

Rotary Knife

Camming Advanced Programming Workshop

Requirements

- Files
 - Desktop Demo Archive
 - Machine Archive
 - **RK OPERATION** POU (to import later)
 - **RK IO** POU (to import later)

Goals

- Create a functional high-speed rotary knife application using electronic Cam
- Learn the fundamental building blocks to any electronic cam based application
- Work with the SFC programming language.

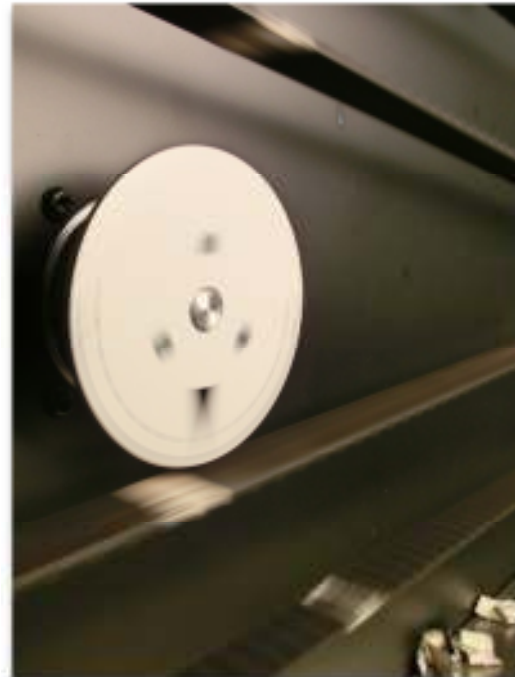
Outline / Lab Overview

General Sequence and Training Philosophy: “Learning by Doing”

- Create part of the machine operation.
- Test that part manually using debug mode and demo switches.
- Repeat until all parts work individually.
- Write code to sequence the operation of the working parts.

This document will guide the participant through the following steps:

- **Startup (2 hours):** Understand the application, run the demo, initialize variables.
- **Blend (3 hours):** Generate cam profiles and smoothly ramp in and out.
- **Shift (4 hours):** Calculate phase shift to control starting point and part length
- **Buffer (1 hours):** Capture positions of conveyor belt when product is sensed.
- **Sequence (4 hours):** Automate the calculation and execution of blend, shift, and buffer
- **Machine (1 hour):** Transfer the code to the machine for further analysis.



KEY

IV. Section Title

? Question

A. Action or Information

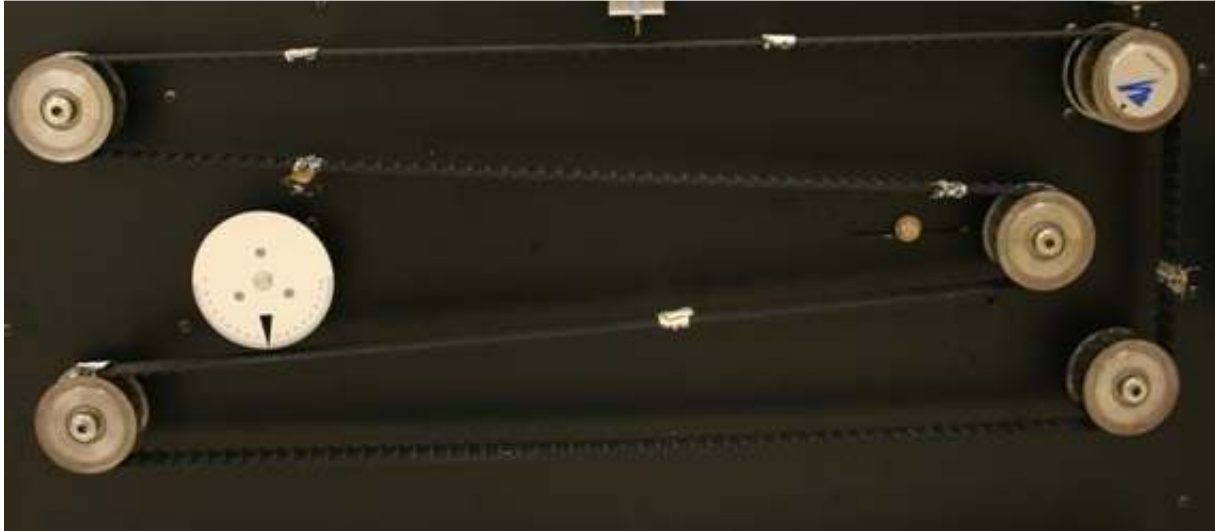
Formatting: **VARIABLE** *POU*

FUNCTIONBLOCK *FunctionBlockInput*

Startup

I. System Overview

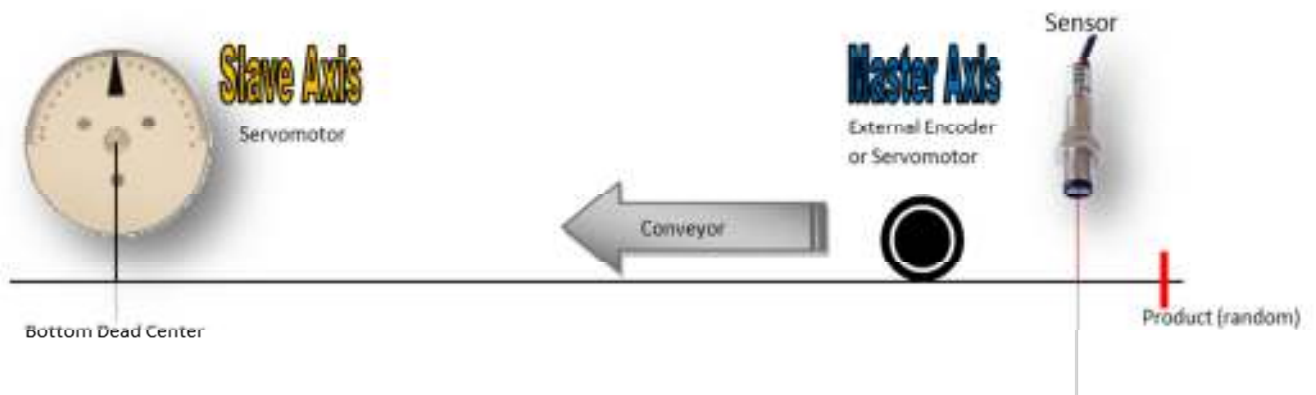
Become familiar with the application and equipment.



- A. Look at the machine to answer these questions
- ? What Yaskawa products are used?
 - ? Do the motors have any gearbox or holding brakes?
 - ? Which axis on the system do you suspect is NOT part of the rotary knife application?
 - ? Locate the optical sensor “product sensor”. The orange wire for this sensor connects on the back panel to an orange breakout terminal. To what two devices and input pin numbers is the sensor connected? This will be important later on.
 - ? What applications could this demo machine simulate?

B. Machine Description and Operation Concept

1. Consider the following simplified diagram of the machine

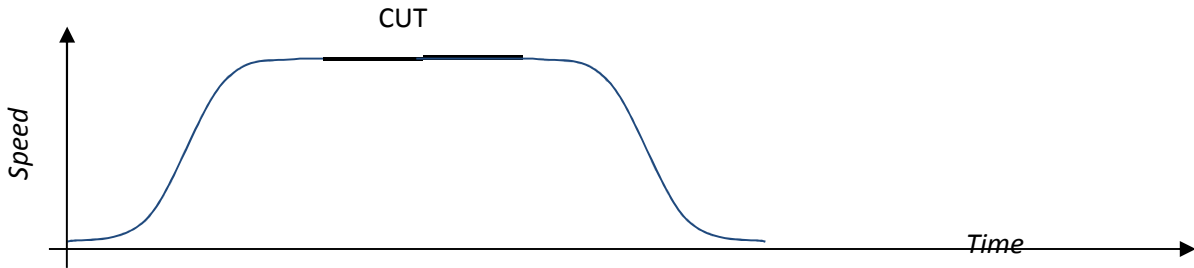


2. Operation

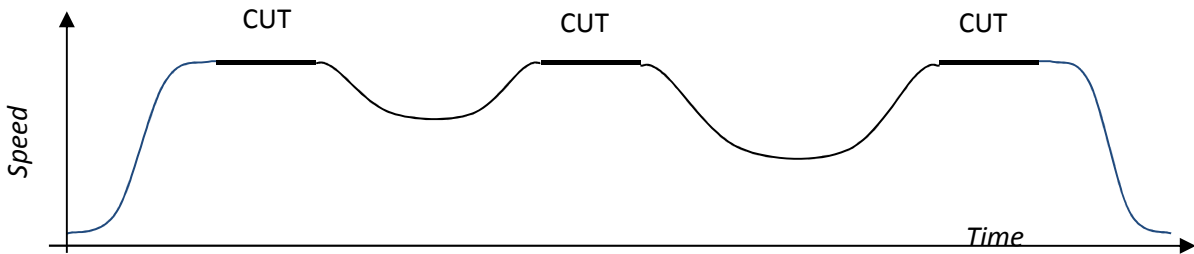
- a. Material is moving continuously. In the diagram this is from right to left.
- b. The knife “blade” is to synchronize with the product throughout the entire cutting cycle as it moves along the conveyor belt, even if the material were to speed up or slow down.
- c. The “product” is represented by a registration mark printed on the conveyor belt. If there is material stretching or slippage in a web of material, this mark ensures it will be cut at the correct location relative to the other graphical printing on the material.
- d. The knife will rotate in only one direction to cut the material while moving. In this diagram that is the clockwise direction (CW).
- e. The space between products is random and can vary greatly.
- f. The location “Bottom Dead Center” (BDC) is where the knife fully cuts the product. BDC is both a knife angle in degrees and a product position in inches.

3. Expected Motion Profile

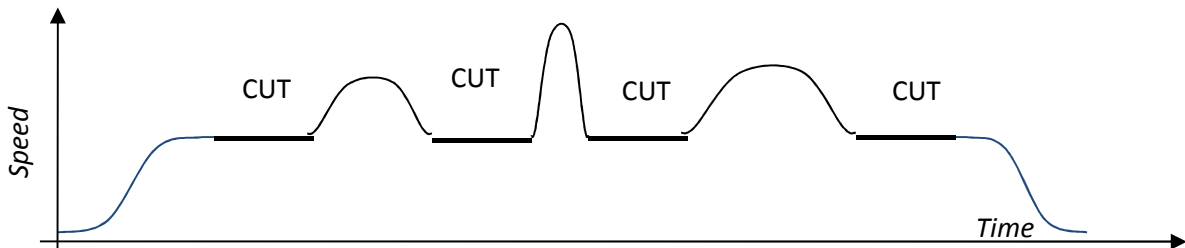
- **Case 1: Very distantly spaced. The knife stops at UP position between cuts, waiting for another cut mark to be sensed.**



- **Case 2: Distantly spaced. The knife keeps moving, slowing down between cuts.**



- **Case 3: Closely spaced. The knife keeps moving, speeding up between cuts.**



- **Case 4: Closely and distantly spaced. The knife slows down or speeds up between cuts.**

? What might the speed vs time profile look like if the products are both closely and distantly spaced?



II. Solution Approaches

A. Set The Speed Once

At first glance, the solution to this application seems simple. Read the speed of the conveyor and move the motor forward at that speed for the appropriate amount of time.

Breakout Lesson

This method, however, does not consider the fact that the motor must perfectly align with an exact position of the material on the conveyor in order that the cut be made accurately. So some type of algorithm would have to monitor and match the position of the conveyor with the servomotor. This can be cumbersome, and the effectiveness is limited.

The possibility also exists that the conveyor could change speed while the motor is moving. The speed of the knife must exactly match the speed of the conveyor for the whole time that cutting is taking place. Therefore, setting the speed once is not sufficient.

B. Update The Speed Continuously

So the simple fix then seems to be to continuously monitor the conveyor speed and update the motor speed to match it during the critical part of the move. This could be done with a simple program loop that reads the speed of the conveyor and sets the same speed to the motor.

This approach does work to a limited degree, but it will soon become apparent that the motor is not following the speed exactly. This is because there is too much delay. The encoder on the conveyor is sending pulses, which must be converted to speed by dividing pulses by time. Time is necessary to calculate speed, so a time delay is unavoidable.

Compare this speed-matching problem to the familiar experience of driving an automobile. To match the speed of another car, a driver naturally tries to maintain a constant distance between cars. It would be far less effective to match speedometer readings, even if a wireless display of the other car's speedometer was available. The same is true in servo applications. If you match position, you will also match speed. But matching speed does not ensure you will match position.

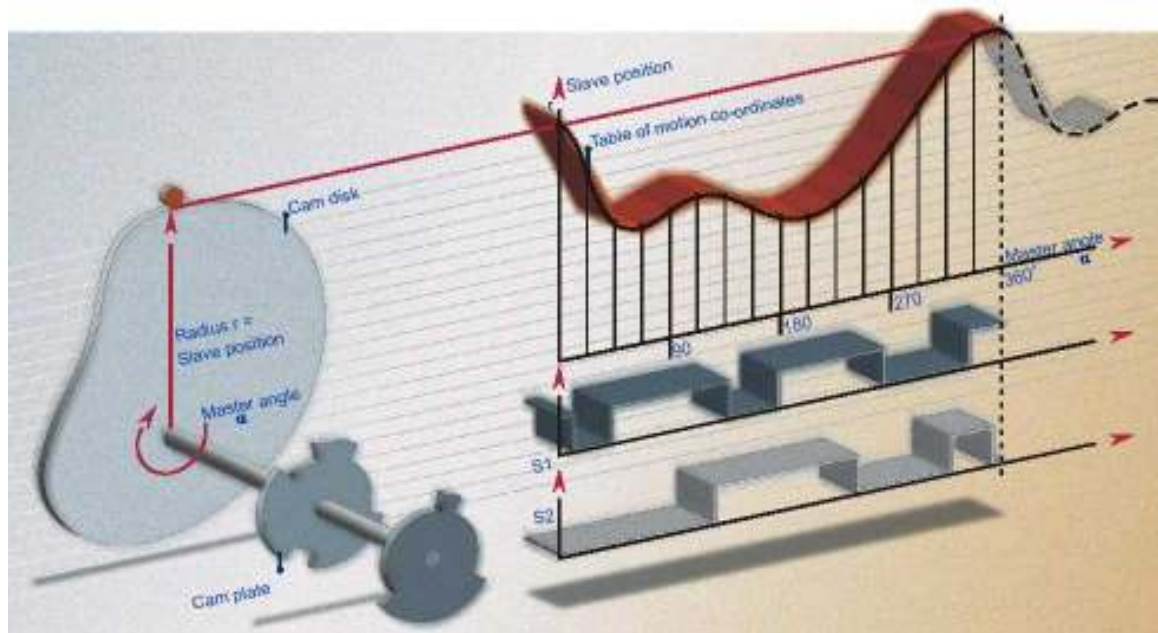
C. Use Electronic Gearing

Electronic gearing is designed for just that purpose - to match the motor's position to that of an external axis. This external axis is referred to as the **master axis**. In the case of this application, the encoder on the conveyor is the master axis. The **slave axis** is the motor being controlled. If geared properly, the slave will follow the master exactly.

The approach would then be to smoothly transition into electronic gearing for the time given, and quickly position back for the next cut. This would ensure that the position is followed during the cutting part of the move. The problem is that it is very difficult to synchronize exactly. Additionally, there is always a shock if electronic gearing is engaged while the motor is in motion. Steps can be taken to counteract these problems and make gearing work. But they end up making program development time consuming and cumbersome.

D. Electronic Cam (To Be Implemented)

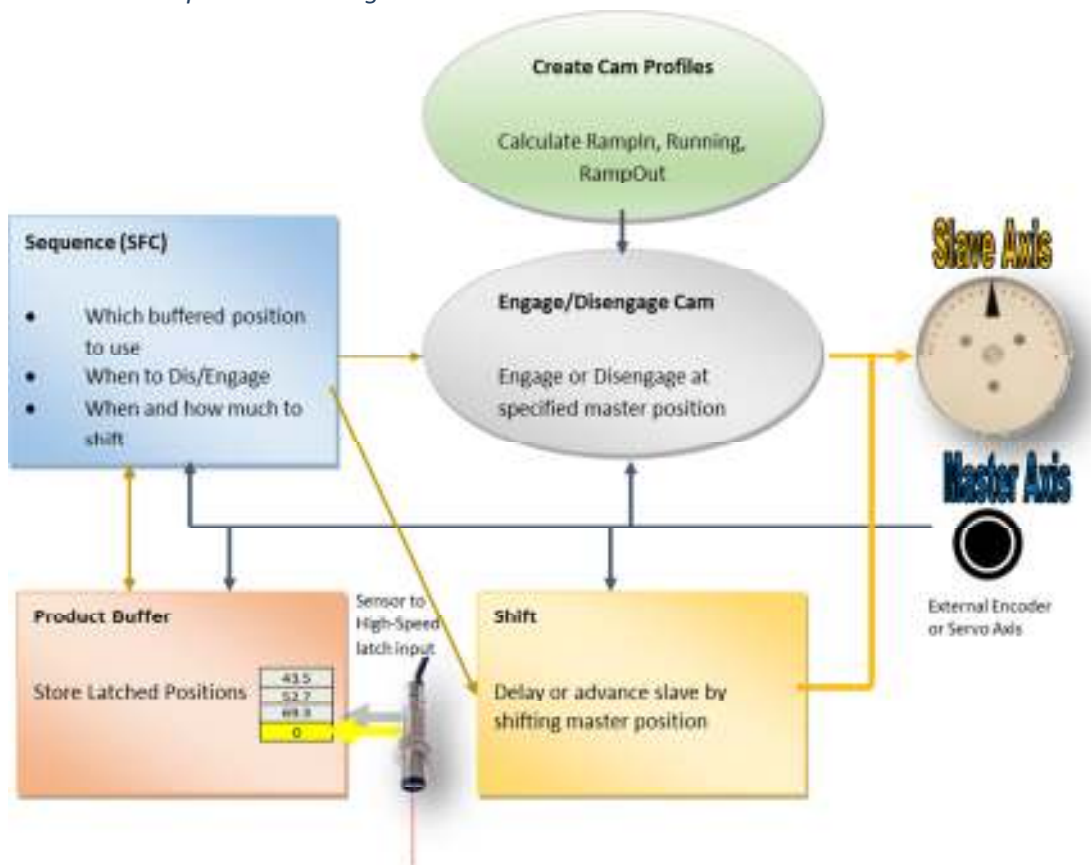
Gearing and electronic CAM are similar. With gearing, the position of the slave is controlled to be directly proportional to the position of the master, in proportion to a constant value called the gear ratio. With electronic cam, the position of the slave is individually controlled according to specific positions of the master rather than at a constant proportion. These positions are defined in a **CAM TABLE**. As the master position increases, the slave position could increase, decrease, or stay the same. If the master is running at a constant speed, the slave positions can be set to correspond to a desired velocity profile. But if the master were ever to change speed, stop, or reverse, electronic CAM ensures that the slave motor is always in exactly the right position relative to the master, regardless of any velocity profile expected from the slave.



With electronic cam, the issue of conveyor speed disappears. It is totally out of the equation. The only reason to consider the conveyor speed is to calculate an **EXPECTED VELOCITY PROFILE** for the slave motor. The expected velocity profile is the velocity profile you *expect* the slave to have when the master axis moves at a constant speed. But this profile will not be followed if the master does not move as expected.

E. Programming Block Diagram And Class Flow

- *This is a preview of what is to come*
- *General strategy is to verify operation of the individual sections of the program and then sequence them together.*



1. Create cam profiles
 - a. Use the **CamGenerator** block
 - b. Use Cam Editor
 - c. Straight-line “running cam”
 - d. Ramp in and Ramp out
2. Engage and Disengage the cam
 - a. Use the **CamBlend** block
3. Shift the cam
 - a. Use **Y_CamShift** block
 - b. Pre-shift, Running-Shift, Un-shift
4. Capture product positions
 - a. **ProductBuffer** block
5. Sequencing
 - a. Use an SFC program to operate the working parts of the program into an automatic sequence.

III. Startup

- *Get the desktop demo up and running.*

A. Confirm Communications to each device

1. MPiec Web UI
2. VIPA Web UI

B. Install project archive in the Web UI

1. Check with instructor – this may have already been completed.
2. Perform “clean install” of `Archive_RotaryKnifeDemoConfig.zip` and reboot as indicated.
3. Write Drive Parameters and reboot.

Note: The 3-slot base and LIO-06 in slot1 will be offline until you connect to the machine, which causes the controller to generate alarms. These alarms can be cleared or ignored.

C. Start a new MP3300iec project

1. Save under new name `RotaryKnife`
2. Adjust Hardware Configuration for the project
 - a. Go online with Hardware Configuration
 - b. Choose “Save Startup Configuration On the Controller”
 - c. Modify VIPA SLIO 01 IP Address to match your demo IP address.
 - d. Use “Get Online Configuration” to refresh connected slices.
 - e. Remember online save

D. Hard po

- E. Import the **RK OPERATION** POU and **RK IO** POU
1. Run **RK OPERATION** POU in SlowTsk
 2. Run **RK IO** POU in FastTsk (if not already)

- F. Open the worksheets of the imported POU and become familiar with their purpose

1. The **RK IO.TB_PANEL** allows operation without switches and provides convenient access to other variables, similar to a watch window
2. **RK IO.Logic** WORKSHEET CONCEPT: The **G_** variables will execute blocks in other POU. All requests to these **G_** variables are routed through the **RK IO** POU, whether from a physical input or the sequence code. This strategy allows manual execution, which is useful for debugging.

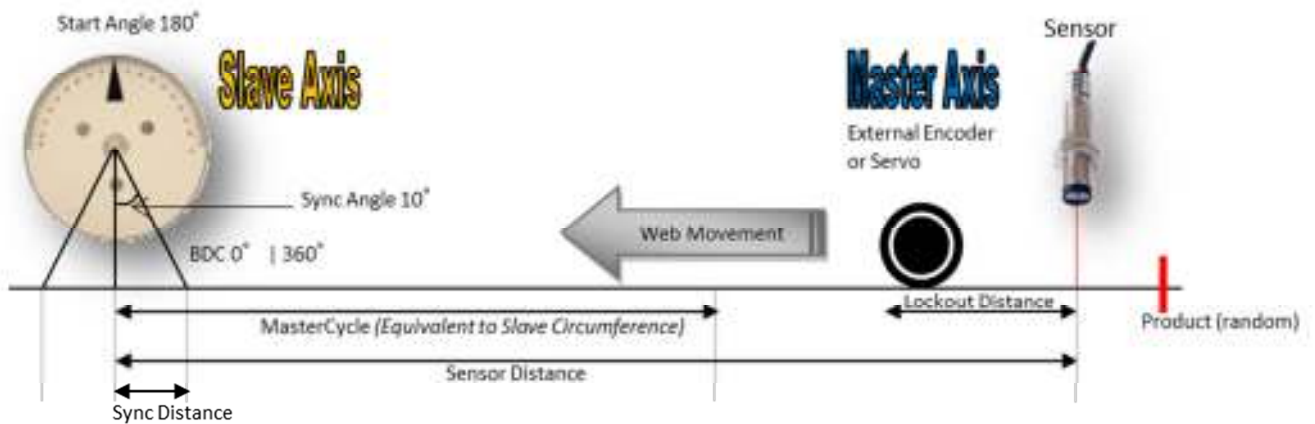
*** How to Import a POU**

1. Click on the Logical POU's folder (in project tree)
2. File → Import → Extended IEC61131-3 Import, POU, OK
3. Navigate to the class materials directory, select POU's, OK
4. Add an instance of the POU to a task
5. Right-click the instance, choose "Create Global..."

- G. Run the Code and Confirm Operation

1. DI-0 = Servo ON
2. DI-2 = Jog Conveyor at speed in **G_JOGVELOCITY_CONVEYOR**
 - a. Test different speeds.
 - b. 5 [in/sec] is a good speed for application development.
 - c. For **G_JOGVELOCITY_CONVEYOR** enter an initial value of 5.0 and mark "retain"
3. DI-3 = Move knife to **G_STARTANGLE**
 - a. Set the zero position of the knife (pointing down) using the **RK OPERATION.ZERO** worksheet in debug mode. Under normal operation you will not have to do this again.
 - b. Set **G_STARTANGLE** to 180.0 [deg]. More on this in the next section.

IV. Initialization
 A. Machine Description and Operation



B. Performance Requirements

1. Conveyor runs as fast as 400 [ft/min] = 80 [in/sec]
2. Conveyor runs at a speed that may be adjusted by the operator. The material may suddenly slow down or stop.
3. The knife must stop smoothly, pointing straight up while waiting for the product.
4. The knife must be synchronized with the material during the entire cutting cycle. The default cutting cycle for the default material is 10 [deg] on each side of BDC.
5. Motor sizing shows the knife can move the remaining 340 [deg] in 0.042 [s]

C. Important Measurements, Specifications and Calculations.

- ? What is the diameter of the knife [in] ? Measure it. Calculate the circumference. *Note: The cam master cycle will be set to the circumference of the knife.*
- ? Sensor Distance can be measured using the feedback position of the conveyor. It has been found to be 28.5 inches. Is this distance realistic? Measure it.
- ? What is the position of the Start Angle?
- ? What is the Sync Angle?
- ? What distance on the conveyor is proportional to the Sync Angle? In other words, how much knife circumference is there in 10 [deg]? This is the Sync Distance.

- ? How far does the conveyor move at max speed while the knife makes the 340 [deg] return in 0.042[s]?

- ? Based on the above, what is the shortest part that can be cut at maximum conveyor speed? Any products closer together than this can be “locked out”. This is the “lockout distance”, which will be applied later on.

- ? What is the resultant maximum parts per second and parts per minute based on Lockout Distance and Max Speed?

D. Program the **INITIALIZE.INITIALIZE** worksheet with mechanical constants. Use calculations whenever possible.

1. G_STARTANGLE
2. G_KNIFECYCLE
3. G_BDC
4. G_SYNCANGLE
5. G_KNIFEDIAMETER
6. G_KNIFECIRCUMFERENCE*
7. G_MASTERCYCLE
8. G_SENSORDISTANCE
9. G_MAXCONVEYORSPEED
10. G_SYNCDISTANCE*
11. G_KNIFERETURNTIME*
12. G_LOCKOUTDISTANCE*

** These constants can be calculated*

EXAMPLE: First lines of the **INITIALIZE** POU

```

1  (* Mechanical measurements and properties of the machine *)
2  G_StartAngle:= LREAL#180.0; (* start angle of the rotary knife *)
3  G_KnifeCycle:= LREAL#360.0; (* knife machine cycle degrees/rotation*)
4  G_BDC:= LREAL#360.0; (* bottom dead center *)
5  G_KnifeDiameter:=LREAL#4.0; (* inches *)
6  G_KnifeCircumference:= LREAL#3.14159* G_KnifeDiameter; (* 4 inch diameter *)

```

E. Run the code and verify that the variables calculate as expected.

The above mechanical specifications will be used for further calculations throughout the application. If mechanical constants change due to machine design revision, only these variables will be updated for the code to run.

End Of Section

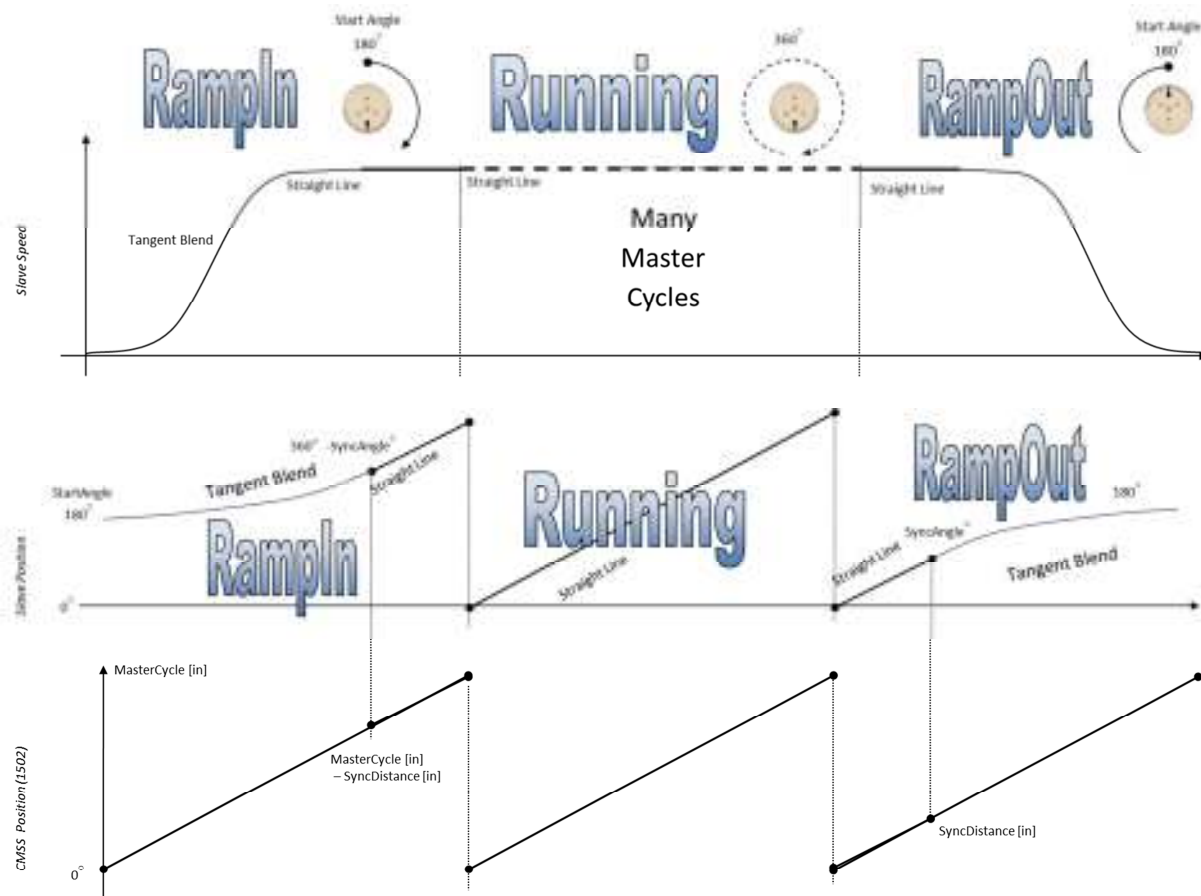
Please review your results with the instructor before continuing. The instructor will mark the section completed in the Certification Checklist.

Blend

V. Cam Blend Summary and Review

- *Pre-requisite eLearning Video on Cam Blend also covers this material.*
- A. **CamBlend** can be used in place of **Y_CamIn** and **Y_CamOut**
- B. **CamBlend** is a function block designed to smoothly engage and disengage a 1-way cam that is always in motion, such as the rotary knife in this application.
- C. **CamBlend** uses three cam tables
 1. RampIn
 2. Running
 3. RampOut

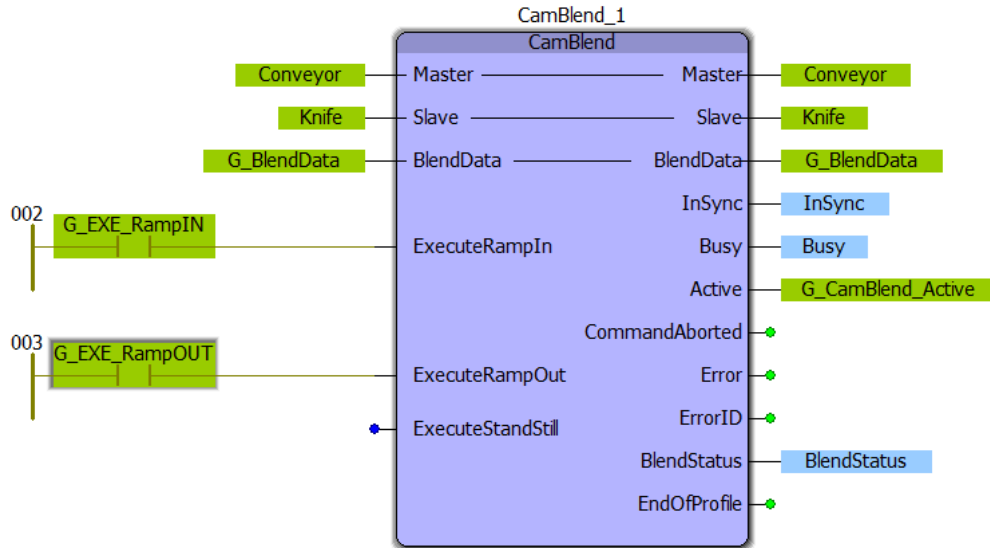
Breakout Lesson



- D. **CamGenerator** can be used to calculate the tables in the controller, based on specific application data.
- E. MotionWorks IEC Cam Editor can model the cam tables and produce ST code to initialize the data elements.

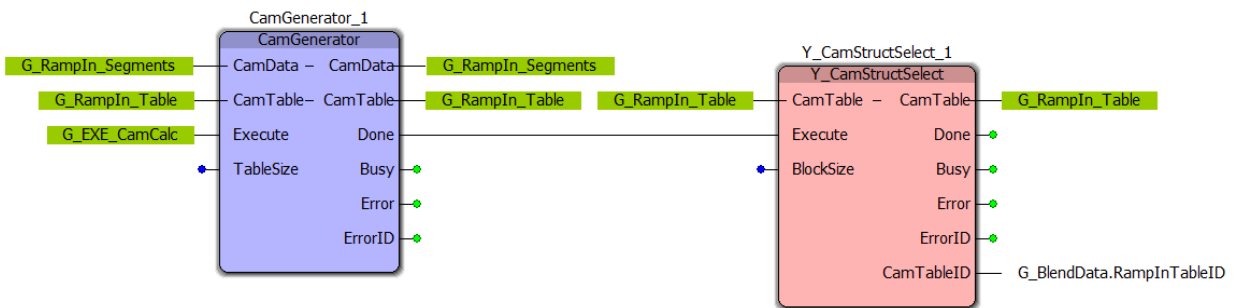
VI. Cam Blend Initial Programming

- Implement *CamGenerator*, *Y_CamStructSelect*, and *CamBlend* function blocks
 - A. Create a new LD Program POU called **BLEND** and run in FastTsk, then program as shown
 1. Insert the Cam Toolbox user library in order to use *CamBlend*



? Look at the elements of **G_BLENDDATA** (BlendStruct datatype). What are the names of the cam TableIDs required? These cam tables will be calculated in another POU.

- B. Create a new LD Program POU called **CAMCALC** and run in SlowTsk, then program *CamGenerator* and *Y_CamStructSelect* for each of the three required cam tables.
 1. **G_RAMPIN_SEGMENTS** and **_TABLE**
 2. **G_RUNNING_SEGMENTS** and **_TABLE**
 3. **G_RAMPOUT_SEGMENTS** and **_TABLE**



? Look at the elements of **G_RAMPIN_SEGMENTS**. What CamParameter structure elements appear in the array? These elements will be initialized by the ST code produced by Cam Editor.

VII. Yaskawa Cam Editor

- Use variables to generate the profile
- Cam Editor is taught in MPiec Intermediate Applications and in the pre-requisite eLearning videos


A. Launch Yaskawa Cam Editor

1. Save the Cam Editor project (File menu)
2. Turn on *Blend Mode* (Tools menu)
3. Use *Blended Profile view* (View menu)

B. Import variables

1. Copy all code from the **INITIALIZE.INITIALIZE** worksheet
2. Paste to *Structured Text Code* tab
3. Use *Convert ST code to cam segment table* (Tools menu)
4. *Variables* tab shows values

CamSegmentStruct: G_RampIn_Segments			
Cam Segment Table	Structured Text Code	Variables	Cam Table Data
	Variable	Value	
	G_StartAngle	180	
	G_BDC	360	
	G_KnifeCircumference	12.56636	
	G_MasterCycle	12.56636	
	G_SensorDistance	28.5	
	G_LockoutDistance	3	
	G_SyncAngle	10	
	G_SyncDistance	0.349065555555556	

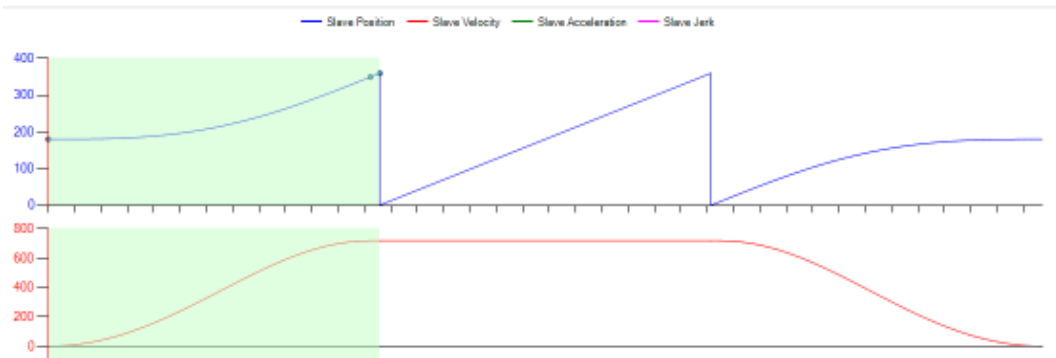
- C. Create Cam Segments with variables and expressions
 1. Refer to diagrams in the Breakout Lesson to create segments with variables and expressions
 2. Select the profile you will define using the  button
 3. Paste the CamSegmentStruct variable from the *CamGenerator* function block
 4. Example Cam Segments for RampIn

CamSegmentStruct: G_RampIn_Segments

Row	Master	Slave	Curve Type	Resolution
0	0	G_StartAngle		
1	G_MasterCycle - G_SyncDistance	G_BDC - G_SyncAngle	Tangent Blending	.1
2	G_MasterCycle	G_BDC	Straight Line	0
3				

NOTE: Set the Cycle Rate (Cycles /Min) to adjust the Resolution Warning

5. When correct, the graph looks very much the same as the illustration in the Breakout Session.



- D. Save the Yaskawa Cam Editor Project

VIII. Test CamBlend

A. Initialize *CamGenerator*

1. Create a new worksheet in the **INITIALIZATION** POU named **CAM**.
2. Generate ST code in Cam Editor
3. Paste ST code into this worksheet, **INITIALIZATION.CAM**
4. Use F5 to add variables to local list

B. Run the code

1. Confirm that each *CamGenerator* and *Y_CamStructSelect* executes without error.
2. Confirm that **G_BLENDDATA** receives the corresponding cam table IDs.

C. Manually blend in and out

1. Jog the conveyor
2. Use **RK IO.TB PANEL** or switches DI-4 and DI-5.
3. Try different speeds

? Look at **G_BLENDDATA.WINDOW** to the watch window. What is the value?

4. Run the conveyor at **MAX SPEED**. Set the window to a **LOW** value, such as 0.01 [in]. Execute RampIn and RampOut to see what happens. Also try a **HIGH** value for the window.
5. Run the conveyor at a **SLOW SPEED**. Leave the window at a **HIGH** value, such as 5.0 [in]. Execute RampIn and RampOut to see what happens
6. Read the help on *CamBlend* and follow the link to BlendStruct datatype
7. Read the help on *Y_CamIn* and follow the link to EngageWindow

? What is the required engage / disengage window as calculated from the MECHATROLINK interval and conveyor speed?

D. Test the calculated value for **G_BLENDDATA.WINDOW**

E. Initialize **G_BLENDDATA.WINDOW**

1. Create a new worksheet in the **INITIALIZATION** POU, named **BLEND**.
2. Write code to calculate and initialize the required value for **G_BLENDDATA.WINDOW**. The code shown calculates the distance over 2 MECHATROLINK intervals at max conveyor speed.
3. The MECHATROLINK interval can be derived as the inverse of the system variable **PLC_TICKS_PER_SEC**.

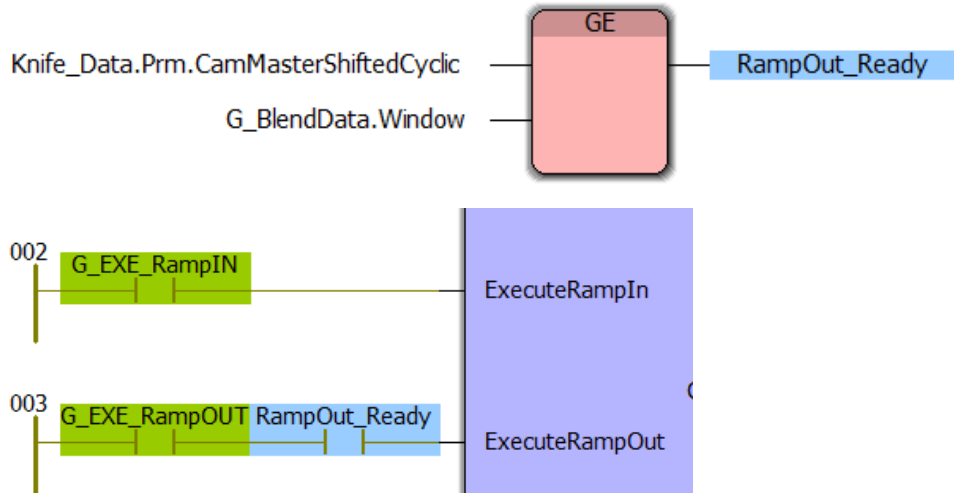
```

1  (* calculate the blend window based on speed and MECHATROLINK interval *)
2
3  (* System variable PLC_TICKS_PER_SEC is the inverse of MECHATROLINK INTERVAL *)
4  G_MECHATROLINK_Interval:= LREAL#1.0/INT_TO_LREAL( PLC_TICKS_PER_SEC);
5  (* Distance conveyor moves at max speed over 2 MECHATROLINK intervals *)
6  G_BlendData.Window:= LREAL#2.0* G_MECHATROLINK_Interval * G_MaxConveyorSpeed;|

```

F. Prevent early Ramp Out

1. It is possible for both RampIn and RampOut to execute within the same window. This results in an undesirable stopping position.
2. This can be observed at slower speeds by executing RampIn (DI-4) followed by immediately holding RampOut (DI-5).
3. Add the following code to the **BLEND** POU to prevent the early execution of RampOut within the same



- a. **KNIFE_DATA.PRM.CAMMASTERSHIFTEDCYCLIC** data is the position of the master within one cycle, and is updated within **RK_IO.MONITOR**.
 - b. This code assumes the default values of 0.0 for **G_BLENDDATA.RAMPINSWITCHOVERPOS** and **G_BLENDDATA.RAMPOUTSWITCHOVERPOS**
4. Test the code to confirm that early Ramp Out is not possible.