INTRODUCTION

FORMAT

The format of the Advanced Programming Workshop class is different from the typical Yaskawa training class. In this class you'll write an advanced rotary knife application, following this Application Resource Manual, and working at your own pace. The instructor is available to answer your questions about the material and to check to be sure you are working on track.

DEMO EQUIPMENT

The vast majority of the program can be tested on the desktop demo. At key points in the development of your code it will be necessary to test it out on the "real machine". The real machine is the Rotary Knife / Flying Shear demo. You are welcome to try your code on the real machine at any time. If you need to use the machine but it is in use by another student, please make this known to the instructor, but continue working ahead in this Application Resource Manual.

MATH AND EQUATIONS

This class does require that the student understand quite a few mathematical calculations and motion profile analysis. However, for the most part, the equations and profile analysis is explained so that the student can concentrate on incorporating these calculations in the program, rather than spending time developing them. So save yourself some time by reading ahead a few pages to find these calculations before solving them on your own.

TRAINING ACTIONS

Throughout this document are many headings titled "Training Action". These may consist of questions to answer, procedures to follow, or programming tasks to solve. This class is designed for you to "learn by doing" accompanied by simple, yet accurate descriptions to keep you on track. Please don't skip over the Training Actions. They serve as a checkpoint for the instructor to monitor your progress and to be sure you understand the material before getting too far ahead.

TRAINING ACTION

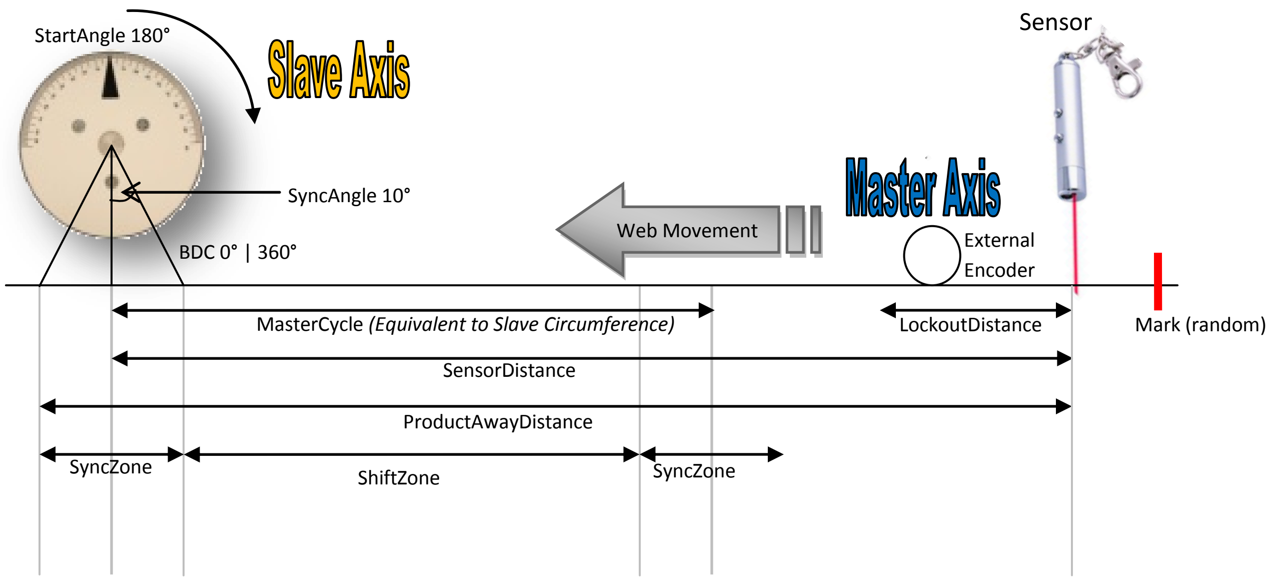
- Read the application description.
- Ask the instructor to clarify anything that is questionable or unclear.
- Think about how you would use the controller to solve the application

TABLE OF CONTENTS

CONTENTS

APPLICATION DESCRIPTION: ROTARY KNIFE

MACHINE EXAMPLES: Any machine that performs a process to a continuous "web" of moving material by means of a rotary actuator. This workshop will assume the process to be cutting, but the process could also be printing, forming, sealing, or others.

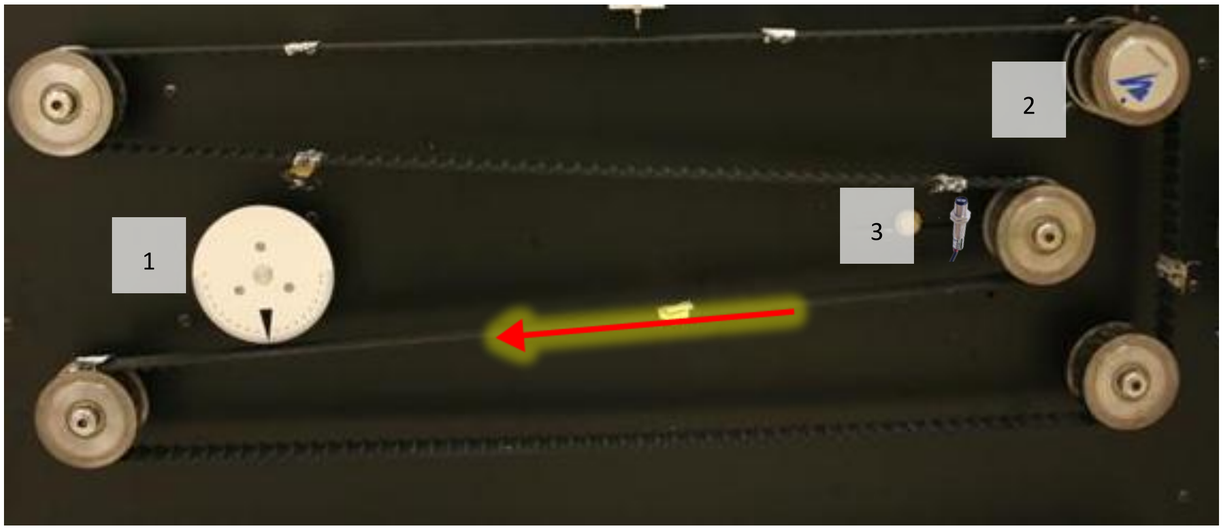


MACHINE DESCRIPTION AND OPERATION

- Material is moving continuously, in the diagram this is from right to left
- The material is printed with a mark and must be cut on the leading edge of this registration mark with a tolerance of [±0.001 Inches].
  - Even if there is material stretching or slippage, it will be cut at the correct location relative to the other graphical printing on the material.
- The knife will rotate in only one direction (CW in this diagram) to cut the material while moving.
- The rotary knife is driven by a rotating servo axis, as shown in the diagram, which is controlled by the MP2000iec.
- This knife is to penetrate a single point on the material as it moves, moving synchronously with the material throughout the entire cutting cycle, even in the unlikely case that the material were to speed up or slow down.
- The cutting is "random" in the sense that the registration marks may not be exactly the same distance apart. They are random to within 1.0 [inches]
- The final product, and therefore the product length must be able to change "on the fly"

MECHANICAL SPECIFICATIONS

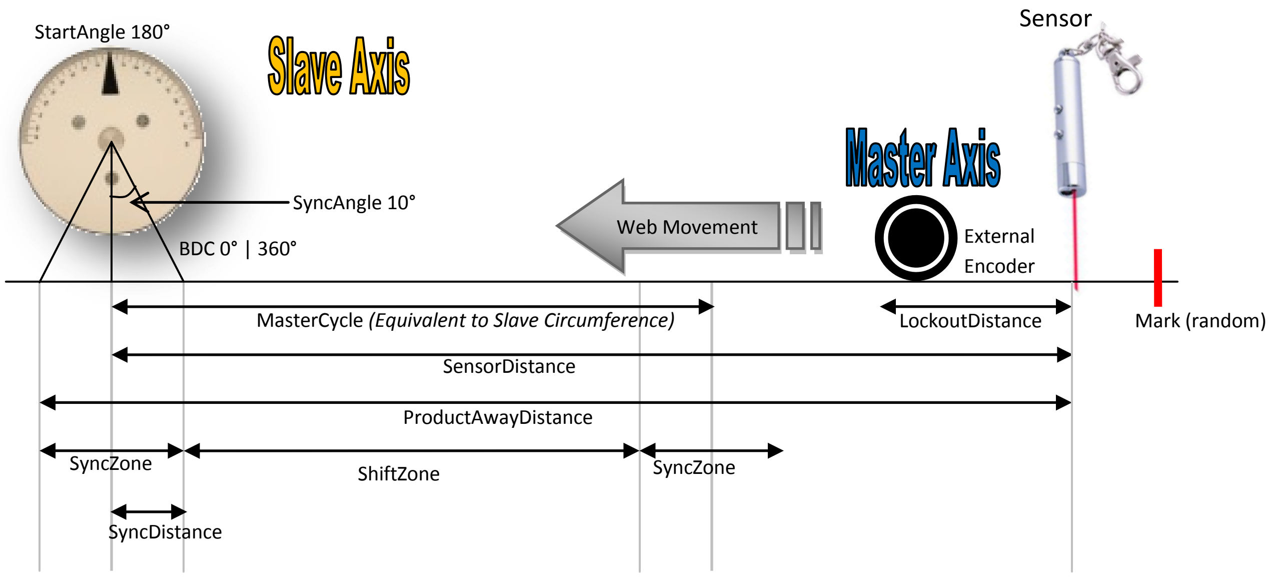
1. The servomotor that directly drives the rotary knife is a Sigma-2 motor (driven by a Sigma-5 amplifier) and has a resolution of 8192 [post-quadrature counts / rev]
    - Knife diameter is 4.000 [inches], giving a circumference of 4*pi = 12.566 [inches].
2. The controller measures the position of the material using an external encoder. On this demo that encoder is the master axis.
    - The external encoder on the web of material has a resolution of 8192 [post-quadrature counts / rev]
    - One revolution of the external encoder represents 8 [inches] of material movement
3. The sensor is located a relatively long way away from the knife cutting point. The distance from the registration sensor to the rotary knife bottom dead center is roughly 28.5 [in].



PERFORMANCE REQUIREMENTS

- Cut any length between 5[in] and 10[ft].
- The conveyor travels up to 400[ft/min] = 80[in/sec]
- The material feeds at a speed that may be adjusted by the operator. The material may suddenly slow or stop altogether due to an unexpected power outage or emergency shutdown. Therefore the knife must move at the exact speed of the material during the cutting cycle, or else the material will be wasted or the mechanism could jam.
- The knife must stop smoothly, pointing out of the material while waiting for product.

IMPORTANT MEASUREMENTS

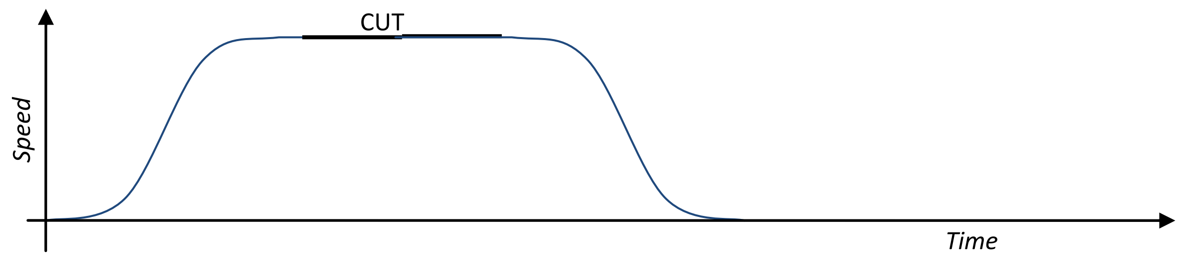


- **StartAngle**: Blade angle of 180 degrees, pointing up. The knife waits here until the sensor is latched. The knife will not stop here if another product has already been sensed.
- **BottomDeadCenter (BDC):** Blade position 360 degrees (= 0 degrees) pointing down. Full cut penetration at this point.
- **SyncAngle:** Blade angle in degrees, ±BDC during which the knife must remain synchronized with the material. Typical ±10 degrees. This SyncAngle can be adjusted based on material thickness.
- **SyncDistance:** Linear distance of the angle projected by the knife from BDC to first contact point of blade with material. Will be adjusted based on SyncAngle.
- **MasterCycle**: Linear distance the material moves per knife rotation, if the knife were to remain synchronized all the time. This distance is equal to the circumference of the knife.
- **ShiftZone, SyncZone**: The sync distance defines a zone within the master cycle in which the master can be shifted or must remain synchronized.
- **Sensor Distance:** Linear distance from registration sensor to knife BDC
- **ProductAwayDistance:** Linear distance from sensor to where the knife clears the product. At this distance the knife is free to adjust speed for the next cut mark.
- **LockoutDistance:** Filter to reject false latches after the current latched position.

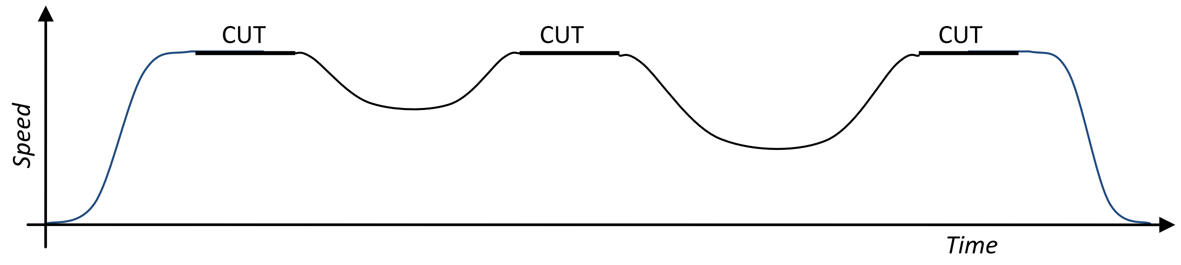
EXPECTED MOTION PROFILE

Before and after the cut, the rotary knife is expected to keep moving, slowing down or speeding up between cuts as needed. It will stop only if the sensor has detected no more marks. The most critical part of the knife motion occurs during the cut. The knife must follow the master during the cut. If the master is moving at a constant speed, then the knife will follow at this same constant speed.

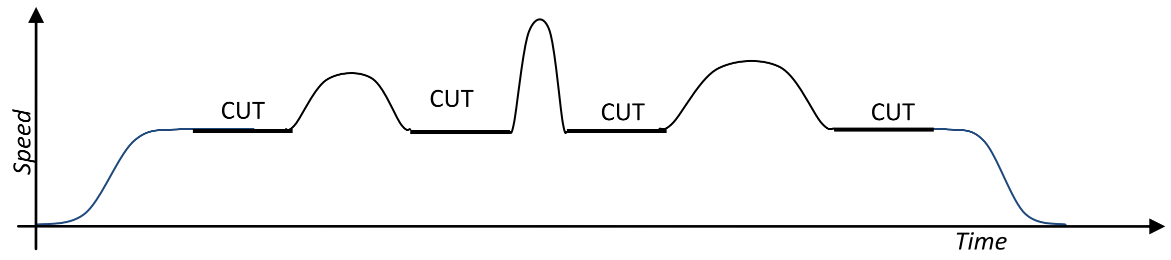
- **Case 1: Very distantly spaced. The knife stops at home position between cuts, waiting for another cut mark to be sensed.**



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



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



For a truly random mark, the knife must work for any combination of the above 3 cases. For example, the very first cut will by definition follow the acceleration of Case1. After all, the knife was stopped before power was turned on! Then as marks are sensed, it may alternate randomly between distantly and closely spaced cuts as the marks are sensed. Eventually no more marks are sensed, and the knife will come to rest again as in the end of Case1, waiting for the possibility of another mark.

SOLUTION APPROACHES

THREE INCORRECT APPROACHES

SET THE SPEED ONCE

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

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 customer has stated that the speed of the saw 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.

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.

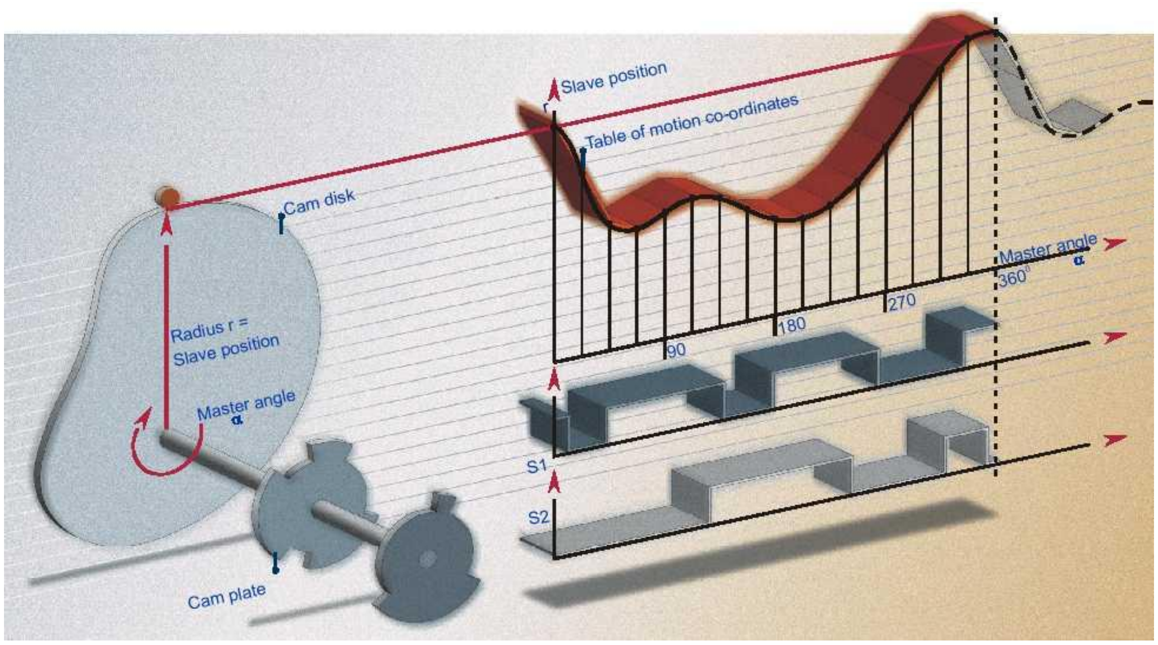
USE ELECTRONIC GEARING

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.

THE CORRECT APPROACH: ELECTRONIC CAM

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.

SOLUTION APPROACH

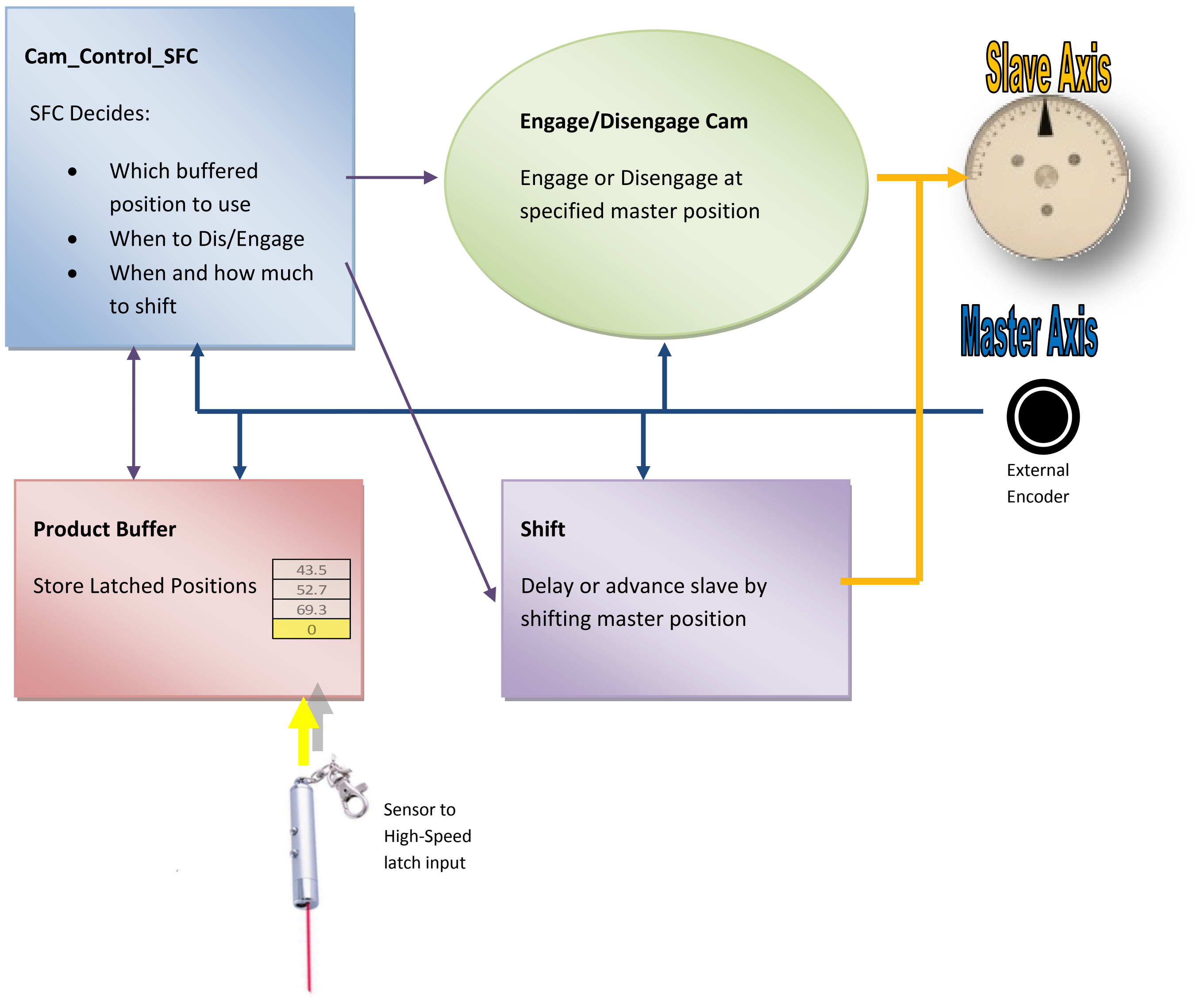
Use a 1:1 straight-line cam profile. Shift the cam when the master cycle is in the shift zone. Adjust the shift amount based on the distance between latched positions

Phase1: Automatically control cam engage and shift using SFC logic

Phase2: Calculate the cam table file internally using CamGenerator

Phase3: Use CamBlend for smooth engage and disengage

Phase4: Adjust and measure performance

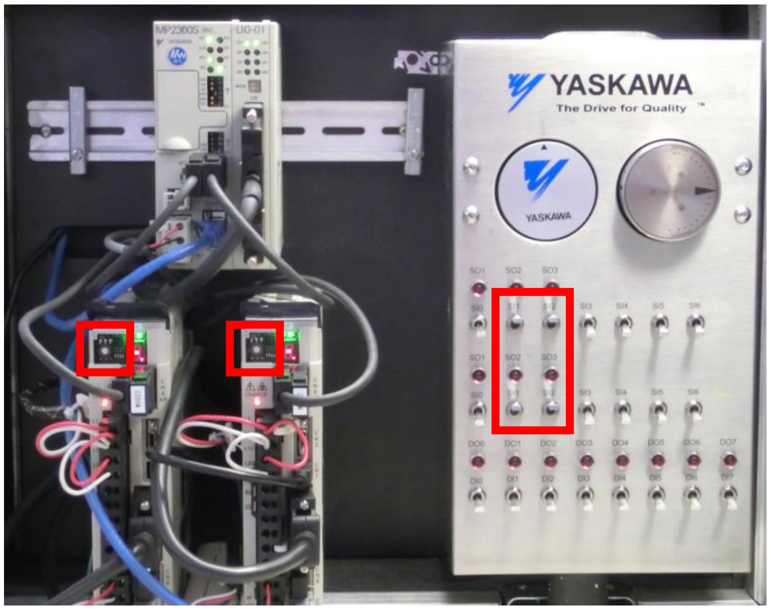

Cam_Control_SFC
SFC Decides:
• Which buffered position to use
• When to Dis/Engage
• When and how much to shift
Engage/Disengage Cam
Engage or Disengage at specified master position
Slave Axis
Master Axis
External Encoder
Product Buffer
Store Latched Positions
43.5
52.7
69.3
0
Shift
Delay or advance slave by shifting master position
Sensor to High-Speed latch input

STARTER PROJECT

SETUP

M-LINK Address

M-LINK Cables



P-OT, N-OT up

Training Action: Prepare Desktop Demo

- All inputs down except P-OT (SI1) and N-OT (SI2) up to simulate normally closed over-travel sensors.
- MECHATROLINK cables securely connected.
- MECHATROLINK address rotary switch set. Left = 1, Right = 2

TRAINING ACTION: STARTER PROGRAM OVERVIEW

It is important to send the archive (and not only the .mwt project) so that all data, including the configuration and the cam table are sent. Servopack Parameters and Absolute Encoder must also receive configuration.

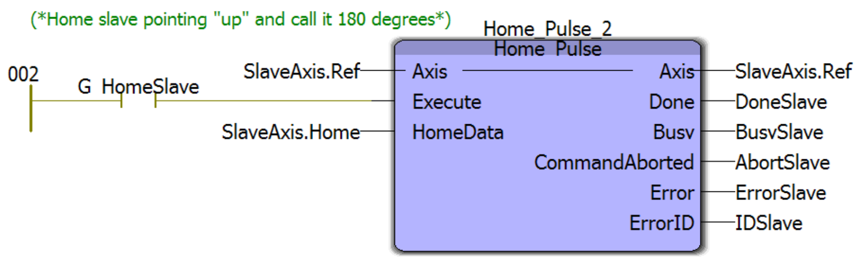
- In Web Server (Login as Admin/MP2300S – case sensitive)
  - Project Archive, Browse for "`Archive_APW1….zip`", Send to Controller
  - Drive Parameters – Write All User Pns
  - Reboot
  - Alarm A.CC0? (1)Machine operations (2)Drive Pn tab (3)Select Axis (4)Multi-turn Reset (5)Reboot
- In MotionWorks IEC – Pro
  - Open/Unzip the `APW1Starter.zwt` project file to the project folder and RENAME
  - Install the latest "Toolbox Installer" from Yaskawa.com
    - Benefit: Enables right-click help on the toolbox
    - Add the most recent version of CamToolbox and PLCopen Toolbox libraries, delete old
  - Open Hardware Configuration and confirm units. Update IP address.
  - Make, Download, and Cold-Start, then turn Debug mode ON
- Open the "Controls" POU to see how to use the different switches and to become familiar with the different POUs of the project.
  - Enable both axes (Switch SI3)
  - Jog the master (Switch DI2)
  - Adjust master speed (View – Watch Window, "Jog" tab)
    - 80.0 is the application maximum. 5.0 is good for testing.
- Open the Init POU and verify the mechanical constants, comparing to the application description.

QUESTIONS:

- The desktop demo has the Left and Right motor. Which motor is the cam master and which is the slave?
- Open the Hardware Configuration. How many inches does one revolution of the master axis represent on this desktop demo?
- How is this different than the physical inches per revolution of the master on the machine (Refer to Mechanical Specifications, p.4)?
- How many degrees does one revolution of the slave represent?

HOMING THE DEMO

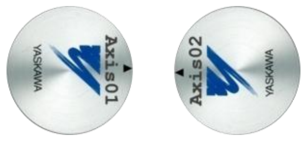
While homing is not the focus of this application, it is beneficial to home the axes to start at the correct position. The starter project takes advantage of Yaskawa's PLCopen Toolbox user library and the Home_Pulse function block to home to the encoder's "C-pulse" (sometimes called "marker pulse" or "index pulse"). The relationship to the c-pulse and the arrow on the disc varies from demo to demo, so a home offset must be set to calibrate the home position on the particular demo in front of you.



TRAINING ACTION: HOME THE DESKTOP DEMO

1. Open the Home POU (debug mode)
2. Open the Watch window, Home tab, set SlaveAxis.Home.Offset = 0.0
3. Execute the homing routine (DI1)
4. Disable both servos (SI 3) and move the motors by hand to the calibration **zero position**

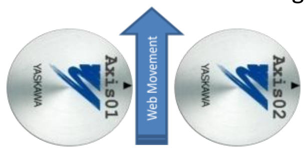
Calibration ZERO position



Home Calibration Angle **0°**

5. Set SlaveAxis.Home.Offset equal to the SlaveAxis.Prm.ActualPosition viewed in the watch window.
6. Enable both servos (SI 3) and home the axes again (DI1) to confirm. The slave axis will stop at the 180 deg position, pointing "away"

Position after successful homing



Start Angle **180°**

7. The demo uses absolute encoders, and so it never has to be homed again. The "real machine" uses incremental encoders, and so must be homed after every power cycle.
8. The SlaveAxis global variable is retained during power cycle. SlaveAxis.Home.Offset is initialized to a value of -89.0 (specific to Rotary Knife Demo) only at cold start in the "Cold" system task. Therefore do not do cold start again on the desktop demo or else you will have to calibrate the home position again.

ADDITIONAL INFORMATION ON PLCOPEN TOOLBOX "HOME_PULSE"

The Home_Pulse function block uses the "HomeData" input. This input uses the "HomeStruct" datatype included in the PLCopen Toolbox library. This structure itself is an element to a greater structure called "AxisStruct". You can see in the global variables that we have a variable SlaveAxis and MasterAxis each as an AxisStruct, which means it includes the HomeStruct. Add from the global variables list both SlaveAxis and MasterAxis to visually see the elements of these data structures. The HomeStruct element is already in the watch window and appears as MasterAxis.Home and SlaveAxis.Home. Expand it to see the elements of the structure. The Init POU "home" worksheet loads default values into the structure.

CAM ENGAGE/DISENGAGE

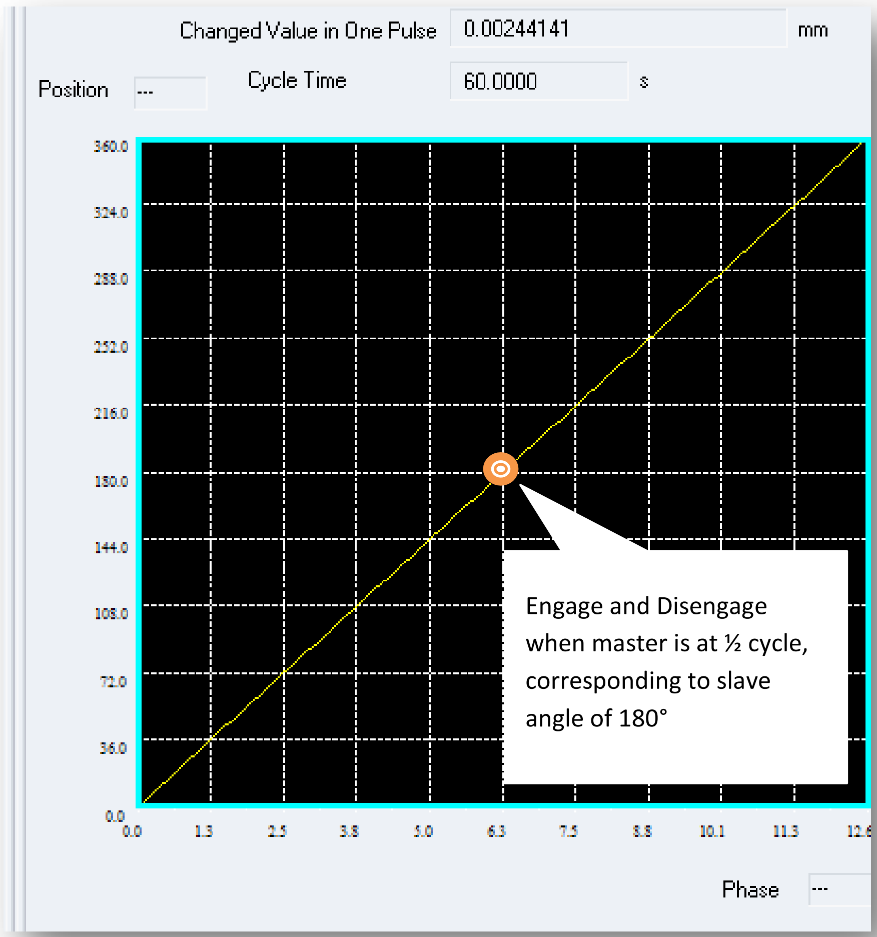
It is critical to understand how the controller engages and disengages the cam. Please do the following on the demo for hands-on experience.

TRAINING ACTION

- Open the CamInOut POU
- Jog the master slowly with DI2 (or disable the master and move by hand)
- Engage and disengage the cam with DI3 which executes the Y_CamIn block
- Open the watch window to see the master and slave position as you engage and disengage
    - Roughly, what is the change in master position over one cam cycle?
    - At what slave position does the slave engage and disengage?

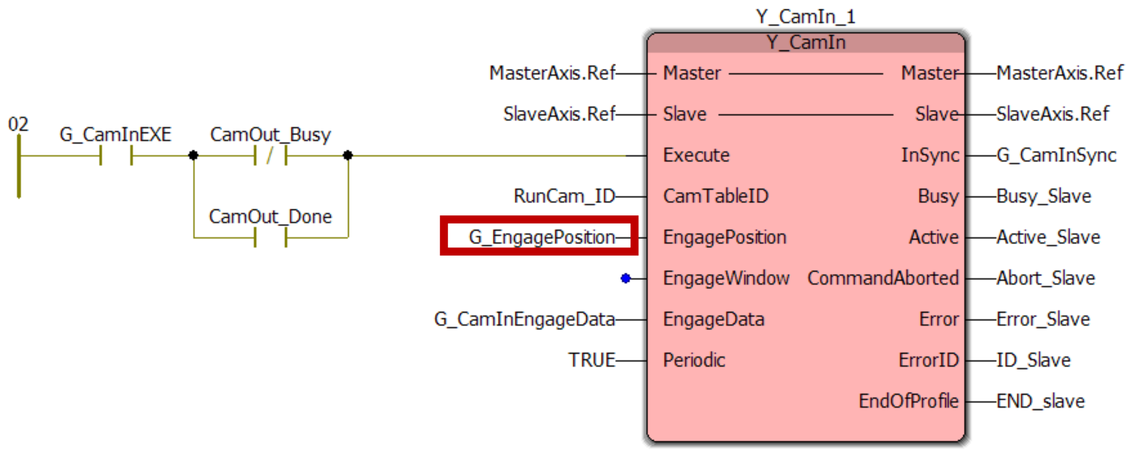
RUNNING CAM PROFILE

When we engage the cam, the controller moves the motor according to a simple 1:1 linear cam. This is a cam profile that behaves in many ways like electronic gearing. The cam table exists on the controller as a CSV file of master and slave positions (created by Yaskawa's Cam Tool software). The cam profile looks like this:

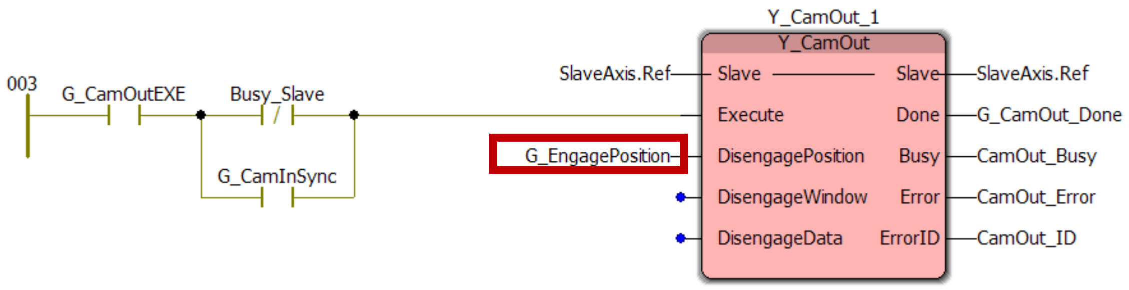


As the master goes through its cycle (zero to 12.566 [in]), the slave does one revolution (zero to 360). Once engaged, you can see the slave is essentially "gearing" to the master at a ratio of 1:1. We call this simple cam profile the "Running Cam".

In order for the slave to stop and start at the "up" angle of 180 [deg], the EngagePosition is set to ½ the master cycle. See the Init POU.



The same is done for the Y_CamOut instruction so that the slave stops in the "up" position.



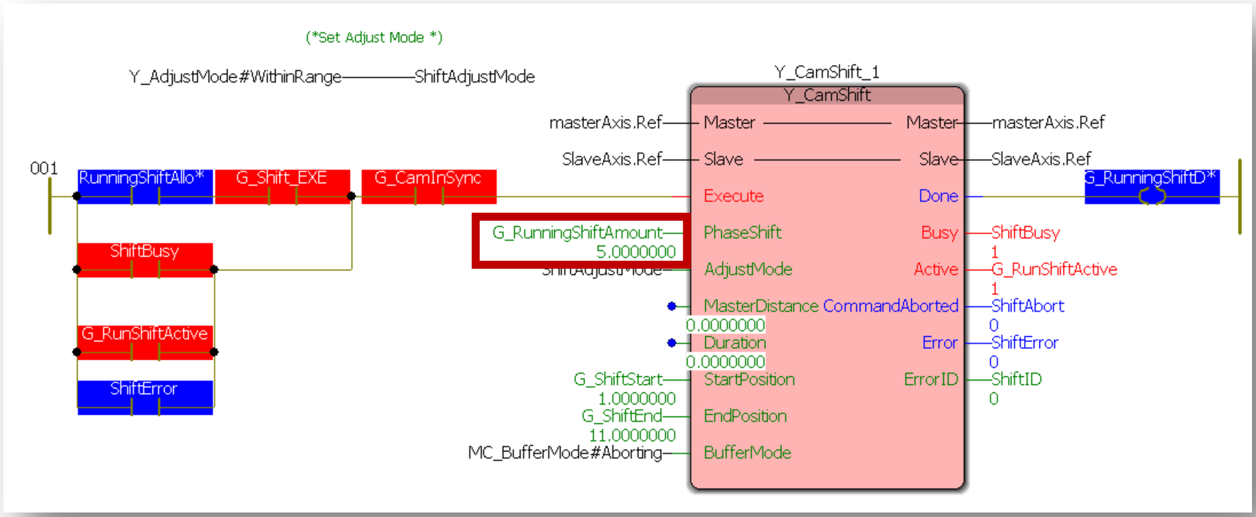
You'll notice that the slave quite abruptly starts and stops as it engages and disengages. This will be remedied in Phase 3 with CamBlend and two more profiles called "RampIn" and "RampOut".

SHIFT INTRODUCTION

It is also critical to understand the concept of shifting the cam master.

TRAINING ACTION: EXECUTE "RUNNING CAM SHIFT"

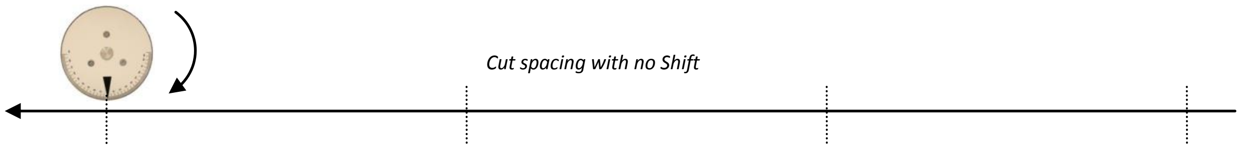
- Open "Shift_Running" POU
- Run the master slowly, engage the slave, and execute the shift (by toggling DI4)
- Adjust G_RunningShiftAmount
    - +9.0, +5.0, +1.0, 0.0, -1.0, -5.0, -15.0, -20.0, etc
- What is the maximum shift allowed?
- Is there a minimum shift? What happens with an extreme negative shift such as -40.0?



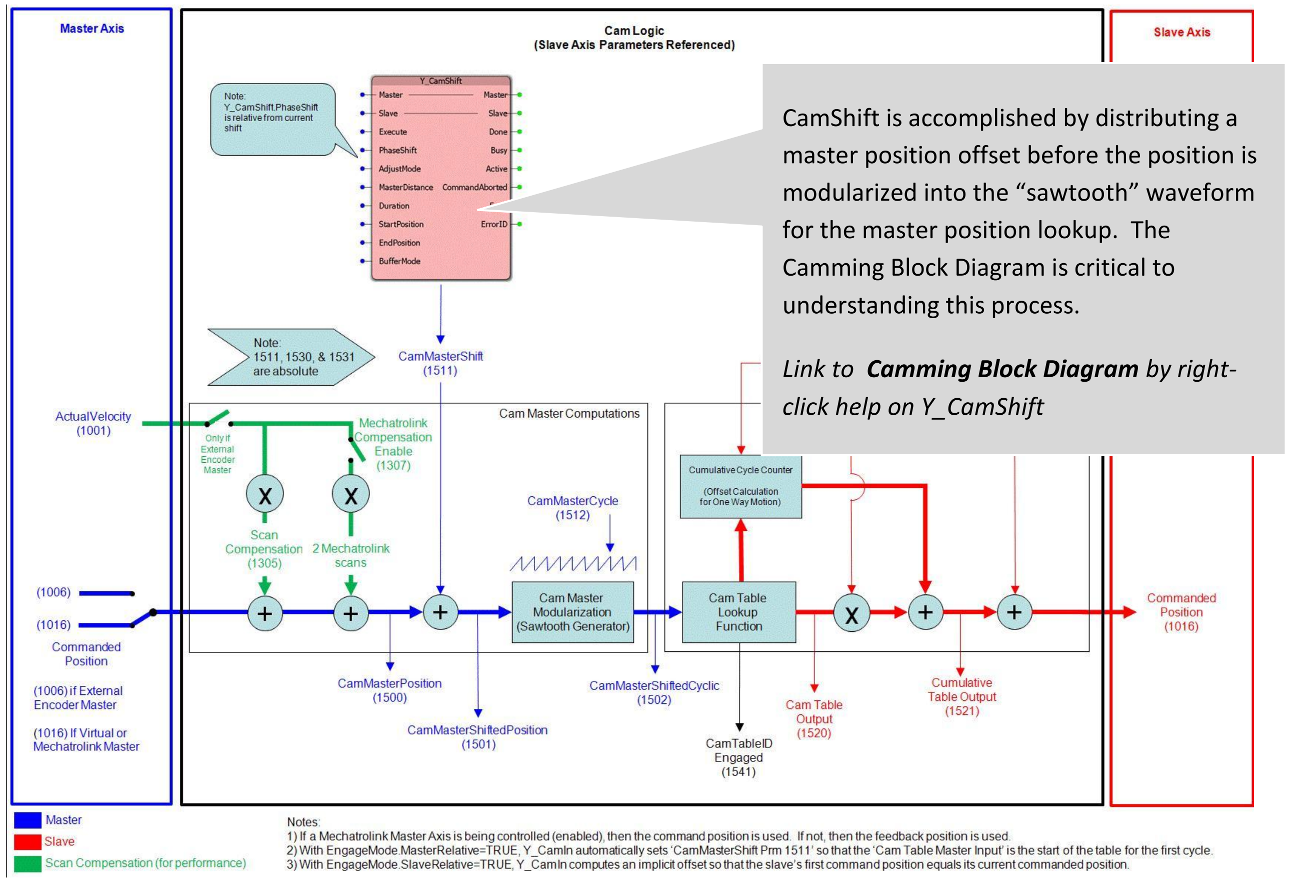
CAM SHIFT CONCEPT OVERVIEW

The concept is to shift the master during the running cam profile, either forward or reverse so that it speeds up or slows down, and returns to the normal cam profile at the required master position so that the knife cuts on the mark.

The amount of shift required depends on how much shorter or longer the cut spacing is. Think of it this way. What if there was no shift at all and the knife was simply "gearing" to the linear cam profile? This is the current condition of the project.



Shifting the slave while the cam is running allows us to shrink or grow this cut spacing. The slave and the cam block don't know about this; they just see the master position increasing and respond according to the cam lookup table. But the raw master position is not changed by shift, and neither is the position of the moving product. Instead, an intermediate CamMasterShiftedPosition(1501) is created inside the controller for reference.


Master Axis
Cam Logic
(Slave Axis Parameters Referenced)
Slave Axis
Note:
Y_CamShift.PhaseShift is relative from current shift
Y_CamShift
Master
Slave
Execute
PhaseShift
AdjustMode
MasterDistance
Duration
StartPosition
EndPosition
BufferMode
Master
Slave
Done
Busy
Active
CommandAborted
ErrorID
Note:
1511, 1530, & 1531 are absolute
CamMasterShift (1511)
ActualVelocity (1001)
Only if External Encoder Master
Mechatrolink Compensation Enable (1307)
Cam Master Computations
Scan Compensation (1305)
2 Mechatrolink scans
CamMasterCycle (1512)
Cam Master Modularization (Sawtooth Generator)
Cumulative Cycle Counter
(Offset Calculation for One Way Motion)
Cam Table Lookup Function
(1006)
(1016)
Commanded Position
(1006) if External Encoder Master
(1016) If Virtual or Mechatrolink Master
CamMasterPosition (1500)
CamMasterShiftedPosition (1501)
CamMasterShiftedCyclic (1502)
CamTableID Engaged (1541)
Cam Table Output (1520)
Cumulative Table Output (1521)
Commanded Position (1016)
X
+

CamShift is accomplished by distributing a master position offset before the position is modularized into the “sawtooth” waveform for the master position lookup. The Camming Block Diagram is critical to understanding this process.

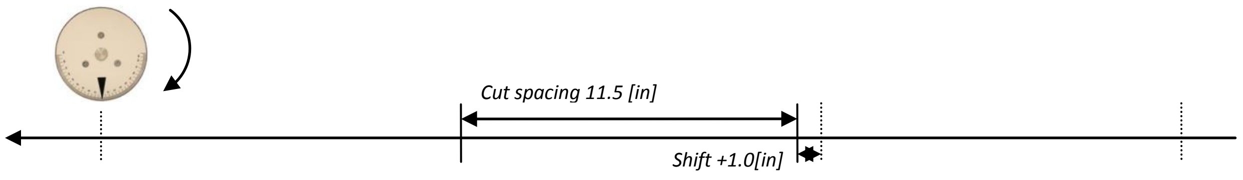
Link to ***Camming Block Diagram*** *by right-click help on Y_CamShift*

🟦 Master
🟥 Slave
🟩 Scan Compensation (for performance)

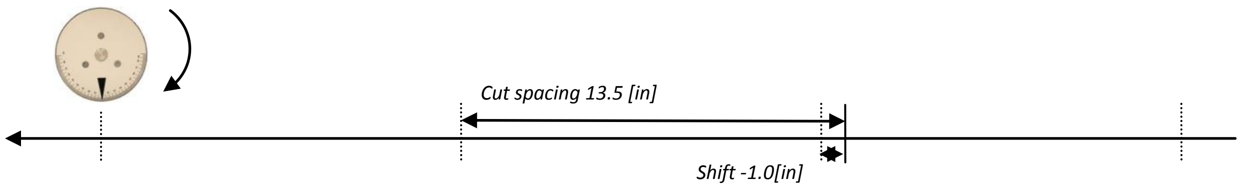
Notes:
1) If a Mechatrolink Master Axis is being controlled (enabled), then the command position is used. If not, then the feedback position is used.
2) With EngageMode.MasterRelative=TRUE, Y_CamIn automatically sets ‘CamMasterShift Prm 1511’ so that the ‘Cam Table Master Input’ is the start of the table for the first cycle.
3) With EngageMode.SlaveRelative=TRUE, Y_CamIn computes an implicit offset so that the slave’s first command position equals its current commanded position.

Consider the following example.

Shift the cam by +1.0 [in] (a forward shift) ➔Cut spacing will be 12.5-1.0 = 11.5[in].



If we shift the cam by -1.0[in] (a reverse shift), then the cut spacing will be 12.5-(-1.0) = 13.5 [in].



TRAINING ACTION

Observe the effect of positive and negative shifting

- What is the length of the part being cut without executing Y_CamShift ?
- What is the mathematical relationship between shift amount, part length, and master cycle?
- What is the shortest part length possible (with present values of Y_CamShift inputs)?
- What happens to the position of the master relative to the slave after a shift?
- Does the slave return to normal speed at the same position?
- Does the “cut point” relative to the master change with every shift?
- If you execute a positive shift “too late”, does it shift on the next cycle?
    - The Shift_ExecutionRange worksheet is calculating if there is still enough master cycle left to complete the shift. Otherwise Y_CamShift would generate error 4398.