运动控制产品

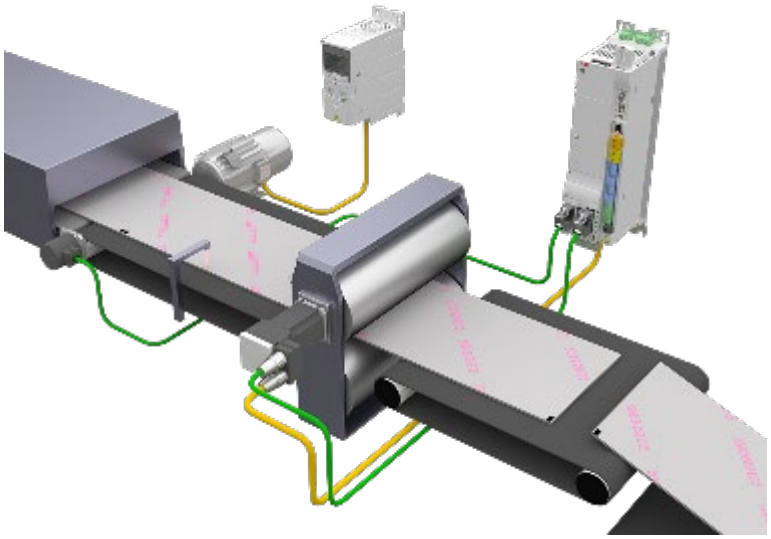
应用说明文档

旋切

AN00226

版本 D（中文）

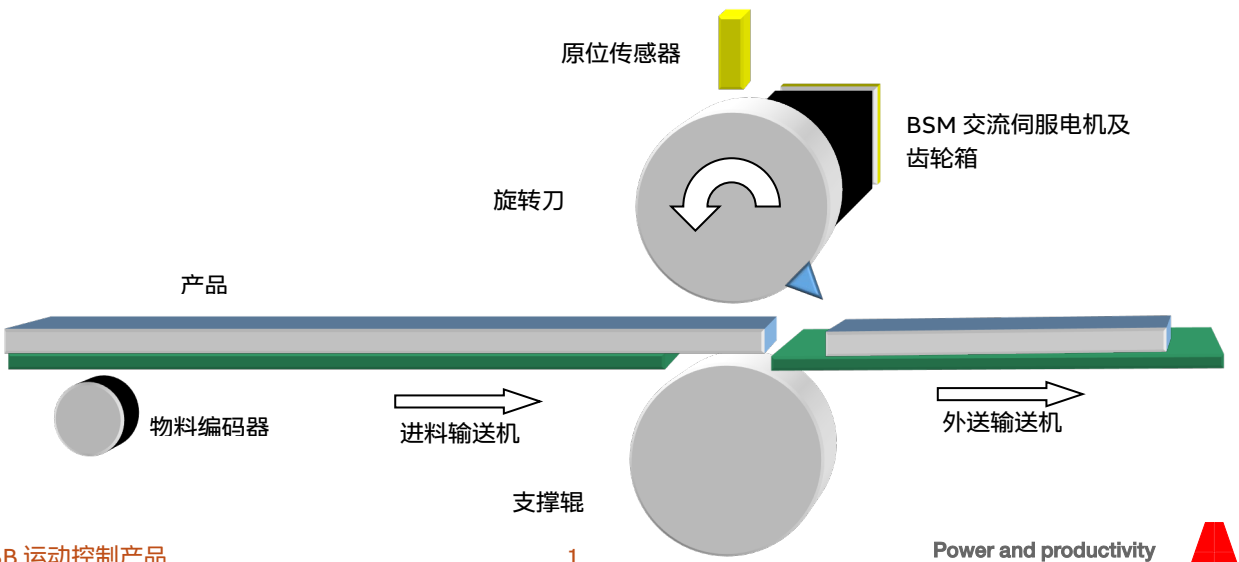
“飞剪”通常会让人想到线性切割机构与移动产品同步，将移动产品切割成一定长度的应用。但 Mint 也可以使用同样的技术，达到用旋转式切割机构取得的出色效果



简介

各类应用可能都需要两轴之间的同步。AN00116 显示出了这方面的一个示例，即线性飞剪机应用，其要求轴速度在规定的产品和轴行程上精确同步。AN00116 展示了如何使用 Mint FLY 和 MASTERDISTANCE 关键字轻松解决这类应用。
随后，AN00122 对这些概念进行了扩展，使用间歇压花轮作为示例应用，展示如何使用相同的关键字来使旋转轴与移动产品同步。

本应用说明 AN00226 提供了使用 Mint FLY 和 MASTERDISTANCE 关键字的另一个示例，使用的是非常常见的应用类型 - 旋转。本应用说明提供的编码将在 MicroFlex e190 和 MotiFlex e180 智能伺服驱动器上执行，同时还提供了一个 CP600 系列 HMI 项目示例供参考。



上图显示了旋切应用的典型特征简化版本。输送机是以恒定的线速度进行输送，并将其切割成一定长度的产品。输送机可由感应电机和变频驱动器（如 ACS355）驱动。编码器检测输送机运行时的位置和速度。这通常被称为“主”或“物料”编码器。第二个输送机通常比第一个输送机运行得稍微快一些，以带走切割下来的物料（并因此在成品之间形成间隙）。旋转式切割机本身由交流伺服电机驱动（通过合适的齿轮箱，确保负载和电机之间的良好惯性匹配），该电机反过来连接到 MicroFlex e190 或 MotiFlex e180 智能伺服驱动器。

驱动器检测物料编码器（连接到驱动器上的编码器通道 2），并用它来确定切割机电机的速度/位置轮廓。物料编码器和旋转式切割机按比例按线性行程单位计算（因此，对于物料，为每单位输送机行程的计数数，而对于旋转式切割机，为每单位切割机周长行程的计数数）。

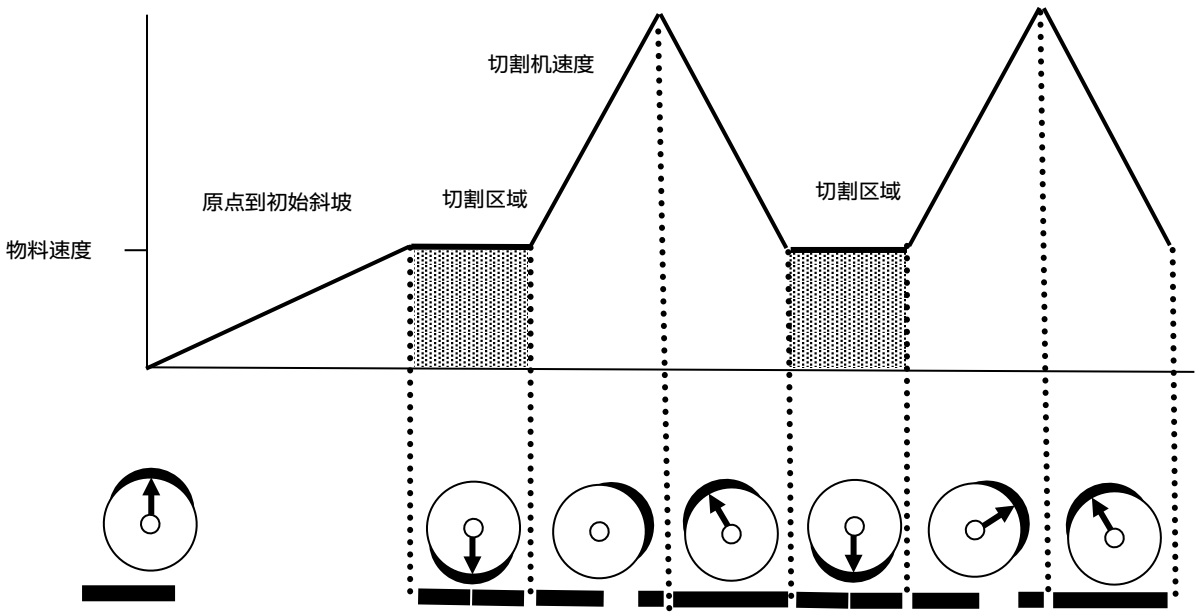
运动轮廓

旋转式切割机的表面速度通常在切割机本身的指定角度上（例如 30 度，在切割机旋转的下止点处的切割点两侧各 15 度）与物料同步。
当切割机按比例按周长行程单位计算时，该角度被转换为线性行程（例如，如果切割机周长为 300 毫米，则 30 度等于 300*30/360=25 毫米的周长行程）。

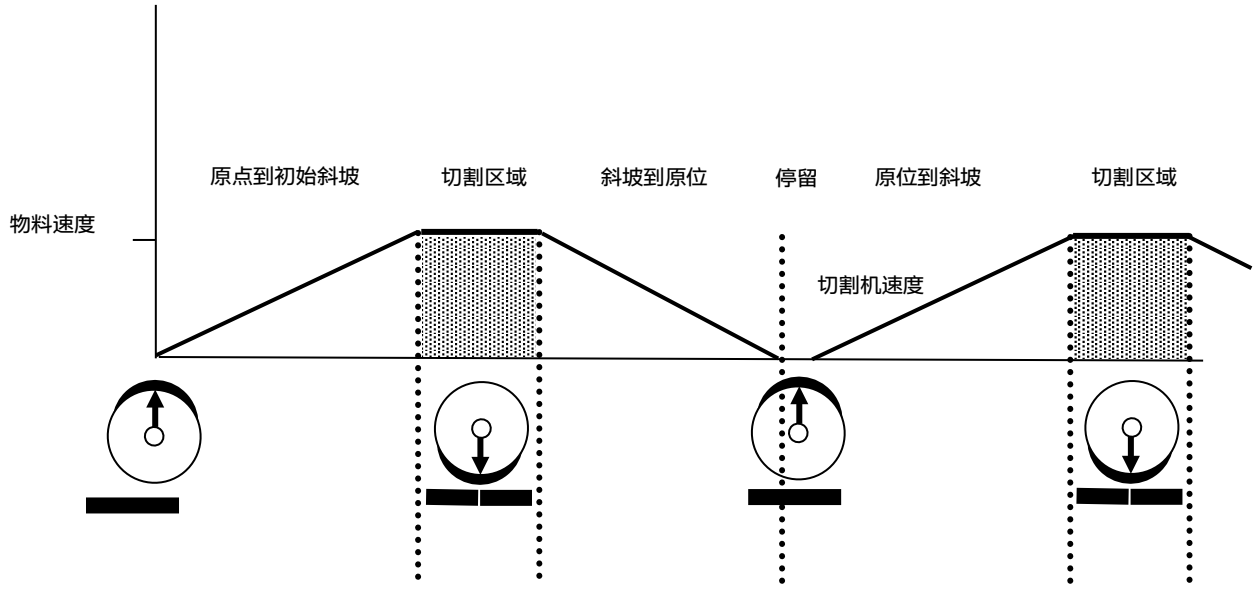
通常两个轴之间的同步是 1:1，所以如果切割机在切割区域内绕其周长移动 25 毫米，物料本身也将移动 25 毫米。因此，为了达到总的切割长度，切割机必须在剩余的物料行程上完成一次旋转的剩余部分（例如，如果所需的切割长度为 150 毫米，则切割机必须在接下来的 125 毫米物料行程上旋转其周长的剩余 275 毫米）。

下图说明了两轴速比之间的关系。在我们的示例中，我们将假设切割机的起始位置与原位传感器对齐（即在“12 点钟”位置，如我们的原始简化应用图所示）。

阴影区域表示当旋转轴的表面速度与物料的线速度完全匹配时：



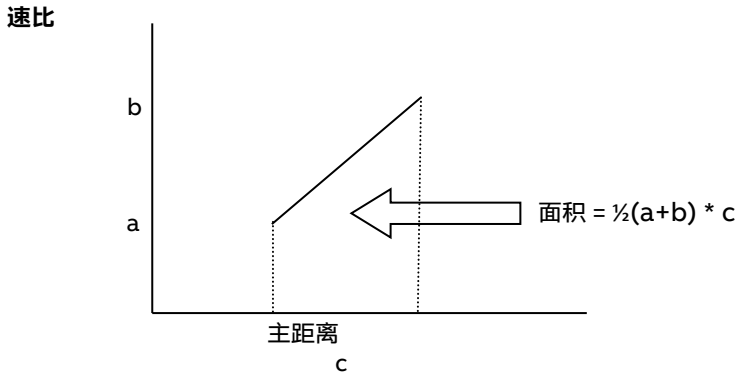
上面的运动轮廓显示了对于短于切割机本身周长的切割长度，切割机轴速度可能出现的情况（即，切割机必须在连续切割之间加速）。随着所需切割长度的增加，切割之间的速比减小，直到最终切割长度太长，切割机不得不停止/停留一段时间后再重启……



所需速度轮廓的每个元素都可以被分割成所谓的“段”（上图中已作说明）。Mint 可以使用 MASTERDISTANCE 关键字规定物料在每段的移动距离。

每段中曲线下方的面积对应的是切割机轴移动的距离。在 Mint 程序中，可以使用 FLY 关键字规定该距离/面积。

如果我们检查任何段，我们会看到，该段的面积（即 FLY 的值）取决于：



面积=主距离*（初速比+终速比）/2

把这个转换成 Mint 编码，我们得到：

FLY(0) = MASTERDISTANCE(0) *（初速比+终速比） /2

如果只有 MASTERDISTANCE 是已知的，这个通用公式可以用来计算 FLY，反之亦然。

应用程序简单示例

考虑到我们的旋转式切割机具有以下特点：

传输带信息：
通过直径为 100 毫米的测量轮（100 x pi 毫米周长）上的层驱动物料编码器
物料编码器具有 4096 行（16384 正交）分辨率
编码器连接到 MicroFlex e190 或 MotiFlex e180 伺服驱动器上的编码器通道 2

切割轴信息：
切割机直径 = 200 毫米（200 x pi 毫米周长）
电机编码器分辨率 = 每转 131072（正交）计数
变速箱比 = 3:1
总同步角 = 40 度
切割速比 = 1:1

请注意，在本示例中，我们对同步角、切割机直径和切割速比的值都进行了“固定设置”，但实际上，这些值通常都可以通过 HMI 进行调整，因此通常在计算中使用变量，以使编码更通用，并适用于具有不同尺寸/特性的系统。

我们要做的第一件事是将我们的物料编码器和切割机轴按比例计算成线性行程（单位：毫米）：

ENCODERSCALE(2) = 16384 / (100 * pi)‘每转编码器计数/测量轮周长
SCALEFACTOR(0) = (131072 * 3) / (200 * pi) ‘每转齿轮箱输出计数/切割机周长

现在，我们可以使用我们前面创建的飞剪机通用公式，开始计算我们的切割机轮廓各段的 MASTERDISTANCE 和 FLY 段值……

FLY = MASTERDISTANCE *（初速比+终速比）/2
⇨ MASTERDISTANCE = (2 * FLY) /（初速比+终速比）

起始段：
在本示例中，我们将假定切割机从原位（上止点）开始，并且必须上升到同步切割区域的起始位置（下止点前 20 度；即从原位开始移动 160 度），以便速度与物料 1:1 匹配。

初始比率 = 0；最终比率 = 1；FLY = (200 * pi) * 160 / 360
⇨ MASTERDISTANCE = ((2 * 200 * pi) * 160 / 360) / (0 + 1) = (400 * pi) * 160 / 360

因此，在我们的 Mint 程序中，我们将包含这两行编码来启动切割机：

MASTERDISTANCE(0) = (400 * _pi) * 160 / 360
FLY(0) = (200 * _pi) * 160 / 360

如果我们计算这个 MASTERDISTANCE 值，我们会发现结果约为 558.5 毫米。因此可以看出，第一个切割长度将至少是这个距离加上物料到切割点的行程（即，我们接下来将计算的同步 MASTERDISTANCE 的一半）。因此，在实际应用中，我们可能会考虑其他方法来启动切割机，具体视第一次切割是否需要产生正确长度的产品而定。我们将在本应用说明后面详细讨论这一点。

同步切割段：
初始比率 = 1；最终比率 = 1；FLY = (200 * pi) * 40 / 360
⇨ MASTERDISTANCE = ((2 * 200 * pi) * 40 / 360) / (1 + 1) = (200 * pi) * 40 / 360

因此，在我们的 Mint 程序中，我们将包含这两行编码来执行同步切割：

MASTERDISTANCE(0) = (200 * _pi) * 40 / 360
FLY(0) = (200 * _pi) * 40 / 360

（由于我们的速度匹配比是 1:1，并且两个轴都被按比例计算成线性行程（单位：毫米），所以 MASTERDISTANCE 和 FLY 值相同并不奇怪！）

切割机剩余段：
要计算这些剩余的段，我们必须首先确定我们是只需要加载两个段（每个段将剩余的切割机行程和物料行程分成两半），仍需加载三个段（包括一个停留段，以确保我们的整体切割长度正确）。

为了做这个决定，我们需要计算在以下时间内实现的总主距离（物料行程）：

- 同步切割区域
- 使切割机在切割比停止
- 重启切割机至所需的切割比

如果这些主距离之和小于所需的切割长度，则我们需要包括停留段以延长切割周期。如果这些主距离之和大于（或等于）所需的切割长度，则只需加载同步切割段，然后将剩余距离对半。

我们已经计算了同步切割区域的主距离：(200 * _pi) * 40 / 360
当我们计算起始段值时，我们已经计算了切割机从零速度到切割比所需的主距离：(400 * _pi) * 160 / 360
如果我们假设切割比固定在 1:1 并且永远不变，则我们还可以看到停止切割机的主距离等于启动切割机所需的主距离：(400 * _pi) * 160 / 360

我们可以为我们的应用程序计算这个“总主距离”，并将其存储在一个名为 fTotalMSD 的 Mint 变量中……

fTotalMSD = ((200 * _pi) * 40 / 360) + (2 * (400 * _pi) * 160 / 360)

如果我们假设切割长度通过 HMI 输入到系统中，并最终存储在名为 fCutLength 的 Mint 变量中，则我们就拥有了编写所需应用程序逻辑需要的所有信息…

如果 fTotalMSD < fCutLength，则

```
‘需要三段……停止、停留和重启……
MASTERDISTANCE(0) = (400 * _pi) * 160 / 360
FLY(0) = (200 * _pi) * 160 / 360

MASTERDISTANCE(0) = fCutLength – fTotalMSD
FLY(0) = 0

MASTERDISTANCE(0) = (400 * _pi) * 160 / 360
FLY(0) = (200 * _pi) * 160 / 360
```

其他

```
‘需要两段……每段都是剩余距离的一半
 MASTERDISTANCE(0) = (fCutLength – ((200 * _pi) * 40 / 360)) / 2
 FLY(0) = ((200 * _pi) * (360 – 40) / 360) / 2


 MASTERDISTANCE(0) = (fCutLength – ((200 * _pi) * 40 / 360)) / 2
 FLY(0) = ((200 * _pi) * (360 – 40) / 360) / 2
```

End If

加载了一个完整的循环之后，我们的应用程序可以持续重复加载同步切割段和所需的“剩余段”，以不断将物料切割到所需的长度。有些应用程序要求在物料停止之前将切割机停在原位，有些应用程序则要求切割机一旦开始就永久“循环”，因此停止位置完

全随机，具体取决于物料停止的时间。在我们的简单示例中，我们将假设切割机保持“循环”，并且随着物料的停止和启动而停止和启动。

必须设置 Mint MOVEBUFFERSIZE 以确保有足够的空间加载所需的段，并且必须使用适当的运动触发命令（通常为 Go）来操作运动，有关移动缓冲区大小和触发运动的详细信息，请参考 AN00116 或 Mint 帮助文件。

轴漂移补偿

从前面的示例编码可以看出，加载 MASTERDISTANCE 和 FLY 段的大部分逻辑都涉及浮点数学计算。因此，随着时间的推移，不可避免地会有一些误差，随着时间的增加，切割机的轴切割位置会产生漂移。（最终导致切割机切点在 40 度区域意外的部分，速度将与物料同步）。

因此，我们在编程中需要一个机制来检测切割机在其循环内的位置，并将其与控制器切割机应该在的位置进行比较。利用 MOVEPULSEOUTX 指令（允许控制器以相对于运动轮廓仪的精确位置为数字输出脉冲）和 Mint 锁存器事件（可用于捕获切割机轴编码器值）的组合，最容易实现这一目标。通过数字量输出触发来配置锁存事件，可以比较切割机在一个循环内的实际位置与其理论位置。然后，使用 Mint OFFSET 命令对轴进行“移相”，以补偿测量到的漂移。

配置步骤如下：

(a) 为切割机轴配置 ENCODERWRAP。在示例中，切割机电机装有 3:1 齿轮箱，电机编码器分辨率为每转速 131072 个计数，致使 ENCODERWRAP(0) 为 393216。如果应用属性使得每个切割机循环的编码器计数中有小数，则需要在每次循环时使用原位传感器（通过附加锁存器事件）将切割机编码器值重置为给定原位传感器位置的期望值

(b) 同步切割段的中间位置包含一个 MOVEPULSEOUTX 命令。通过将 MOVEPULSEOUTX 放置在此处，我们可以确保在切割之前，不会执行后续的 OFFSET 命令以校正测量的漂移。要将 MOVEPULSEOUTX 插入同步区域的中间位置，需要将前一段分成两半（此指令需要移动缓冲区中的额外空间）

```
MASTERDISTANCE(0) = ((200 * _pi) * 40 / 360) / 2
FLY(0) = ((200 * _pi) * 40 / 360) / 2
MOVEPULSEOUTX(0, 1) = 10  ‘脉冲输出 1 持续 10ms
MASTERDISTANCE(0) = ((200 * _pi) * 40 / 360) / 2
FLY(0) = ((200 * _pi) * 40 / 360) / 2
```

(c) 配置一个锁存器事件（在 Mint 启动模块中），当输出 1 运行时，该事件捕获切割机锁存编码器值……

```
LATCHENABLE(1) = 0
LATCHSOURCE(1) = _lsENCODER
LATCHSOURCECHANNEL(1) = 0
LATCHTRIGGERMODE(1) = _ltmOUTPUT
LATCHTRIGGERCHANNEL(1) = 1
LATCHTRIGGEREDGE(1) = _ltePOSITIVE_EDGE
LATCHMODE(1) = _lmAUTO_ENABLE + _lmINHIBIT_VALUE
LATCHINHIBITVALUE(1) = 0.8 * 200 * _pi
```

在本示例中，我们使用锁存通道 1。注意，通过使用抑制值模式，我们可以将切割机周长的 80% 用作滤波器距离，以防快速锁存器意外触发。

(d) 一旦轴成功归位使用，启用“防漂移”锁存器……

```
LATCHENABLE(1) = 1
```

(e) 配置切割轴的 OFFSETMODE 以在一定距离上应用漂移校正，这意味着校正不会快速应用，但同样允许它们在下一个同步切割段发生之前完成……

```
OFFSETMODE(0) = _ofTWO_SEGMENT
OFFSETDISTANCE(0) = fCutLength * 300 / 360  ‘下一个 360 度中的 300 度
```

(f) 包括用于计算和应用漂移校正的锁存器事件……

```
锁存器事件 1
 ‘每次发生此锁存器事件时（即移动 180 度时），切刀应位于下止点（切割点）……
 fLatch = LATCHVALUE(1)
 fDrift = WrapOffset(fLatch, (180 / 360) * (_fKnifeDia* _pi), 0, (_fKnifeDia* _pi))
 如果 (AXISMODE(0) And _mdOFFSET) <> _mdOFFSET，则
  OFFSET(0) = fDrift
  GO(0)
 End If
结束事件
```

修改起始段

同样，对于本主题，为使操作原理更容易理解，我们将使用应用程序示例中的硬编码值，但在实际操作中，最好使用变量并通用性地计算相关数据，以便程序做出有关程序流的决策（有关如何实现这一点的示例，请参阅本应用说明附带的 Mint 应用程序编码）。

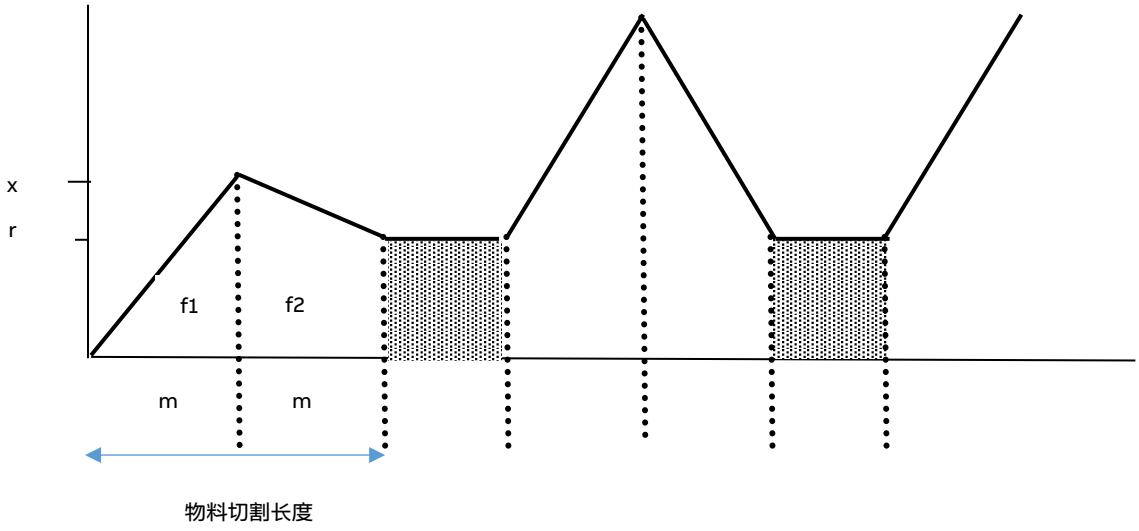
如前所述，在第一次切割发生之前，通过将切割机轴从原位倾斜到 1:1 的比例，物料将移动 558.5 毫米加上同步段的一半（34.906 毫米，总计约 593.4 毫米）。对于较长的产品，此初始“错误长度”可能不会造成不便，但对于较短的产品（例如 200 毫米的产品），这可能是一个很大的浪费，并且可能会证明输出输送机和任何后续过程难以处理。因此，对于长度小于 593.4 毫米的产品，修改起始段会是有利的，以便立即获得正确的切割长度。

类似地，对于较长的产品，在加载默认起始段之前，最好包括初始延迟（即，FLY 为零的 MASTERDISTANCE）。

在这两种情况下，典型的做法是首先将切割机放回原位，以便物料从“新的”产品前缘开始进料。

对于较长的产品，很容易计算停留段的 MASTERDISTANCE，即所需的切割长度与我们将为默认起始段和一半同步段（在我们的示例中为 fProductLength – 593.4）行进的总主距离之间的差值。

对于较短的产品（在我们的示例中小于 593.4 毫米），通常将起始段分成两部分，起始段的主距离一分为二，使两段的每个主距离为所需切割长度的一半。随后用于启动切割机的两个 FLY 段的总和必须等于切割机绕其周长，从原位行进到同步段起始位置的线性距离。由于同步角为 40 度宽，下止点两侧为 +/-20 度，因此 FLY 段的总和必须为 (200 * _pi) * (180-20) / 360 = 279.253 毫米（该值中的任何微小误差将通过前述漂移补偿编码进行校正）。下图说明了这一原理：



m = 切割长度/2[在本示例中，假设我们需要 200 毫米的切割长度]

f1 + f2 = (200 * _pi) * (180 – 20) / 360　　[279.253 毫米，但为了工作清晰起见，我们将其称为 F]

x 是我们未知的速比，它作为我们第一个 FLY 段的终速比和我们第二个 FLY 段的起始速比

r 是第二段的终速比（通常为 1:1），其随后用于启动同步切割段

利用飞剪机通用公式，我们可以导出下列方程……

0.5 * (0 + x) * m = f1

0.5 * (x + r) * m = f2

我们可以用 f1 + f2 = F 来代替这两个方程式得出……

(0.5 * x * m) + (0.5 * (x + r) * m) = F

扩大这些得出……

(0.5 * x * m) + (0.5 * x * m) + (0.5 * r * m) = F

⇨ (x * m) + (0.5 * r * m) = F

重新排列以得出 x 的结果：

x = (F – (0.5 * r * m)) / m

然后，我们可以通过将该 x 值代回到 f1 和 f2 的两个原始等式中来加载我们的两个段：

MASTERDISTANCE(0) = m

FLY(0) = 0.5 * (F – (0.5 * r * m)) / m

MASTERDISTANCE(0) = m

FLY(0) = 0.5 * (((F – (0.5 * r * m)) / m) + r) * m

[其中 F=279.253，m=100]

示例文件

还有许多其他特性可以添加到这样的应用程序中，而不是将其全部文档化，本应用说明附带了一个示例 Mint 程序（以及使用通过 Modbus TCP 连接的 CP600 的 HMI 项目），它提供了以下功能：

- 经常在“运行时间”调整的应用程序属性的 HMI 条目（例如，同步角度、原位速度、切割长度、切割比）
- 已完成切割计数器的 HMI 显示
- 能够在切割机运行时改变切割参数（如切割比和切割长度）
- 演示如何使用程序变量创建通用解决方案，而不是硬编码的特定解决方案
- 漂移补偿逻辑
- 将 Mint 应用程序错误显示为 HMI 上的字符串数据（例如“超出误差设定”）
- 用于处理 Mint 停止输入激活和恢复的示例程序结构
- 用于处理 Mint 错误和恢复的示例程序结构
- 用于存储和检索需要是非易失性的 HMI 数据的示例程序结构
- 首次使用 Mint 程序时设置默认值的示例程序结构
- 用于控制驱动器的使能状态并将其与其它程序逻辑联锁的示例程序结构
- 例如回零程序,说明如何确保回零逻辑始终确保在回零期间执行初始剪切

– 确保第一次切割是正确的长度的逻辑（假定切割机最初已回零，直到回零完成后才开始送料）

如果您需要有关旋切应用所需的任何附加功能的帮助，请与当地 ABB 办事处联系以获得技术支持。

联系我们

欲了解详情，请联系
当地 ABB 代表或：

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