输出值传递给 doPID 函数，因此该示例使用的此函数形式稍有不同 ...

```
Function doPID(ByRef PIDin As TPID, ByVal fTarget, ByVal fFeedback) As Float
Dim fKIntLimit

‘等待伺服循环计时器到期
Pause PIDin.tTime > PIDin.PIDLoopTime
PIDin.tTime = 0

‘计算 PID 误差
PIDin.fError =  fTarget - fFeedback
```

```
‘计算新的积分值并限制积分影响
If Sgn(PIDIn.fError) <> Sgn(PIDIn.fErrorLast) Then PIDin.fIntegral = 0
PIDin.fIntegral = PIDin.fIntegral + PIDin.fError
fKintLimit = ((hmiPID_KILIMIT / 100) * PIDin.PIDMaxOut)
If (PIDin.fError > 0) And (PIDin.fIntegral > fKIntLimit) Then PIDin.fIntegral = fKIntLimit
If (PIDin.fError < 0) And (PIDin.fIntegral < -fKIntLimit) Then PIDin.fIntegral = -fKIntLimit

‘计算微分值
PIDin.fDerivative = PIDin.fError - PIDin.fErrorLast
PIDin.fErrorLast = PIDin.fError

‘计算新的 PID 输出
PIDin.fUnlimitedPidOut = (PIDin.PIDKP * PIDin.fError) + _
(PIDin.PIDKI * PIDin.fIntegral) + _
(PIDin.PIDKD * PIDin.fDerivative)

‘范围检查 PID 输出
If (PIDin.fUnlimitedPidOut > PIDin.PIDMaxOut ) Then
doPID = PIDin.PIDMaxOut
ElseIf (PIDin.fUnlimitedPidOut < PIDin.PIDMinOut) Then
doPID = PIDin.PIDMinOut
Else
doPID = PIDin.fUnlimitedPidOut
EndIf

End Function
```

主程序循环简单地将 PID 结构、请求位置（换算为用户单位）和测量位置传递给函数 ...

```
Loop
If ipStart Then
‘从 HMI 更新 PID 控制器增益值
PositionPID.PIDKP      = hmiPID_KP
PositionPID.PIDKI      = hmiPID_KI
PositionPID.PIDKILimit = hmiPID_KILIMIT
PositionPID.PIDKD      = hmiPID_KD
‘使用 PID 回路调整目标轴的速度给定值
VELREF(_axX) = doPID(PositionPID, cmPOS_DEMAND * _fPositionScale, POS(_axX))
Else
CANCEL(_axX)
Pause(IDLE(_axX))
End If
End Loop
```

用户只需要调整最大和最小过程输出（在本例中为速度给定值），轴的加速和减速率以及 PID 回路的比例、积分和微分项的限值，就可以获得所需的响应。本文档包含一个 Mint 示例程序（Position PID Control.mnt）以供参考。

联系我们

要了解更多信息，请联系您的当地的 ABB 代表，或以下一种方式：

new.abb.com/drives/low-voltage-ac/motion
new.abb.com/drives
new.abb.com/channel-partners
new.abb.com/plc