Motion Control Products

Application note
Continuous clutch follow

AN00100

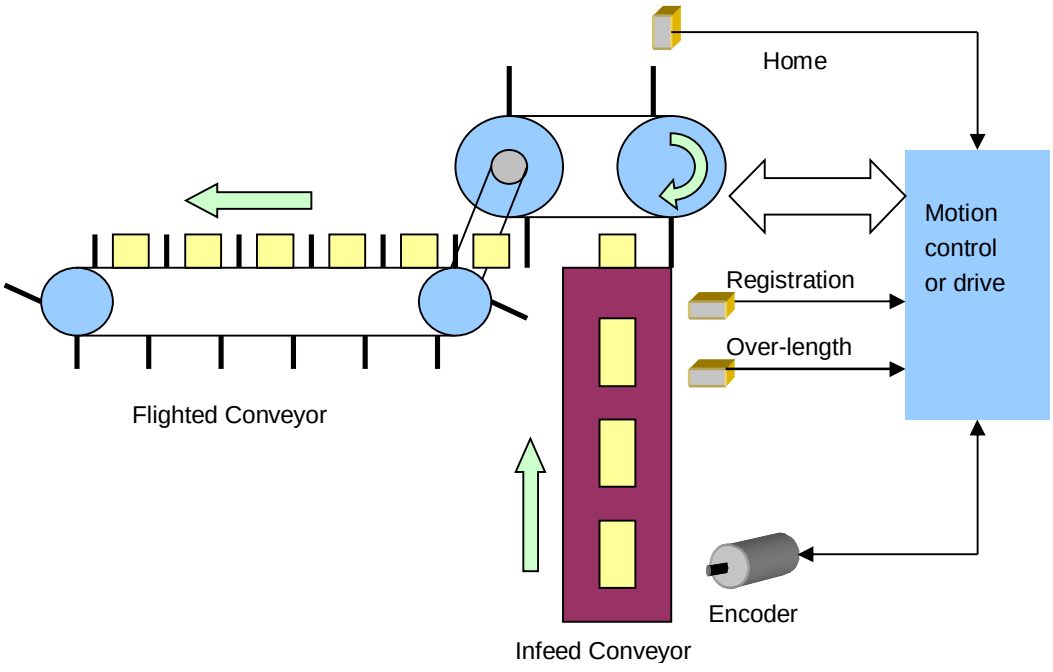
Rev D (EN)

Intelligent servo controlled systems are often used to replace mechanisms driven purely via mechanical gearing. Use of an electronic system allows precise position lock to be maintained at all times and gear ratios can be modified, at will, at the touch of a button, even on the fly. Even mechanical clutch systems can be emulated. Where a mechanical system may produce unpredictable results in terms of distances traveled during the engagement and disengagement of the clutch, an electronic system allows these to be precisely specified.

Introduction

Mint provides a simple to use FOLLOW keyword to allow axes to run at specified gear ratios with respect to a master axis or encoder. The FOLLOW keyword can be configured in a number of pre-determined modes, one of which allows clutching distances to be defined when changing following ratio (continuous or extended clutch mode). Phase adjustments to the axis, to allow product registration for example, whilst following is also possible using the Mint OFFSET command.

The illustration below, shows the layout of a machine where this mode of following may prove useful (a paddle used to sweep products between conveyors running perpendicular to each other):



Another example of a system that would typically use a clutched follow (to allow the axis to be geared but then stopped in a known position) would be a scroll (screw) infeed system, pitching products at regular spacing into a downstream system.

In our example paddle system above, products are fed onto an Infeed Conveyor from the production line. The production line output is such that the gap between products is reasonably regular and is nominally 50% of the product length.

The Infeed Conveyor drives an encoder so that product position can be determined and tracked (this is connected to the master/auxiliary encoder input on the motion controller or intelligent drive).

Two photoelectric sensors detect products as they approach the paddle sweeping section of the machine. The sensor nearest the paddles operates as a 'position latch' input (fast input). This input latches the position of the paddle axis, when the front edge of a product is detected, to enable registration of the paddle to each product.

The second sensor is used to detect over-length products (e.g. two products joined together). This sensor is adjusted to suit product length such that if both sensors are activated simultaneously an over-length product must be present.

Directly in line with the Infeed Conveyor (behind the paddle's line of sweep) is a backstop that doubles as a reject gate. Solenoid valves, controlled by digital outputs, are used to open/close the backstop so that over-length products can be rejected if required.

A BSM servomotor is used to drive the paddles. A mechanical coupling also allows a flighted conveyor to be driven by the same motor. This is designed so that one product sweep is associated with one product flight.

A proximity sensor allows the paddle axis to be homed on start-up or after an Emergency Stop.

As the paddle axis has to generally follow the Infeed Conveyor encoder, has to be able to adjust position for registration and stop and restart when required it is an ideal candidate for 'Continuous Clutch Mode Following'.

Paddle/Sweep axis

The general principle of operation for the paddle axis (axis 0) is that the paddles must traverse one paddle pitch for each product that is received. This relationship can be described as ***encoder following***. The follow ratio can be determined from the formula below:

Follow ratio = paddle pitch / product pitch

(where paddle pitch and product pitch are expressed in user units….this may be quadrature encoder counts or we may decide to scale both our paddle axis and our master encoder to represent linear travel in mm).

For this example, we will assume we are working in encoder counts on the paddle axis and the master encoder (i.e. our scale factor for each is 1). If the paddle pitch on our machine is 12000 counts and the product pitch (distance between lead edges of products) is 4000 counts, then the follow ratio would be calculated as 3.

The distance between paddles is fixed and hence we set our ENCODERWRAP for this axis to match this distance (i.e. 12000).

Starting the paddles

When the machine first starts, the paddles must be positioned so that:

a) a programmable number of products can initially be rejected through the reject gate
b) the paddles can be ramped up to speed to avoid mechanical shock and to reduce peak torque requirements

This position is treated as a zero position and the homing routine ensures that the axis starts motion from this point on power up or after an Emergency Stop by datuming to a switch and then performing an offset move.

Once the required number of products has been rejected, the paddles need to ramp up to the following ratio. This is possible by using an 'extended clutched mode' follow set via the Mint code;

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

This mode allows the user to specify a master distance over which the slave axis will ramp up to the following ratio. This can be considered as a Flying Shear and Mint even allows this motion to be specified as a FLY segment (instead of specifying the follow ratio) if desired.

The generic formula for flying shear segments is shown below:

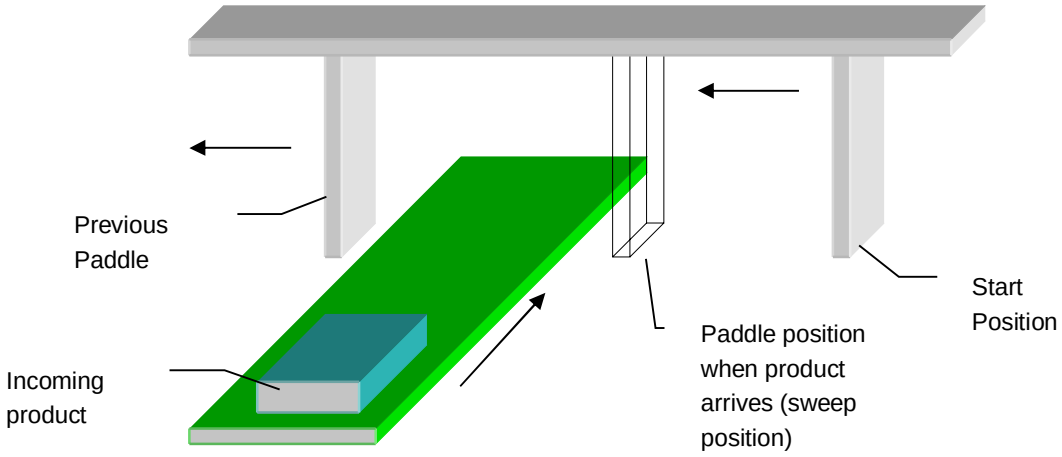
FLY = ((Initial Ratio + Final Ratio)/2) * MASTERDISTANCE

(see application note AN00116 for an introduction to the FLY keyword).

When we start the paddles the initial gear ratio is 0 (they are stopped). Our final ratio is the follow ratio (3 in this example). Hence the formula above shows that the relationship between the slave distance traveled on startup (FLY) and the incoming product distance traveled during startup (MASTERDISTANCE) will be:

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

Let us examine what we know so far:



The diagram above shows the starting position for the paddles (our encoder value will be zero at this point). The diagram also shows the position we want the paddles to be in when the incoming product reaches the backstop (the reject gate acts as a backstop when closed) – we have called this position the sweep position.

By measuring the distance (in encoder counts) between the starting position and the sweep position we can determine the required FLY segment for this axis. Substituting this into our previous formula will therefore allow us to calculate the corresponding MASTERDISTANCE:

FLY = (3/2) * MASTERDISTANCE

MASTERDISTANCE = FLY * 2/3

If the distance traveled by the paddles to reach the sweep position is 3000 counts then the MASTERDISTANCE will be 2000 counts.

The start has to be triggered by detection of the incoming product (via a photo-electric sensor for example). We can therefore conclude that this sensor must be located at least 2000 conveyor axis counts away from the backstop.

In this example the Infeed Conveyor produces 20 counts per mm of linear travel – hence the sensor needs to be at least 100mm from the backstop.

It is important to consider the minimum product length at this stage. If the product length is such that more than one product will fit in the gap between the sensors and the backstops (remembering to include the nominal gap between products) then we have two options:

a) consider setting a new start distance to reduce the minimum sensor distance

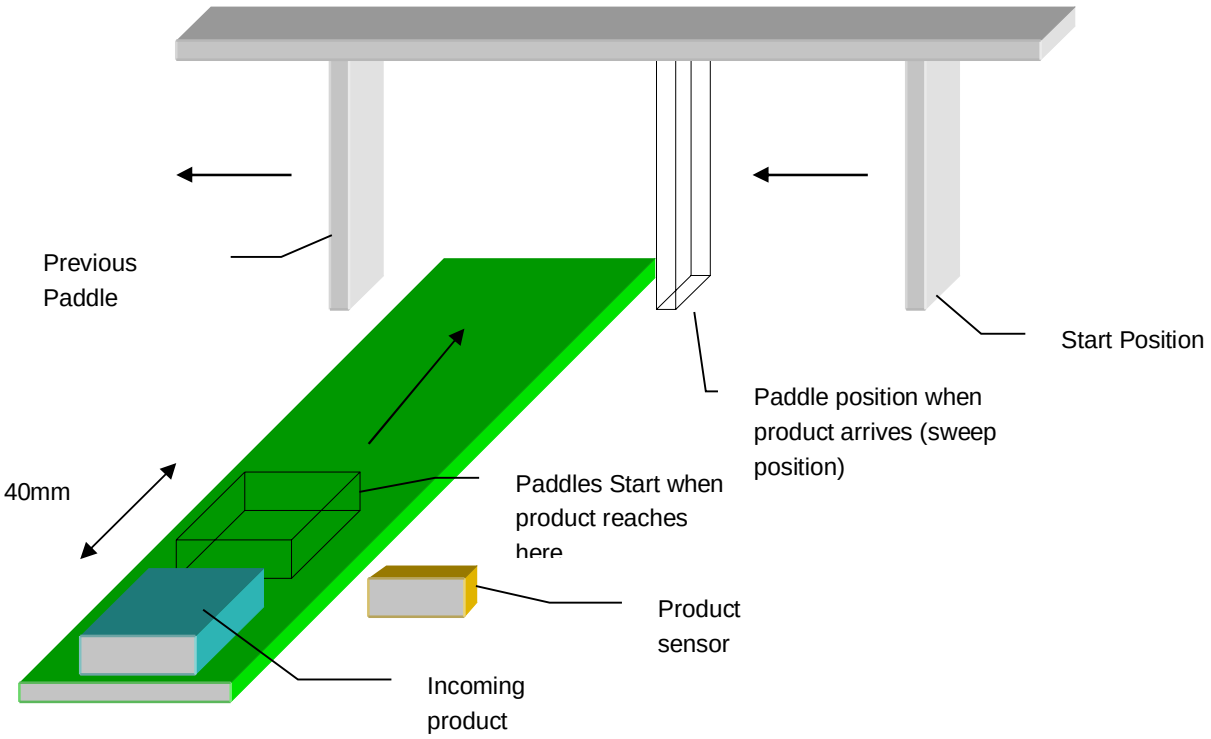
b) continue with the existing setup – this however will then require the software to 'buffer' the product triggered signals (e.g. registration corrections) so that they can be applied to the correct paddle

For this example, our product pitch was 4000 counts or 200mm. We can therefore conclude that the start position is acceptable at present.

However, let's assume that there is a mechanical restriction preventing the sensor from being mounted 100mm from the backstop. Instead it has to be mounted 140mm (or 2800 counts) from the backstop. This still meets the requirement that only one product fits between the sensor and the backstop, but now we cannot trigger the clutched follow or flying shear segment at the point of product detection. We can't modify our starting flying shear segment as this will change the start position of the paddles and then products may not be able to pass through to the reject gate.

Instead we need to delay starting motion until the product reaches the point at which it is 100mm from the backstop.

This can be achieved with a second flying shear segment, where the MASTERDISTANCE is the product delay distance and the FLY segment is specified as zero (so the paddles do not move).



If we assign variable names to all of our system parameters, then we can easily calculate these and code the logic to start the paddles when a product activates fast input 0.

```
Dim fFollowRatio As Float                ' Final follow ratio for paddles
Dim nPaddleStartDistance As Integer      ' Distance paddles move between start and sweep
Dim nDelayDistance As Integer            ' Distance to wait before paddles start moving
Dim nRampDistance As Integer             ' Distance over which paddles are ramped up to speed
Dim nPaddlesStopped As Integer = _True   '  Flag used to decide whether to start paddles
Const _nEyeToBack As Integer = 2800      ' Distance from product sensor to backstop

'Use flying shear segments when changing ratio
FOLLOWMODE(0) = _fmContinuous_Clutch

'Calculate product distances for paddle start sequence
nRampDistance = Int(( nPaddleStartDistance * 2 ) / fFollowRatio )
nDelayDistance = _nEyeToBack – nRampDistance

Loop
  'Main parent task
End Loop

Task StartPaddles
  nPaddlesStopped = _False               ' Consider the paddles to have started
  MASTERDISTANCE(0) = nDelayDistance     ' Wait for product to reach start point
  FLY(0) = 0: GO(0)
  MASTERDISTANCE(0) = nRampDistance      ' Paddles now ramp up to speed
  FOLLOW(0) = fFollowRatio                       ' Or FLY(0) = (fFollowRatio/2) * MASTERDISTANCE
End Task

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

Stopping the paddles

There are a number of conditions that may require the paddles to be stopped in a position where products can be rejected – i.e. the paddles have to be stopped in the position from which they first start (at an encoder value of 0).

By coding the stop logic in its own task we can 'trigger' the stop to occur from a number of different events or conditions throughout the application whilst the other tasks continue to execute. For example, we may request the paddles to stop if an over-length product is detected (if Fast input 0 and Input 1 are both on simultaneously).

As we are using an extended clutch mode follow we can use the FLY command to bring the paddle axis to a halt at a prescribed position. We have seen from the generic formula for flying shears that:

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

When the paddles are continually running (following) we have already seen that they are following the infeed conveyor axis at a follow ratio of 3 – hence 3 is our initial ratio. To bring the paddles to a halt the final ratio must be 0.

We can therefore conclude that the formula for our FLY segment and associated MASTERDISTANCE will be of the form:

FLY = (3/2) * MASTERDISTANCE

If the paddle axis comes to a halt at the correct position the axis encoder value should read 0. Hence by capturing the current axis position when we start to stop the paddles we can derive the distance to be traveled during the stop – i.e. the FLY segment value.

For example:

If the encoder value for the paddles is read as 5000 when we initiate the stop (remembering that the axis has an ENCODERWRAP value of 12000) then the distance to travel to an encoder value of 0 is (12000 – 5000) = 7000 counts.

Once we have calculated the required FLY segment our generic formula can be used to calculate the associated MASTERDISTANCE.

Our Mint paddle stop task should therefore appear as follows:

```
Task StopPaddles
  Dim nStopFly As Integer                          'Declare variable to store fly segment
  nStopFly = ENCODERWRAP(0) – ENCODER(0)           'Calculate paddle travel distance
  MASTERDISTANCE(0) = nStopFly * 2/3               'Calculate associated masterdistance
  FLY(0) = nStopFly                                'Issue the fly to stop paddles
  GO(0)
End Task
```

Registering the paddles

Once the paddles are following the Infeed Conveyor at our nominal following ratio they should continually collect and sweep the incoming products through 90 degrees. However, the success of this process is dependent on the incoming products being regularly and accurately spaced. Any variation in the position of the incoming products will require that the paddles are either advanced or retarded to ensure successful sweeping still occurs.

This advancing or retarding of position (whilst still maintaining an overall follow ratio) can be considered as *product registration*.

Mint allows product registration to be implemented with ease via the OFFSET keyword. The OFFSET keyword requires the user to specify a correction distance and Mint automatically calculates the necessary speed adjustments to perform this correction.

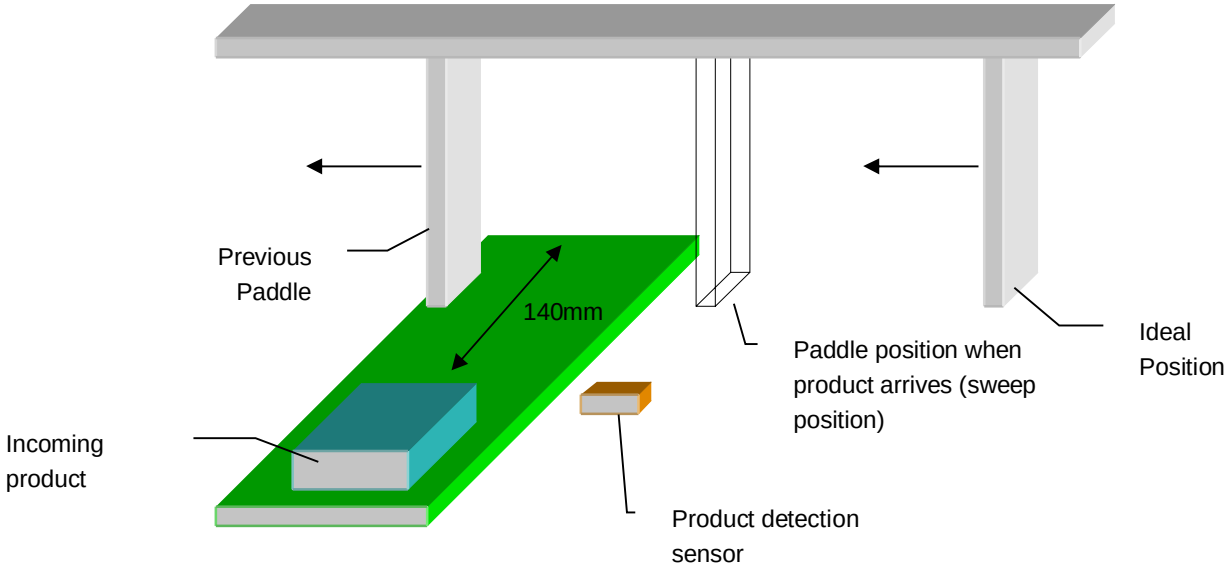
We have already seen how EVENT LATCH0 could be used to detect incoming products and start the paddles following if necessary. This same input can be used to capture the position of the running paddles and determine whether the position of the incoming product requires the paddles to be advanced or retarded.

The configuration of the latching will vary depending on the controller to be used. In our case we will assume an intelligent servo drive (e.g. MicroFlex e190 with Mint memory module) is to be used (as we are only controlling one motor/axis). In this case fast input 1 is configured to capture the wrapped encoder for the paddles (encoder 0) via the following Mint code segment:

```
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
```

All that remains now is to calculate the registration requirements for each incoming product.

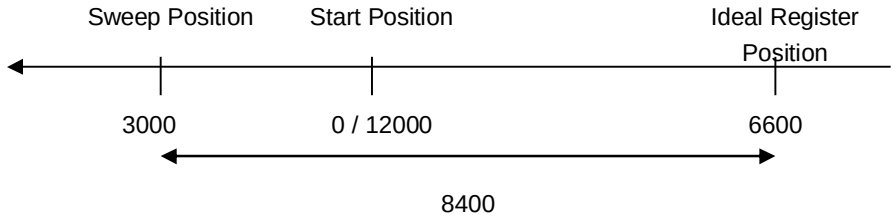
Let's have a look at what's typically happening when a product is detected:



We already know from our work in starting the paddles that the incoming product will be 140mm (or 2800 counts of the Infeed Conveyor axis) away from the backstop when the product detection sensor detects it. We have also seen from this work that when the product reaches the backstop the paddles have traveled 3000 counts past their initial start position (where the initial start position has an encoder value of 0).

Once the paddles have completed their initial ramp up to speed they follow the Infeed Conveyor axis at a fixed ratio (3:1). We can therefore conclude that, if the paddles are in register, they should travel 3 * 2800 (8400) counts whilst the product travels to the backstop.

Therefore, the ideal position for the paddles when a product is detected is 8400 counts 'back' from encoder value 3000. The diagram below illustrates how we calculate the ideal register position from this information:



Having calculated the 'ideal' position we can compare this against the position captured automatically by the fast interrupt every time a product is detected. The difference between the captured position and our 'ideal' position can then be applied as an OFFSET to ensure the paddles remain in register.

The following Mint code segment illustrates how we can combine our start, stop and registration work into a single Mint application:

```
Dim fFollowRatio As Float                 ' Final follow ratio for paddles
Dim nPaddleStartDistance As Integer             ' Distance paddles move between start and sweep
Dim nDelayDistance As Integer             ' Distance to wait before paddles start moving
Dim nRampDistance As Integer              ' Distance over which paddles are ramped up to speed
Dim nPaddlesStopped As Integer = _True    '  Flag used to decide whether to start paddles
Dim nIdealPos As Integer                  ' Paddle ideal position on detection of product
Const _nEyeToBack As Integer = 2800       ' Distance from product sensor to backstop

'Use flying shear segments when changing ratio
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

'Calculate product distances for paddle start sequence
nRampDistance = Int(( nPaddleStartDistance * 2 ) / fFollowRatio )
nDelayDistance = _nEyeToBack – nRampDistance

'Calculate ideal register position for paddles
nIdealPos = ENCODERWRAP(0) - ((_nEyeToBack * fFollowRatio) – nPaddleStartDistance)

Loop
  'Main parent task
End Loop

Task StartPaddles
  nPaddlesStopped = _False                    'Consider the paddles to have started
  MASTERDISTANCE(0) = nDelayDistance          'Wait for product to reach start point
  FLY(0) = 0: GO(0)
  MASTERDISTANCE(0) = nRampDistance           'Paddles now ramp up to speed
  FOLLOW(0) = fFollowRatio                          'Or FLY(0) = (fFollowRatio/2) * MASTERDISTANCE
End Task

Task StopPaddles
  Dim nStopFly As Integer                     'Declare variable to store fly segment
  nStopFly = ENCODERWRAP(0) – ENCODER(0)      'Calculate paddle travel distance
  MASTERDISTANCE(0) = nStopFly * 2/3          'Calculate associated masterdistance
  FLY(0) = nStopFly                           'Issue the fly to stop paddles
  GO(0)
End Task

Event Latch0
  'This event is generated every time a product is detected (lead edge)

  'Start the paddles if they are stopped
  If nPaddlesStopped Then
    Run StartPaddles
    Exit Event          'No point processing any further code if just starting
  End If

  'Stop the paddles if an overlength product is detected
  If INSTATEx(1) Then
    Run StopPaddles
    Exit Event          'No point processing any further code if paddles are being stopped
  End If

  'If this point is reached the paddles must be following and registration corrections can be applied as necessary
  OFFSET(0) = WRAP(nIdealPos – LATCHVALUE(0), -ENCODERWRAP(0) / 2, ENCODERWRAP(0) / 2)
  GO(0)

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