Overview

Mint is an easy to use high-level programming language rich in features, including facilities to write modular, block-structured programs. The Mint language includes subroutines, functions, tasks, structures (user-defined data types), conditional statements, looping statements, etc.

Mint also provides an extensive range of specialized functions to interface to the hardware of Baldor controllers and includes a variety of keywords to vastly simplify the implementation of complex motion control solutions.

Mint is a sequential language (i.e. a particular line of code is not executed until the previous line has completed execution) that also provides the user with the ability for parallel processing (via multi-tasking) and event/interrupt driven operation (e.g. via digital input events) making the language as a whole incredibly flexible and particularly suited to machine and motion control applications.

This application note provides an insight into some of the techniques which can be used to easily implement a Cut to Registration machine such as that illustrated by the animation at www.baldormotion.com/solutions.

Cut to Registration Animation

The Cut to Registration animation has the following features:

- Servo controlled Nip Roller, Product conveyor and Knife axes (Brushless AC Servo Motors)
- NextMove e100 motion controller
- Three MotiFlex e100 AC servo drives connected over EPL (Ethernet PowerLink)
- Registration sensor connected to a fast latch input on the MotiFlex e100 which is controlling the Nip Roller

The sample code shows how, using the fast latch functionality of the e100 products triggered by the registration sensor, we can easily implement cutting to registration.

Set up

Axis Configuration

The example application uses the following axes...

(a) Nip Roller (Used to pull and position the uncut product through the machine)
(b) Knife (An axis is used to control the knife’s movement up and down but this could be achieved using a pneumatic valve driven by a digital output)
(c) Conveyor (To move away the cut product)

These axes are configured in the Mint Startup block.
**Fast Latch Configuration**

To set up fast latching on e100 products a selection of LATCH...... keywords are used.

The latch needs to be set up in the parameter table of the MotiFlex e100 as shown below. The latch input must be connected to the MotiFlex e100 on which we wish to capture the position from, in our case the Nip roller. Digital inputs 1 and 2 are available to be used as fast latch inputs on the MotiFlex e100.

![Parameter Table](image)

Alternatively using the code below in the NextMove e100 program we can set channel 0 (_nNipRollLatch is defined as being equal to 0, see later section on Program constants) to be triggered by the rising edge of digital input 2, and latch the position of the Nip roller axis. This method of setting information on a remote node is known as redirection.

```vba
Dim RemoteDrive As Controller = {_busETHERNET, _ndNipRoller}  
RemoteDrive -> LATCHTRIGGERCHANNEL(_nNipRollLatch) = 2  
RemoteDrive -> LATCHTRIGGERMODE(_nNipRollLatch) = _ltmINPUT  
RemoteDrive -> LATCHTRIGGEREDGE(_nNipRollLatch) = _ltePOSITIVE_EDGE  
RemoteDrive -> LATCHSOURCE(_nNipRollLatch) = _lsAXIS_POSITION  
RemoteDrive -> LATCHSOURCECHANNEL(_nNipRollLatch) = _axNipRoller  
RemoteDrive -> LATCHMODE(_nNipRollLatch) = _lmAUTO_ENABLE  
LATCHENABLE(_nNipRollLatch) = _true  
```

The latch is set to automatically re-enable itself and initially enabled ready for use.
System Configuration

Along with using one of the methods above to configure the latch on the MotiFlex e100 (Controlled node) it must also be set up on the NextMove e100 (Manager node). To do this you need to go to the System Configuration Wizard and either start a new configuration or edit an existing configuration file. When on the Configure Ethernet Powerlink Devices screen (shown below) you will see a list of managed devices, their node address, update type and mapped resources.

Configure ETHERNET Powerlink Devices

If the controller is the manager of the network, use the Add, Edit and Delete buttons to create a list of all network devices that the Manager will communicate with. For each one you will be prompted to select information and functionality for each device.

<table>
<thead>
<tr>
<th>Device</th>
<th>Node Address</th>
<th>Update</th>
<th>Mapped Resources</th>
</tr>
</thead>
<tbody>
<tr>
<td>MotiFlex e100</td>
<td>1 (Hex 01)</td>
<td>Normal</td>
<td>1 Axis, 1 Latch</td>
</tr>
<tr>
<td>MotiFlex e100</td>
<td>2 (Hex 02)</td>
<td>Normal</td>
<td>1 Axis</td>
</tr>
<tr>
<td>MotiFlex e100</td>
<td>3 (Hex 03)</td>
<td>Normal</td>
<td>1 Axis</td>
</tr>
</tbody>
</table>

You will need to have 3 MotiFlex e100’s mapped as shown above. The node address will need to match the node address set on the front of the drives (Note this is not the axis number). If you do not have any devices listed click on the Add Device button and select the Node ID as set on the front of the drive and the device type (MotiFlex e100), then click OK. You will then be presented with a Resource mapping window (shown below) where you can define local hardware, remote hardware and process data.
By double clicking on Axis 0 you can map a manager (NextMove e100) axis to this axis. You will be prompted with a pop up (shown below) that allows you to set the axis number that you will use in the mint program. You can also select how the device is profiled, select Manager node profiled. Finally if this is the Nip Roller axis you are mapping click the Add Latch button. This gives you another pop up window that lets you map a latch channel on the MotiFlex e100 to the NextMove e100. We will be using latch channel 0 on the MotiFlex e100 and latch channel 0 on the NextMove e100.
**Scale factors**

The scale factor allows each axis to be scaled into engineering units, for ease of use. The scale factor is a division factor that is applied to all motion variables for an axis (speed, acceleration, move distances, etc.).

The scale factor can be set using the `SCALEFACTOR` keyword. The number of pulses coming from the feedback device are divided by the scale factor and used to define the size of the user unit (uu). By default, the scale factor is 1 which means all axis motion is recorded in encoder quadrature counts.

In the conveyor example (below), the `SCALEFACTOR` of 60 is derived by starting at the conveyor belt and working back to the motor:
To move the conveyor belt 1 millimetre (the intended user unit), the pulley must turn 0.005 revolutions (0.005 x 200mm).

To turn the pulley 0.005 revolutions, the motor will have to turn 0.015 revolutions, since there is a 3:1 step-down gearbox between the motor and pulley.

There are 4000 quadrature counts per motor revolution, so 0.015 turns will equate to 60 quadrature counts.

The Knife axis has a scalefactor of 11 which allows moves to be demanded in degrees of a cycle, where one cycle (360 degrees) causes the knife to lower, cut and then return to its upper position.

The Nip roller axis has a scalefactor of 80 which similarly to the conveyor allows moves to be demanded in linear mm.

**Program Constants**

To make the code easier to read and to allow the program to be easily updated at a later date some constants are used to define certain aspects of the application.

The axis numbers are all allocated constants as well as the node number of the Nip roller drive and the latch channel of the Nip roller. By utilizing these constants, instead of ‘hard coding’ numerical data later in the program, any changes made to the axis numbers can be easily updated by changing the value of this declaration.

```
Const _axNipRoller As Integer    = 0   'Axis drawing product through
Const _axKnife As Integer         = 1   'Knife axis to cut product
Const _axConv As Integer         = 2   'Conveyor to move cut product

Const _ndNipRoller As Integer    = 1   'Node number of Nip roller drive
Const _nNipRollLatch As Integer  = 0   'Nip roller latch channel
```
Variable Declarations

This application only has one variable which is the value of the target position calculated from the latch value.

Dim fTargetPos As Float = 0

Comms Locations

The initial move distance and offset length values are given comms locations as shown below.

Define cmInitMove = COMMS(1)
Define cmRegOffset = COMMS(2)

This means these values can be set by a PLC, Host application or on a HMI panel and edited while the program is running. This enables the length between cuts and therefore the length of the product to be changed during operation.

The value assigned to cmInitMove is the distance the nip roller should move the product feed after a cut is performed. This distance should be long enough to feed a registration mark to the registration sensor. The value assigned to cmRegOffset is the known distance the nip roller must move the product feed after a registration mark is sensed so that the product feed stops in the correct cut position.

Main program

Within the main program loop the knife axis performs a cut and the output conveyor moves away the cut product. This is achieved with the following code.

POS(_axNipRoller) = 0
INCA(_axNipRoller) = cmInitMove
GO(_axNipRoller)
Pause IDLE(_axNipRoller)

MOVER(_axKnife) = 360
GO(_axKnife)
Pause IDLE(_axKnife)

MOVER(_axConv) = 2000
GO(_axConv)
Pause IDLE(_axConv)

We start by setting the position of the Nip roller to zero and then telling it to feed the product through to the initial target position of cmInitMove (declared previously). The program then waits for the Nip roller to become idle (stop moving). If the value of cmInitMove is correctly set so that a registration mark reaches the registration sensor the latch event will be called. This then changes the target position so that the product is fed to the correct position under the knife. The Nip roller becomes idle after finishing the move and the knife can then perform the cut. Once the cut has finished the conveyor is indexed to move the product away.
The whole process is then repeated for the next cut to registration.

**Latch event**

When the latch event is called the following code is executed.

\[
f\text{TargetPos} = \text{LATCHVALUE(\_\text{axNipRoller})} + \text{cmRegOffset} \\
\text{INCA(\_\text{axNipRoller})} = f\text{TargetPos} \\
\text{GO(\_\text{axNipRoller})}
\]

The variable \(f\text{TargetPos}\) is assigned the latched position of the Nip roller axis plus the registration offset length. This value is then loaded as the new target position of the Nip roller and the axis is told to move to this new position with the GO command.