运动控制产品

应用说明
多个对准标记

AN00106

Rev G (CN)

在检测到印刷标记或其他信号时启动校正动作或系统事件的系统中，
当在每个重复长度或机器周期内存在多个信号时，通常很难确定要使用的正确信号。

引言

例如，彩色印刷机通常沿着材料的外边缘有多个标记。这可用于质量检查、色彩一致性检查、批量编号等。然后，用于其它工艺的对准和随后的定线（例如附加彩色印刷、压花，按长度切割或穿孔）。

处理这个问题的方法原则上很简单，但实施起来往往很麻烦。Mint 凭借其强大的高级语言和快速中断处理功能，为解决此问题提供了灵活的平台。

印刷标记窗口

我们感兴趣的对准标记或信号通常位于距材料切割位置的已知距离处，通常称为“偏移距离”，因此，当我们查看材料时，我们自然会在我们期望的区域寻找标记。也是基于此原理，我们滤除掉不需要的标记信号。当然，我们必须考虑到误差，因此可以在一定的容差范围内接受标记。该容差带通常被称为“印刷窗口”或“印刷标记窗口”。

这种实现是基于已了解的偏移距离（我们将其转换为编码器计数）和“印刷窗口”或容差带。

例如，假设我们有一个应用，其中材料通过送料辊向切割器移动。已经在材料中印刷了几个印刷标记，用于下游的后续处理。因此，我们必须过滤它们，只捕获正确的印刷标记。产品切割长度为 1 米，在距离刀具 0.5 米处安装传感器。传感器到刀具的距离即“偏移距离”，该偏移距离为已知（0.5 米）。因为我们已知正确标记相对于刀具的位置，我们可以将窗口或容差带定义为偏移距离的 ± 2.5 cm。这样，我们就可以过滤不在此范围内的任何其他标记。

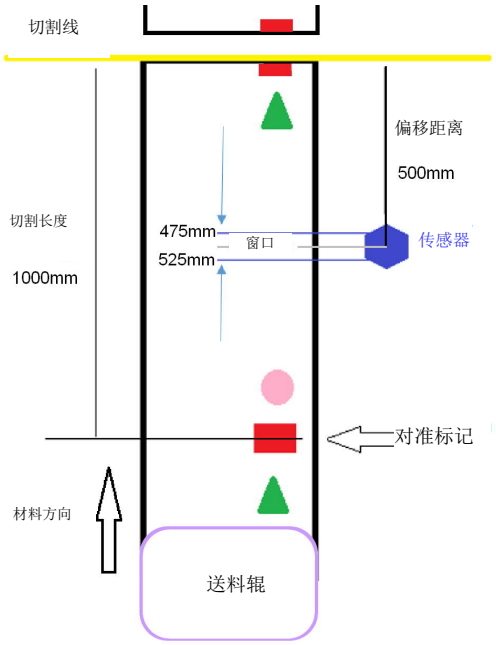
可定义如下常量：

```
Const _nCutLength As Integer = 1000          ' 使用变量设置切割长度，单位为 mm
Const _nOffsetDistance As Integer = 500      ' 定义传感器到切割线的距离，单位为 mm
Const _nStartPrintWindow = 475               ' 定义“印刷”窗口的起点，偏移距离-25 mm 的绝对值
Const _nEndPrintWindow = 525                 ' 定义“印刷”窗口的起点，偏移距离+25 mm 的绝对值
```

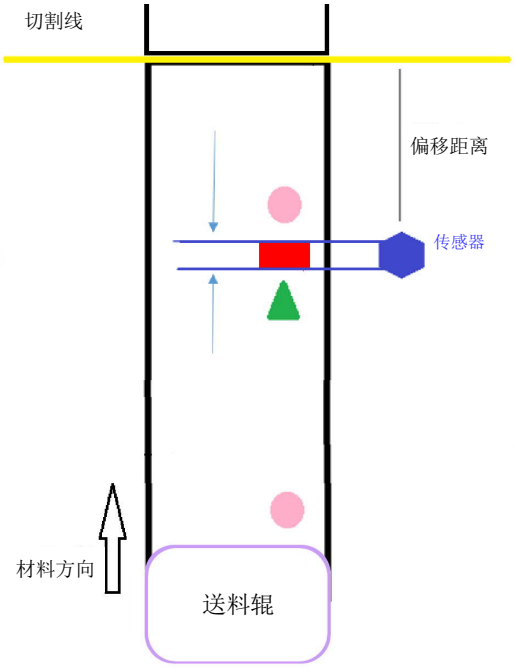
下图显示了该过程的顺序。

在该示例中，我们必须寻找的正确标记以红色矩形表示。它划定了材料应被切割的位置（即，在切割时，红色矩形必须位于切割线上）。两个红色矩形之间的距离是切割长度，因此我们知道正确标记之间有 1000 mm 的距离。圆圈和三角形表示我们必须过滤掉的其他对准标记。

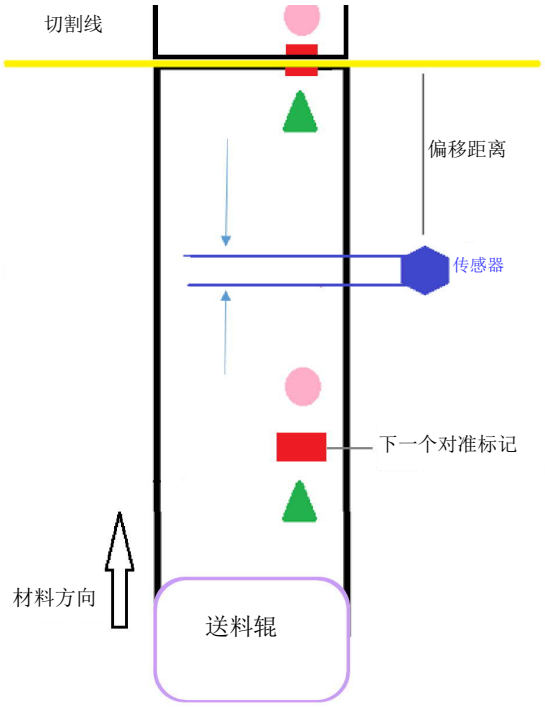
在开始时，辊子开始使用我们预期的 1 米的给料距离将材料移向切割器。由距离切割器偏移距离处的传感器（蓝色六边形）检测所有对准标记。



当材料移动时，传感器最终将在窗口内检测到正确的对准标记，如下图所示。此时，我们知道材料理论上已移动 500 毫米，对准标记位置应为 500 ± 25 毫米。



此时，我们知道材料应向前再移动 500 毫米（偏移距离）以达到切割点。一旦完成 500 毫米的相对移动，红色矩形应定位在切割点上，此时可以进行切割。



有多种方法可以实现相同的结果，但最灵活的是软件方案。但是为了完整性，有时也会使用下文讨论的硬件方案。

硬件控制

一些传感器具有高速使能信号。它用于以电子方式屏蔽或忽略信号，直到传感器的使能信号为真。这可用于控制系统在印刷窗口内启用传感器的情况。在某些 Mint 控制器上，可以使用“快速比较”输出实现此方法。然而，更常见的解决方案以及可以在所有 Mint 运动控制产品上实现的解决方案仍然是通过软件过滤信号。

软件控制

这种方法更为常见，并且它依赖于控制系统来判断所接收的标记是否在容差窗口内，如果不是则忽略它。Mint 中的解决方案只是在接收到快速中断时检查 LATCHVALUE 关键字自动锁定的值是否在此容差窗口内。

用于“启动停止”过程的机器的方法

下面的示例假设 Mint 控制器或驱动器正在控制材料的运动，并且每重复一次，输送轴的绝对位置将置零。这可能在材料静止后在二次过程（例如在材料被切割之后）之后发生。

在下面的示例中，我们可以分析 Latch 事件的代码。假设给料由送料辊执行，由 Mint 程序控制（启动/停止工作方式），其中轴位置在每次切割后置零。

在下一页中，有附带的 Mint 程序的摘录；‘*AN00106 - Multiple reg marks StartStop (LatchWindow).mnt*’

在本代码中，使用固件功能“窗口锁定”对不需要的信号进行过滤。使用它可确保，仅当捕获的位置在“窗口位置”内时，它才是我们可用于实施校正的有效信号。要使用此模式，使用下面这一行代码：

```
LATCHMODE(0) = (_lmAUTO_ENABLE + _lmWINDOW)    ‘启用锁定并启用窗口模式
```

锁定窗口的长度由两个参数设置：起始位置（在本例中是主位置的绝对位置）以及经过该点的距离。 在本例中，窗口是 50mm，起始位置是 475mm。

```
LATCHWINDOWSTART (0)   = _nStartPrintWindow        ‘窗口在 475mm 处开始
LATCHWINDOWDISTANCE(0) = _HalfOfWindow*2           ‘窗口在 475 + 50 = 525mm 处结束
```

每次在传感器检测到标记时，启动 Latch 事件，并且验证锁存位置是否在窗口内。

```
Dim bMarkFound = _false                    ‘使用标志来表示是否找到了标记
Dim nThisPos As Integer = 0            ‘使用缓冲区存储捕获的位置

Const  _axNipRolls = 0

Event LATCH0
If bMarkFound Then Exit Event              ‘一旦找到真正的标记，则忽略其他标记
nThisPos = LATCHVALUE(_axNipRolls)         ‘存储捕获的印刷印标记的位置

'信号有效.....开始采取校正措施
bMarkFound = _true
Print #2, "It is inside the window ", POS(0)
INCA(_axNipRolls) = nThisPos + __nOffsetDistance
GO(_axNipRolls)

End Event
```

我们可以在主任务中使用一个循环，它执行整个切割长度的增量绝对移动，并等待轴完成它。

```
‘ *** TIP *** 使用 Define 将 I/O 替换为有意义的名称！
Define ipRUN_INPUT = Inx(0)

Loop
Pause(ipRUN_INPUT)
bMarkFound = _false                        ‘重置下一个周期的信号发现标志
POS(_axNipRolls) = 0
INCA(_axNipRolls) = _nCutLength            ‘设置默认长度的移动
GO(_axNipRolls)
Pause IDLE(_axNipRolls)                    ‘等待移动完成
End Loop
```

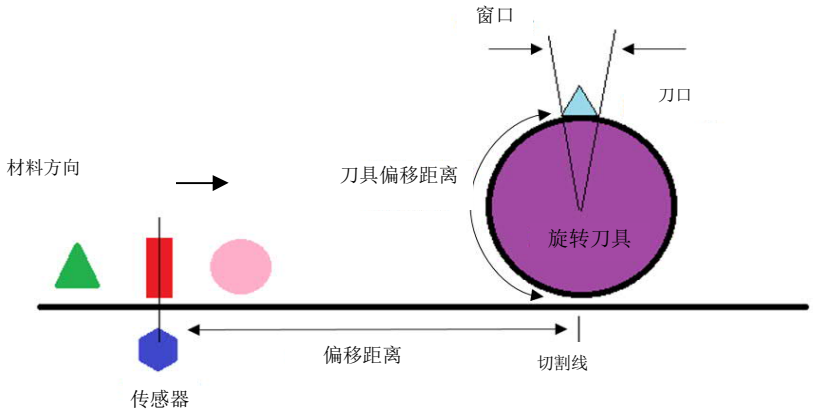
用于连续过程的机器的方法

如果无法工作在启停模式，如在材料持续送进的旋转切割应用中，控制过程略有不同，且有如下特征：

1) 主轴负责将材料拉过机器的旋转切割区域（例如压送辊）。
2) Mint 控制器控制旋转切割器。该切割器被设置为使用 Mint FOLLOW 关键字在“电子齿轮箱模式”下运行
3) 使用 ENCODERWRAP 将旋转刀具的轴编码器值设置为“wrap”。它的值对应于每个印刷长度（即，刀具每旋转一次）接收到的计数，从而允许在接收到印刷对准标记的情况下捕获旋转刀具的绝对位置
4) Mint 控制器使用 Mint OFFSET 关键字计算并应用对准校正
5) 设置一个变量，来指示何时检测到窗口中的有效对准标记。此外，一旦刀具离开窗口，必须清除此“找到标记”变量

本案例的主要任务是设定跟随比率。同时，Latch 事件应该能够验证传感器捕获的标记是否在窗口中，并作出相应的动作。

该过程的剖面图如下图所示：



在示例中，红色矩形仍表示需要检测的对准标记。然而，现在将基于旋转刀具的编码器值（即刀具位置），而不是材料上的标记的位置来定义窗口。刀具偏移距离表示刀具应旋转的距离，以使标记到达切割线时边缘与切割线对齐。

现在代码中的逻辑是，当传感器检测到对准标记时，应该验证刀具的编码器值是否在窗口内。如果刀具不在窗口内，则检测到的标记不正确，因此刀具速度没有动作或校正。

在下文中，有附带的 Mint 程序的摘录；“*AN00106 - Multiple reg marks Continuous (LatchWindow).mnt*”

我们将再次使用“Windowed latching”；

```
LATCHMODE(0) = (_lmAUTO_ENABLE + _lmWINDOW)    ‘启用锁定并启用窗口模式
```

但这一次，我们捕获的位置是以未经换算的计数表示的旋转刀具的位置。因此，这一次我们将采用与之前相同的原理，但将以计数作为单位。

此外，已从启动模块中移除此代码的位置，以允许更简洁和灵活的编码；

```
Dim iWindowLength As Integer = 50*(nKnifePpr/_nCutLength)
Dim iIdealEnc As Integer =(_nOffsetDistance)*(nKnifePpr/_nCutLength) '500mm 定义理想的主编码器值
Dim iStartPrintWindow As Integer = iIdealEnc - (iWindowLength/2) '475mm 定义印刷窗口的开始位置
Dim iEndPrintWindow As Integer = iIdealEnc + (iWindowLength/2) '525mm 定义印刷窗口的结束位置

‘确保在应用校正之前完成相关计算

LATCHWINDOWSTART (0)   = iStartPrintWindow      ‘窗口在 62259 个计数（475mm）时开始
LATCHWINDOWDISTANCE(0) = iWindowLength          ‘窗口在 62259 + 6553 = 68812 个计数（525mm）时结束
LATCHENABLE(0) = 1
```

主要任务代码可能类似于下面的示例：

```
Const _axMaster = 2          '定义我的主编码器
Const _axCutter = 0          '定义刀具轴

‘配置刀具以跟随主编码器的速度
MASTERSOURCE(_axCutter) = _msENCODER
MASTERCHANNEL(_axCutter) = _axMaster

‘以相对于主速度的指定比率运行刀具
While ipRUN_INPUT
FOLLOW(_axCutter) = fFollowRatio
End While
```

锁定事件应配置为在传感器检测到标记时启动。在 Latch 事件中，使用 LATCHVALUE 关键字捕获刀具的实际位置。一旦我们计算出实际位置和理想位置之间的差值，我们就使用 OFFSET 命令进行对准校正。

OFFSET 关键字允许我们相对于切割器的实际速度执行相对位置的移动，根据偏移值加速或减速。OFFSET 命令的指定相对距离将是实际位置和理想位置之间的差值。请注意，仅当轴当前未执行另一个 OFFSET 移动时，才能执行 OFFSET 移动。

```
Event LATCH0

‘一旦找到真正的标记，就忽略其他标记。
If bMarkFound Then Exit Event

‘检测到印刷标记时，存储刀具的编码器值
nThisEnc = LATCHVALUE(_nCutter)

'-------------------------------------------------------------------
‘ 如果捕获的位置在开始和结束窗口位置之内，则
‘ 这是一个有效的信号，我们可以使用这个位置来实现校正
'-------------------------------------------------------------------
‘信号有效.....开始采取校正措施
bMarkFound = _true
‘计算理想值和捕获的编码器值之间的差值
nCorrection = nThisEnc - nIdealEnc

‘检查是否可以执行偏移移动
        If Not(AXISMODE(_nCutter) And _mdOFFSET) Then
        Print #2, "It is correcting"
        OFFSET(_nCutter) = nCorrection
        GO(_nCutter)
        End If
End Event
```

已经在启动模块中设置的是可以应用偏移的距离，它不应超过刀具轴的缠绕值。

```
‘设置偏移移动的偏移距离
OFFSETDISTANCE(_nCutter) = 10000
```

一旦旋转刀具离开印刷窗口，则使用哨兵指令来清除 bMarkFound 标志。这将确保下次刀具进入窗口并找到正确的标记时，它将能够正常执行事件代码。

‘标记通道 0 被配置为当切割器编码器经过窗口的末端时执行 mint 事件

```
SENTINELACTIONMODE(0) = _samOFF
SENTINELSOURCE(0) = _cpENCODER
SENTINELSOURCEPARAMETER(0) = _axCutter
SENTINELSOURCE2(0) = _cpOFF
SENTINELTRIGGERMODE(0) = _ctmRISING
SENTINELACTION(0) = _saEVENT_MINT      ‘这会触发 DPR 事件
SENTINELACTIONPARAMETER(0) = 1
SENTINELTRIGGERVALUEFLOAT(0, _stvLOW) = _nEndPrintWindow
SENTINELTRIGGERVALUEFLOAT(0, _stvHIGH) = _nEndPrintWindow
SENTINELACTIONMODE(0) = _samMANUAL
```

需要执行 Mint DPR 事件，以清除 Mark found 变量并重置哨兵指令。

```
Event DPR
bMarkFound = _false
SENTINELACTIONMODE(0) = _samMANUAL
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