运动控制产品

# 应用说明
# 离合装置的持续跟随

AN00100

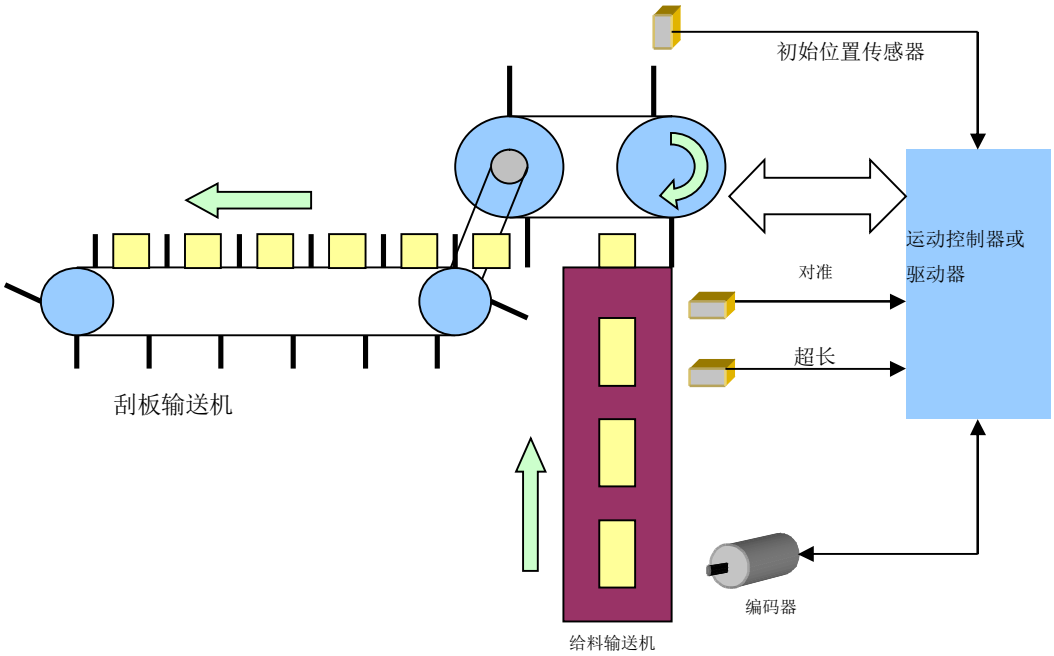
Rev D (CN)

智能伺服控制系统通常用于替代纯粹通过机械传动装置驱动的机构。使用电子系统可以始终保持精确的位置锁定，并且只需按一下按钮，即使在运转中也可以随意修改齿轮比。甚至可以模拟机械离合系统。机械系统在离合器装置接合和脱离期间行进的距离可能无法精确预测，而电子系统则可以精确地指定运行距离。

## 引言

Mint 提供了一个简单易用的 FOLLOW 关键字，允许轴相对于主轴或编码器以指定的齿轮比运行。FOLLOW 关键字可以配置为多种预定义模式。其中一种模式允许在更改跟随比（持续或扩展离合装置模式）时定义离合距离。对轴进行调相，以允许产品对准，同时还可以执行例如使用 Mint OFFSET 命令进行跟随等操作。

下面的插图显示了一台机器的布局，在这种模式下，连续离合跟随非常有用（用于在互相垂直的输送机之间推送产品的挡板）：




另一个使用离合跟随（允许轴被啮合，然后停在已知位置）系统的例子是螺旋进料系统，把产品分隔成等间距，送给下游系统。

在上面的挡板推料系统中，产品从生产线送入给料输送机。生产线输出的产品间距相等，是产品长度的 50%。

给料输送机驱动编码器，以便确定和跟踪产品位置（编码器与运动控制器或智能驱动器上的主/辅助编码器输入相连）。

在产品接近机器的挡板推料部分时，有两个光电传感器对产品进行检测。离挡板最近的传感器用作“位置锁定”输入（快速输入）。当检测到产品的前边缘时，该输入快速捕获挡板的位置，以使挡板能够对准每个产品。

第二个传感器用于检测超长产品（例如，两个产品连接在一起）。调节该传感器以适应产品长度，当两个传感器同时 ON 时，则认定存在超长产品。

有一个后挡直接与给料输送机（在挡板推料区后面）对齐，后挡被兼用作分选门。由数字输出控制的电磁阀用于打开/关闭后挡，以便在需要时可以剔除超长产品。

BSM 伺服电机用于驱动挡板。机械联轴器还允许刮板输送机被同一电机驱动。通过这样的设计，使得一次产品推动与一个产品刮板关联起来。

接近传感器允许挡板轴在启动时或急停后寻零。

由于挡板轴通常必须跟随给料输送机编码器，必须能够在需要时调整位置以进行对准、停止和重新启动。它是“持续离合模式跟随”的理想候选方案。

## 挡板/推动轴

挡板轴（轴 0）的一般操作原理是，对接收的每个产品，挡板必须运行一个挡板间距。这种关系可以被表述为***编码器跟随***。可以通过如下公式确定跟随比率：

**跟随比率=挡板间距/产品间距**

（其中挡板间距和产品间距以用户单位...表示。用户单位可以是正交编码器计数，或者可以决定同时换算挡板推板轴和主编码器的用户单位为 mm，以表示线性行程）。

在当前的例子中，我们假设挡板轴和主编码器的用户单位为编码器数（即每个的比例因子为 1）。如果我们机器上的挡板间距为 12000 个计数，而产品间距（产品前缘之间的距离）为 4000 个计数，则跟随比率为 3。

挡板之间的距离是固定的，因此我们设置此轴的 ENCODERWRAP 以匹配该距离（如：12000）。

## 启动挡板

首次启动机器时，必须对挡板预先定位，以便：

a) 最初可以通过分选门剔除可编程的产品数量
b) 挡板可以沿斜坡加速，以避免机械冲击和降低峰值扭矩要求

该位置被视为零位置，可在上电时或急停后通过寻零过程完成。寻零过程首先寻找初始位置传感器，在找到初始位置传感器后，还可以执行预编程的偏移运动。
一旦剔除所需的产品数量，挡板需要按跟随比率加速至跟随速度。这可以通过使用 Mint 代码的“扩展离合模式”的跟随设置来实现；

```
FOLLOWMODE(0) = _fmContinuous_Clutch
```

此模式允许用户指定从轴沿斜坡上升到跟随比率的主轴距离。可以把它视为飞剪。同时，如果需要的话，Mint 甚至允许将该运动指定为 FLY 段（而不是指定跟随比）。

飞剪段的通用公式如下所示：

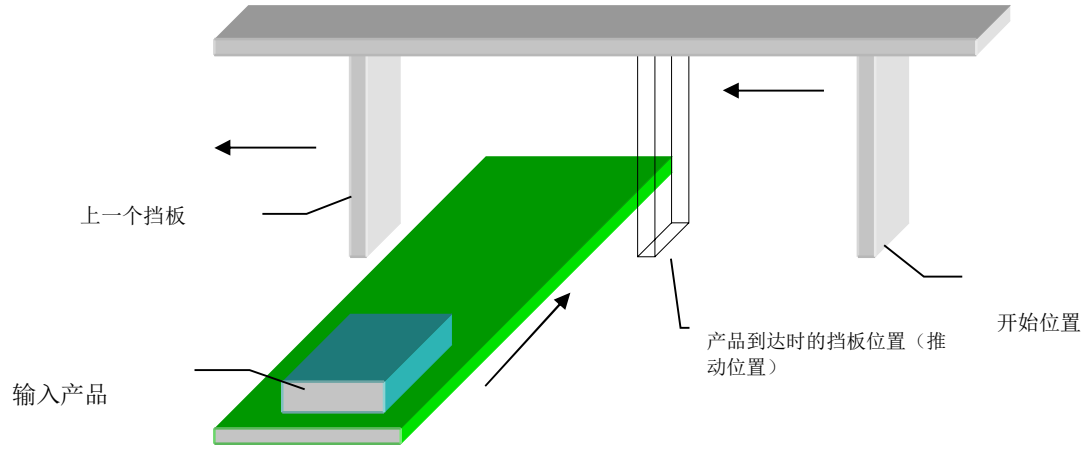
**FLY = ((Initial Ratio + Final Ratio)/2) * MASTERDISTANCE**

（有关 FLY 关键字的介绍，请参阅应用说明 AN00116）。

当我们启动挡板时，初始齿轮比为 0（停止状态）。最终比率是跟随比（在本例中为 3）。因此，上面的公式表明，启动期间行走的从距离（FLY）与启动期间行走的产品距离（MASTERDISTANCE）之间的关系将为：

***FLY* = *(3/2)* * MASTERDISTANCE**

让我们来看看到目前为止我们已知道的：




上图显示了挡板的起始位置（此时我们的编码器值将为零）。该图还显示了当输入产品到达后挡时我们希望挡板进入的位置（分选门在关闭时充当后挡）- 我们将该位置称为推动位置。

通过测量起始位置和推动位置之间的距离（在编码器计数中），我们可以确定该轴所需的 FLY 段。因此，将其代入我们之前的公式，可以计算出相应的 MASTERDISTANCE：

***FLY = (3/2) * MASTERDISTANCE***

***MASTERDISTANCE = FLY * 2/3***

如果挡板到达推动位置的距离是 3000 个计数，那么 MASTERDISTANCE 将是 2000 个计数。

必须通过检测输入产品（例如通过光电传感器）触发启动。因此，我们可以得出结论，该传感器必须位于距离后挡至少 2000 个编码器计数。

在这个例子中，给料输送机每 mm 线性行程产生 20 个计数 - 因此传感器需要距离后挡至少 100mm。

在此阶段考虑最小产品长度非常重要。如果产品长度使一个以上的产品等于传感器和后挡之间的间隙（记住，要包括产品之间的标称间隙），那么我们有两个选择：

a) 考虑设置新的起始距离，以减少最小的传感器距离

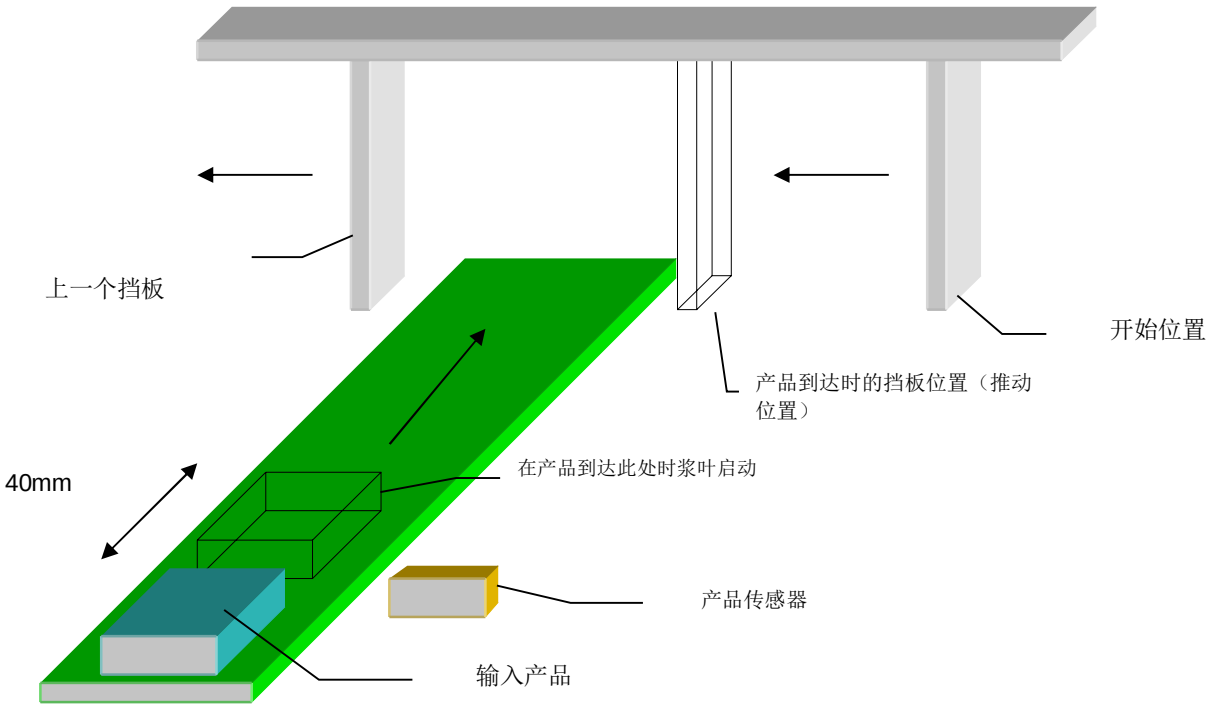
b) 继续使用现有的设置 - 然而，这将要求软件“缓冲”产品触发的信号（例如，对准修正），以便将它们应用于正确的挡板

在本例中，我们的产品间距是 4000 或 200 毫米。因此，我们可以得出结论，目前起始位置是可以接受的。

但是，我们假设由于机械限制，无法把传感器安装在距离后挡 100mm 的位置，相反，传感器必须安装在距离后挡 140mm（或个 2800 计数）的位置。这仍然满足只有一个产品在传感器和后挡之间的要求，但现在我们不能在检测到产品时立即触发离合跟随或飞剪段。我们无法修改我们的起始飞剪段，因为这会改变挡板的起始位置，然后产品可能无法到达分选门。

相反，我们需要延迟开始运动，直到产品到达距离后挡 100mm 的位置。

这可以通过第二个飞剪段来实现，其中 MASTERDISTANCE 是产品延迟距离，并且 FLY 段被指定为零（因此挡板不移动）。




如果我们为所有系统参数分配变量名称，那么当产品激活快速输入 0 时，我们可以轻松地计算这些参数并编写逻辑以启动挡板。

```
Dim fFollowRatio As Float                    ‘ 挡板的最终跟随比
Dim nPaddleStartDistance As Integer   ‘ 挡板在启动和推动之间移动的距离
Dim nDelayDistance As Integer                ‘ 挡板开始移动之前等待的距离
Dim nRampDistance As Integer                 ‘ 挡板沿斜坡加速的距离
Dim nPaddlesStopped As Integer = _True       ‘ 用于决定是否启动浆叶的标志
Const _nEyeToBack As Integer = 2800 ‘ 从产品传感器到后挡的距离

‘ 改变比率时使用飞剪段
FOLLOWMODE(0) = _fmContinuous_Clutch

‘ 计算挡板启动顺序的产品距离
nRampDistance = Int(( nPaddleStartDistance * 2 ) / fFollowRatio )
nDelayDistance = _nEyeToBack  –  nRampDistance

Loop
‘ 主组件任务
End Loop

Task StartPaddles
nPaddlesStopped = _False             ‘ 视挡板为已经启动
MASTERDISTANCE(0) = nDelayDistance       ‘ 等待产品到达起点
FLY(0) = 0: GO(0)
MASTERDISTANCE(0) = nRampDistance    ‘ 浆现在沿斜坡加速
FOLLOW(0) = fFollowRatio                 ‘ 或 FLY(0) = (fFollowRatio/2) * MASTERDISTANCE
End Task
```

```
Event Latch0
If nPaddlesStopped Then Run StartPaddles
End Event
```

## 停止挡板

有许多条件可能要求挡板停在产品可以被剔除的位置 - 即挡板必须停在它们首次开始的位置（编码器值为 0）。

通过在其自己的任务中编写停止逻辑，我们可以在整个应用程序中通过许多不同的事件或条件“触发”停止，而其他任务继续执行。例如，如果检测到超长产品（如果快速输入 0 和输入 1 同时为 ON），我们可以让挡板停止。

因为我们使用的是扩展离合跟随模式，我们可以使用 FLY 命令使挡板轴在规定的位置暂停。我们从飞剪的通用公式中看到：

**FLY = ((Initial Ratio + Final Ratio)/2) * MASTERDISTANCE**

当挡板连续运转时（跟随），我们已经看到它们跟随给料输送机轴线的跟随比率为 3 - 因此 3 是我们的初始比率。要使挡板停止，最终比率必须为 0。

因此，我们可以得出结论，我们的 FLY 段和相关 MASTERDISTANCE 的公式将采用以下形式：

**FLY = (3/2) * MASTERDISTANCE**

如果挡板轴在正确的位置停止，则轴编码器值应为 0。因此，当我们开始停止挡板时捕获当前轴的位置，我们可以得出在停止期间要行进的距离 - 即 FLY 段的值。

**例如：**
如果在我们启动停止时读取到浆叶编码器值为 5000（记住轴的 ENCODERWRAP 值为 12000），则行走到编码器值 0 的距离为（12000 - 5000）= 7000 个计数。

一旦我们计算了所需的 FLY 段，我们的通用公式就可以用来计算相关的 MASTERDISTANCE。

因此，我们的 Mint 挡板停止任务应如下所示：

```
Task StopPaddles
Dim nStopFly As Integer                          ‘ 声明存储飞剪段的变量
nStopFly = ENCODERWRAP(0) – ENCODER(0)           ‘ 计算挡板的行走距离
MASTERDISTANCE(0) = nStopFly * 2/3               ‘ 计算相关的 masterdistance
FLY(0) = nStopFly                          ‘ 发出挡板停止飞剪的命令
GO(0)
End Task
```

## 对准浆叶

一旦挡板以我们的标称跟随比率跟随给料输送机，则它们应该连续收集产品，并以 90 度推动输入产品。然而，该过程的成功取决于输入产品是否有规则、精确的间距。输入产品间距的任何变化，都要求挡板提前或滞后，以确保推动仍能顺利进行。

这种位置的提前或滞后（同时仍保持整体跟随比）可被视为*产品对准*。

Mint 允许通过 OFFSET 关键字轻松实现产品对准。OFFSET 关键字要求用户指定校正距离，Mint 会自动计算必要的速度调整值以执行此校正。

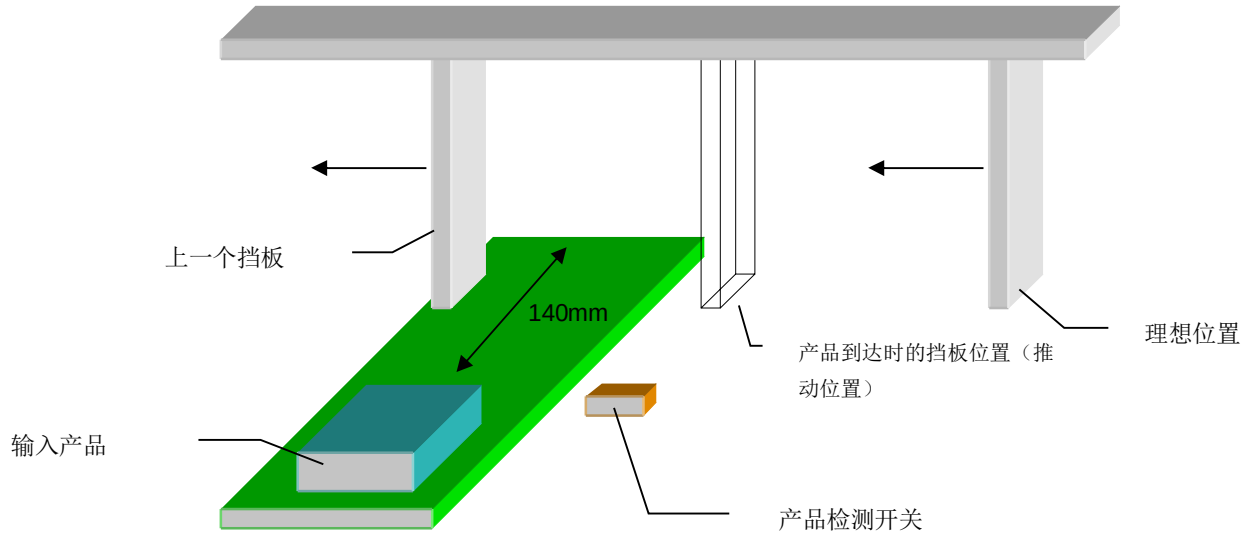
我们已经看到，EVENT LATCH0 如何被用于检测输入产品并在必要时启动挡板。同样的输入可用于捕获运行的挡板的位置，并确定输入产品的位置是否需要挡板提前或滞后。

锁定的配置将根据要使用的控制器而变化。在我们的例子中，我们假设使用智能伺服驱动器（例如带有 Mint 存储器模块的 MicroFlex e190）（因为我们只控制一个电机/轴）。在这种情况下，快速输入 1 被配置为通过以下 Mint 代码段捕获挡板（编码器 0）的编码器数值：

```
LATCHENABLE(0) = _off
LATCHSOURCE(0) = _lsENCODER
LATCHSOURCECHANNEL(0) = 0
LATCHTRIGGERMODE(0) = _ltmINPUT
LATCHTRIGGERCHANNEL(0) = 1
LATCHTRIGGEREDGE(0) = _ltePOSITIVE_EDGE
LATCHMODE(0) = _lmAUTO_ENABLE
LATCHENABLE(0) = _on
```

现在剩下的就是计算每个输入产品的对准要求。

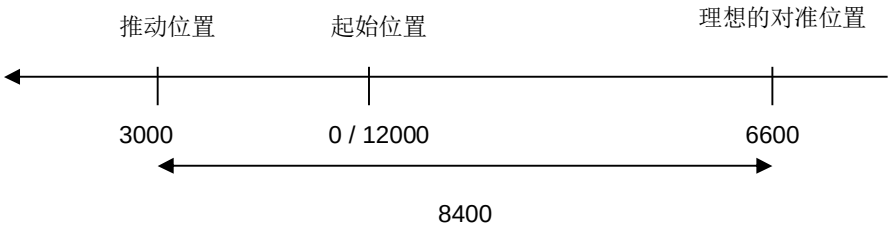
让我们来看看检测到产品时通常会发生什么：




我们已经通过启动挡板的工作了解到，当产品检测传感器检测到产品时，输入产品将距离后挡 140mm（或给料输送轴的 2800 个计数）。我们还了解，当产品到达止挡时，挡板距离其初始启动位置（其中初始启动位置的编码器值为 0）已经移动了 3000 个计数。

一旦挡板完成初始加速，它们就会以固定的比率（3：1）跟随给料输送机的轴。因此，我们可以得出结论，如果挡板处于已对准状态，它们应该在产品移动到止挡时行进了 3 * 2800（8400）个计数。

因此，当检测到产品时，挡板的理想位置是从编码器值 3000 开始回退 8400 个计数。下图说明了我们如何根据以下信息计算理想的对准位置：




计算出“理想”位置后，我们可以将其与每次检测到产品时快速中断自动捕获的位置进行比较。然后可以将捕获位置和我们的“理想”位置之间的差值作为 OFFSET 应用，以确保挡板保持对准状态。

以下 Mint 代码段说明了如何将启动、停止和对准工作组合到单个 Mint 应用程序中：

```
Dim fFollowRatio As Float                    ‘挡板的最终跟随比
Dim nPaddleStartDistance As Integer          ‘浆叶在启动和推动之间移动的距离
Dim nDelayDistance As Integer                ‘ 挡板开始移动之前等待的距离
Dim nRampDistance As Integer                 ‘ 挡板沿斜坡加速的距离
Dim nPaddlesStopped As Integer = _True       ‘ 用于决定是否启动浆叶的标志
Dim nIdealPos As Integer                     ‘ 检测到产品时理想的浆叶位置
Const _nEyeToBack As Integer = 2800 ‘ 从产品传感器到后挡的距离

‘改变比率时使用飞剪段
FOLLOWMODE(0) = _fmContinuous_Clutch
LATCHENABLE(0) = _off
LATCHSOURCE(0) = _lsENCODER
LATCHSOURCECHANNEL(0) = 0
LATCHTRIGGERMODE(0) = _ltmINPUT
LATCHTRIGGERCHANNEL(0) = 1
LATCHTRIGGEREDGE(0) = _ltePOSITIVE_EDGE
LATCHMODE(0) = _lmAUTO_ENABLE
LATCHENABLE(0) = _on

‘计算挡板启动顺序的产品距离
nRampDistance = Int(( nPaddleStartDistance * 2 ) / fFollowRatio )
nDelayDistance = _nEyeToBack  –  nRampDistance

‘计算挡板的理想对准位置
nIdealPos = ENCODERWRAP(0) - ((_nEyeToBack * fFollowRatio)  –  nPaddleStartDistance)

Loop
‘主组件任务
End Loop

Task StartPaddles
nPaddlesStopped = _False                     ‘视挡板为已经启动
MASTERDISTANCE(0) = nDelayDistance                  ‘等待产品到达起点
FLY(0) = 0: GO(0)
MASTERDISTANCE(0) = nRampDistance                   ‘挡板现在沿斜坡加速
FOLLOW(0) = fFollowRatio                            ‘或 FLY(0) = (fFollowRatio/2) * MASTERDISTANCE
End Task
```

```
Task StopPaddles
Dim nStopFly As Integer                          ' 声明存储飞剪段的变量
nStopFly = ENCODERWRAP(0) – ENCODER(0)           ' 计算挡板的行走距离
MASTERDISTANCE(0) = nStopFly * 2/3               ' 计算相关的 masterdistance
FLY(0) = nStopFly                          ' 发出挡板停止飞剪的命令
GO(0)
End Task

Event Latch0
' 每次检测到产品时都会生成此事件（前沿）

' 如果挡板已停止，启动它们
If nPaddlesStopped Then
Run StartPaddles
Exit Event          ' 如果刚开始，则无需处理任何其他代码
End If

' 如果检测到超长产品，停止浆叶
If INSTATEx(1) Then
Run StopPaddles
Exit Event          ' 如果正在停止浆叶，则无需处理任何其他代码
End If

' 如果达到这一点，则必须跟随挡板，并且必要时可以应用对准更正
OFFSET(0) = WRAP(nIdealPos – LATCHVALUE(0), -ENCODERWRAP(0) / 2, ENCODERWRAP(0) / 2)
GO(0)

End Event
```

## 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**