Motion Control Products

Application note
Multiple registration marks

AN00106

Rev G (EN)

In a system where a corrective action, or system event, is initiated on the detection of a print mark or other signal, it is often a difficult to determine the correct signal to use when there are multiple signals during each repeat length or machine cycle

Introduction

Process colour printing machines, for instance, often generate several markings along the outside edge of the material. This may be used for quality checking, colour alignment checks, batch numbering etc. yet this is often the simplest area on the product to use for registration and subsequent alignment of other processes (such as additional colour prints, embossing, cut to length or perforation).

The method of dealing with this is simple in principle but can often prove troublesome to implement. Mint provides a flexible platform for solving this problem with its powerful high-level language and fast interrupt handling features.

Print Mark Window

The registration mark or signal we are interested in will be positioned at a known distance from the cut position of the material, usually known as “offset distance”, and as such when we look at the material we naturally look in the region where we expect to find the mark. The principle by which we filter out any unwanted signals from other printed marks is simply based on this knowledge of expected position. Of course, we must allow for errors and so accept a mark within a certain tolerance band. This tolerance band is often referred to as the ‘print window’ or ‘print marker window’.

The implementation of this is based on knowing the offset distance (which we convert to encoder counts) and the “print window” or tolerance band.

For example, imagine that we have an application where the material is moved by feed rolls towards a cutter. Several print marks have been printed into the material for following processes downstream, so we have to filter them and catch only the correct one. Our product cut length is 1 metre long and we have a sensor located 0.5 metre from the knife. The distance from the sensor to the knife is going to determine my “offset distance” and is already known (in our case 0.5 m). As we know the position that our correct mark should be from the knife, we could define the window or tolerance band at ±2.5 cm of the offset distance. This allows us to filter any other mark which are not within this band.

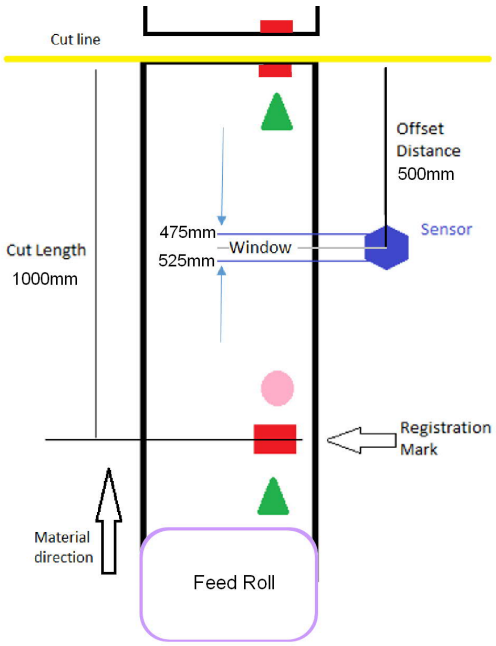
The constants for our application example could be as follows:

```
Const _nCutLength As Integer = 1000          'Use a variable to set the cut length in mm
Const _nOffsetDistance As Integer = 500       'Define the distance from the sensor to cut line in mm
Const _nStartPrintWindow = 475               'Define the start of the Print window -25 mm from Offset distance in absolute terms
Const _nEndPrintWindow = 525                 'Define the end of the print window +25 mm from Offset distance in absolute terms
```

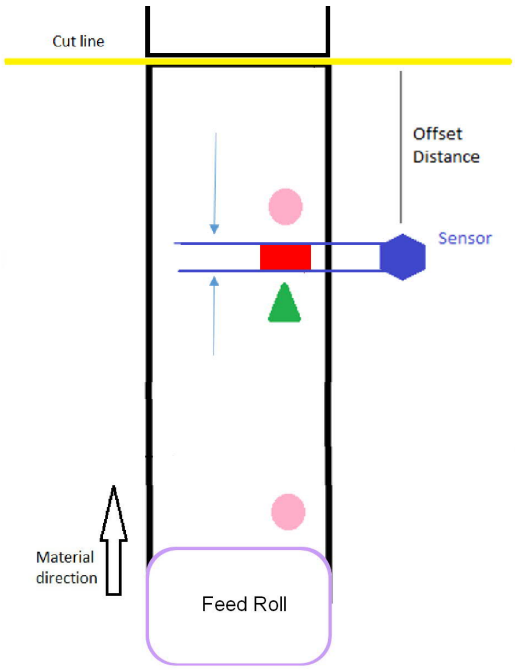
The diagrams below show the sequence of the process.

In the example the correct mark that we must look for is the red rectangle, which delimits the position where the material should be cut (i.e. the red rectangle must be located on the cut line at the moment of the cut). The distance between a red rectangle and the next one is the total cut length, so we know that there is 1000 mm between correct marks. The circle and triangle represent other registration marks that we have to filter out.

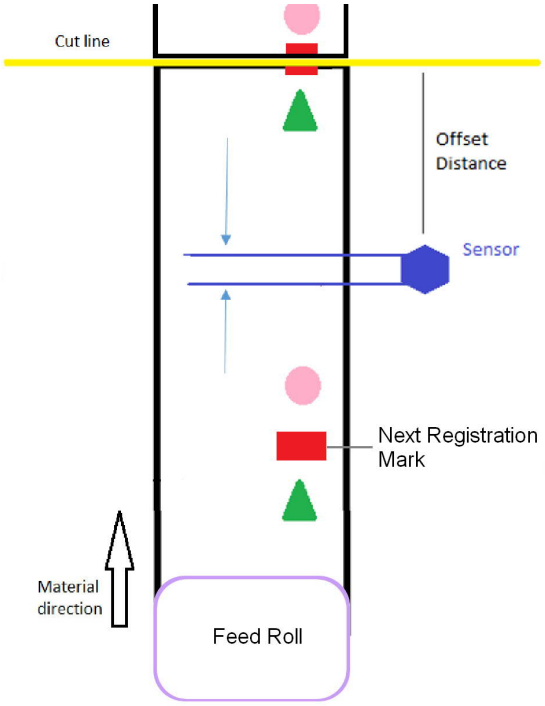
Initially, the rolls start moving the material towards the cutter using our expected feed distance of 1 metre. The sensor (blue hexagon) which is located at the offset distance from the cutter is detecting all the registration marks.



As the material moves the correct registration mark will eventually be detected by the sensor, in turn, within the window, as it shows on the diagram below. At this point, we know that the material has theoretically been moved 500 mm, and the registration mark position should be at 500 ± 25 mm.



At this moment, we know the material should be moved a further 500 mm (the offset distance) forwards to reach the cut point. Once this relative move of 500 mm is completed the red rectangle should be positioned at the cut point, and therefore the cut can be carried out.



There is more than one method of achieving the same result but the most flexible is a software implementation, however for completeness the hardware solution discussed below is sometimes used.

Hardware control

Some sensors have a high speed enable signal, which is used to electronically mask or ignore signals until the enable signal is true. This can be used where the control system will enable the sensor during the print window. This method can be implemented on some Mint controllers using the 'fast compare' output. However, the more common solution, and one which can be implemented on all Mint motion control products, involves filtering of the signals via software.

Software control

This approach is more common and relies on the control system to determine if the received mark is within the tolerance window and if not simply ignore it. The solution in Mint is simply to check whether the value latched automatically by LATCHVALUE keyword, on reception of a fast interrupt, falls within this tolerance window.

Method for machines where the process is 'start stop'

The below example assumes that the Mint controller or Drive is controlling the motion of the material and that once per repeat length the absolute position of the axis is reset to zero. This can occur after the secondary process (e.g. after the material has been cut) when the material is stationary.

In the following example we can see how the code of the latch event could be. It is supposed that the material feeding is carried out by feed rolls, in turn these are controlled by our Mint program (on a start/stop basis) where the axis position can be reset to zero after every cut.

On the following page you will see excerpts from the attached Mint program; '*AN00106 - Multiple reg marks StartStop (LatchWindow).mnt*'

The filtering of the of unwanted signals in this code us used by the firmware feature 'Windowed latching'. Using this will ensure that only if the captured position is within the "window positions", is it a valid signal which we can use to implement a correction. To use this mode we include the line below

```
LATCHMODE(0) = (_lmAUTO_ENABLE + _lmWINDOW)     'Enable latching and enable Window Mode
```

The length of the latch window is set by two parameters the start position which is an absolute position in this case of the master position and the distance past this point in this case the window is 50mm and the start position is 475mm.

```
LATCHWINDOWSTART (0)   = _nStartPrintWindow        'Start the Window at 475mm
LATCHWINDOWDISTANCE(0) = _HalfOfWindow*2           'End the window at 475 + 50 = 525mm
```

Every time that the sensor detects a mark the latch event is launched, and the mark position is verified whether it is in the window or not.

```
Dim bMarkFound = _false                 ‘Use a flag to indicate whether the mark was found
Dim nThisPos As Integer = 0             ‘Use a buffer to store the captured position

Const  _axNipRolls = 0

Event LATCH0
  If bMarkFound Then Exit Event         'Once real mark is found ignore others
  nThisPos = LATCHVALUE(_axNipRolls)    'Store the position of the captured print mark

'Signal is valid…start corrective action
bMarkFound = _true
Print #2, "It is inside the window ", POS(0)
INCA(_axNipRolls) = nThisPos + __nOffsetDistance
GO(_axNipRolls)

End Event
```

In the main task we could have a loop which perform an incremental absolute move of a whole cut length and waits for the axis to complete it.

```
‘*** TIP *** Use Define to replace I/O with meaningful names!
Define ipRUN_INPUT = Inx(0)

Loop
  Pause(ipRUN_INPUT)
  bMarkFound = _false                   ‘Reset the signal found flag for next cycle
  POS(_axNipRolls) = 0
  INCA(_axNipRolls) = _nCutLength       ‘Set up the move for the default length
  GO(_axNipRolls)
  Pause IDLE(_axNipRolls)               ‘Wait for move to complete
End Loop
```

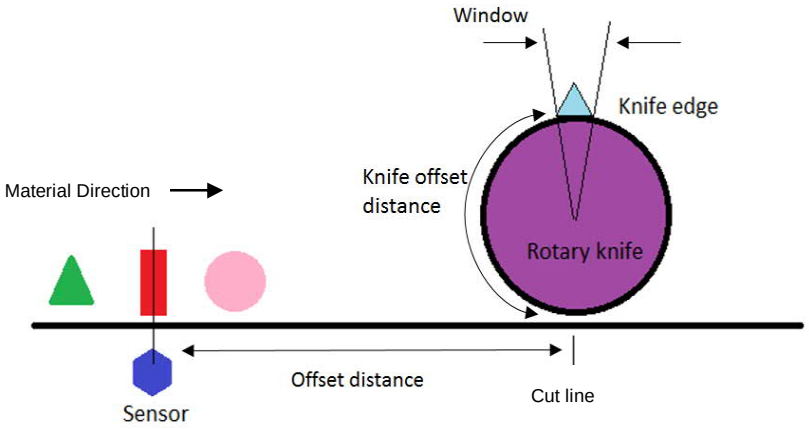
Method for machines where the process is continuous

If it is not possible to do this, for example in a rotary cutter application where the material never stops, then a slightly different process may be used with the following features:

1) A master axis is responsible for pulling the material through the rotary cutter section of the machine (e.g. nip rolls).
2) The Mint controller takes control of the rotary cutter which is set to operate in ‘electronic gearbox mode’ using the Mint FOLLOW keyword
3) The rotary knife axis encoder value is set to ‘wrap’ using ENCODERWRAP at a value corresponding to the number of counts received for each Print Length (and hence each cutter revolution) thereby allowing the absolute position of the rotary knife to be captured on receipt of print registration marks
4) The Mint controller calculates and applies the registration corrections using the Mint OFFSET keyword
5) A variable is set that indicates when a valid register mark in the window has been detected and this ‘mark found’ variable must be cleared once the knife is out of the window

The main task in this case will be responsible for setting the follow ratio; meanwhile the latch event should be able to verify whether the captured mark by the sensor is in the window and act accordingly.

The profile view of the process is shown on the diagram below:



In the example, the red rectangle is going to be our correct registration mark again. However, the window now is going to be defined based on the rotary knife encoder value (i.e. the knife position), rather than the position of the mark on the material. The knife offset distance represents the distance that the knife should rotate to make the edge being positioned aligned to the cut line when the mark reaches it.

The logic that should be implemented in the code is that when the sensor detects a registration mark, the knife encoder value should be verified whether it is within the window or not. If the knife is not in the window the mark detected is not the correct one and therefore there is no action or correction on the knife velocity.

Below you will see excerpts from the attached Mint program; '*AN00106 - Multiple reg marks Continuous (LatchWindow).mnt*'

Once again, we will use 'Windowed latching';

```
LATCHMODE(0) = (_lmAUTO_ENABLE + _lmWINDOW)     'Enable latching and enable Window Mode
```

This time though our captured position is that of the rotary knife which is in unscaled counts so we will apply the same principle as before but this time our units will be in counts.

Also, the position of this code has been moved from the start up block to allow for more elegant and flexible code;

```
Dim iWindowLength As Integer = 50*(nKnifePpr/_nCutLength)
Dim iIdealEnc As Integer =(_nOffsetDistance)*(nKnifePpr/_nCutLength) '500mm Define ideal Master encoder value
Dim iStartPrintWindow As Integer = iIdealEnc - (iWindowLength/2) '475mm Define the start of the Print window
Dim iEndPrintWindow As Integer = iIdealEnc + (iWindowLength/2) '525mm Define the end of the print window

'Ensure dependent calculations are done before corrections are applied

LATCHWINDOWSTART (0)   = iStartPrintWindow        'Start the Window at 62259 counts (475mm)
LATCHWINDOWDISTANCE(0) = iWindowLength            'End the window at 62259 + 6553 = 68812 counts (525mm)
LATCHENABLE(0) = 1
```

The main task code could be similar to the example below:

```
Const _axMaster = 2          'Define my master encoder
Const _axCutter = 0          'Define the cutter axis

'Configuration for the cutter to follow the master's encoder velocity
MASTERSOURCE(_axCutter) = _msENCODER
MASTERCHANNEL(_axCutter) = _axMaster

'Run the cutter at the specified ratio with respect to the master velocity
While ipRUN_INPUT
  FOLLOW(_axCutter) = fFollowRatio
End While
```

The latch event should be configured to be launched when the sensor detects a mark. At that moment the actual position of the cutter is captured using the LATCHVALUE keyword, once we have calculated the difference between the actual and ideal position we proceed to carry out the registration correction using the OFFSET command.

The OFFSET keyword allows us to perform a relative positional move with respect to the actual velocity of the cutter, accelerating or decelerating according to the offset value. The assigned relative distance for the OFFSET command will be the difference between actual and ideal position. Note that an OFFSET move can only be performed if the axis is not currently performing another offset move.

```
Event LATCH0

  'Once the real mark has been found, ignore the others.
  If bMarkFound Then Exit Event

  'Store the encoder value of the cutter when the print mark is detected
  nThisEnc = LATCHVALUE(_nCutter)

  '-------------------------------------------------------------------
  ' If the captured position is within the start and end window positions then
  ' this is a valid signal and we can use this position to implement a correction
  '-------------------------------------------------------------------
    'Signal is valid…start corrective action
    bMarkFound = _true
    'Calculate the diference between the ideal and captured encoder value
    nCorrection = nThisEnc - nIdealEnc

    'Check whether offset move can be performed
        If Not(AXISMODE(_nCutter) And _mdOFFSET) Then
          Print #2, "It is correcting"
          OFFSET(_nCutter) = nCorrection
          GO(_nCutter)
        End If
End Event
```

Already set in Startup block is the distance over which the offset can be applied, this should not exceed the wrap for the cutter axis.

```
  'Set the offset distance for offset move
  OFFSETDISTANCE(_nCutter) = 10000
```

A sentinel is used to clear the bMarkFound flag once the rotary knife is out of the print window. This is going to make sure that the next time that the knife enters the window and finds the correct mark it is going to be able to execute the event code.

'The sentinel channel 0 is configured to execute a mint event when the cutter encoder passes by the end of the window

```
SENTINELACTIONMODE(0) = _samOFF
SENTINELSOURCE(0) = _cpENCODER
SENTINELSOURCEPARAMETER(0) = _axCutter
SENTINELSOURCE2(0) = _cpOFF
SENTINELTRIGGERMODE(0) = _ctmRISING
SENTINELACTION(0) = _saEVENT_MINT    'This fires the DPR event
SENTINELACTIONPARAMETER(0) = 1
SENTINELTRIGGERVALUEFLOAT(0, _stvLOW) = _nEndPrintWindow
SENTINELTRIGGERVALUEFLOAT(0, _stvHIGH) = _nEndPrintWindow
SENTINELACTIONMODE(0) = _samMANUAL
```

The Mint DPR event should execute code which will clear the Mark found variable and reset the sentinel for the next time.

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

Contact us

For more information please contact your local ABB representative or one of the following:

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