Rotary Knife using the PTi-210

Objective
Develop an application solution to accurately cut/perforate/seal registered material using a rotary knife.

Overview
The PTi-210 Programming Module can be configured to perform many different types of rotary knife applications. This application involves moving a rotary drum containing a cutting blade to match speeds with a moving material and position the blade to cut on a registration mark. The PTi-210 accepts signals from a sensor tracking the material, a sensor on the follower to determine the position of the knife on start-up, and a master encoder riding on the incoming material. The essential features used in the PTi-210 for this application are Capture, Queue, Synchronized Indexes, Timed Indexes, and Compound Indexes. Each of these features will be discussed individually, and then used to create a fully functional rotary knife application. This will include actual program code along with a detailed explanation of the code to facilitate understanding of the program instructions. See Figure 1 below for an example of a rotary knife system.

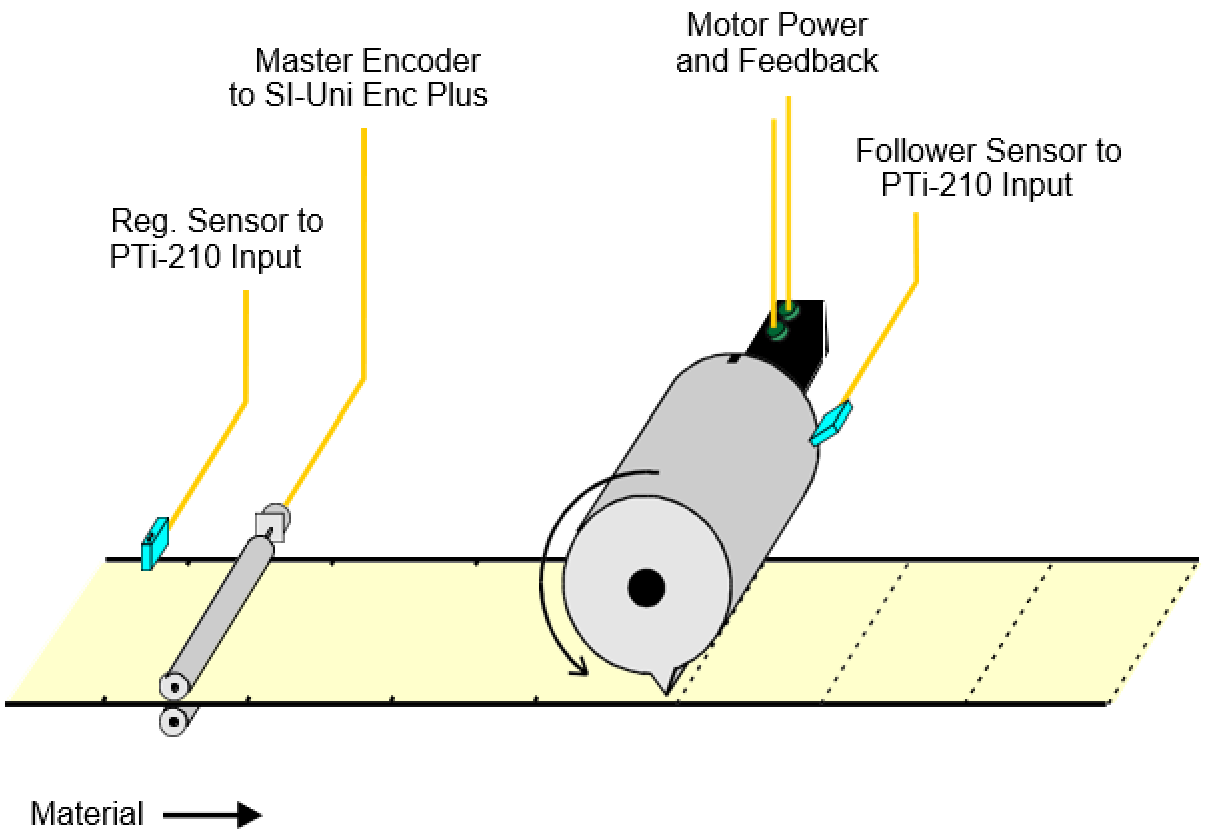


Figure 1 – Example Rotary Knife System

Capture Object

The Capture Object allows the PTi-210 to accurately capture the position of the master encoder at the exact time a registration mark passes the registration sensor. To do this, the registration sensor must be assigned to any one of the digital inputs located on the PTi-210 module (NOT to the M700 inputs or SI-I/O inputs).

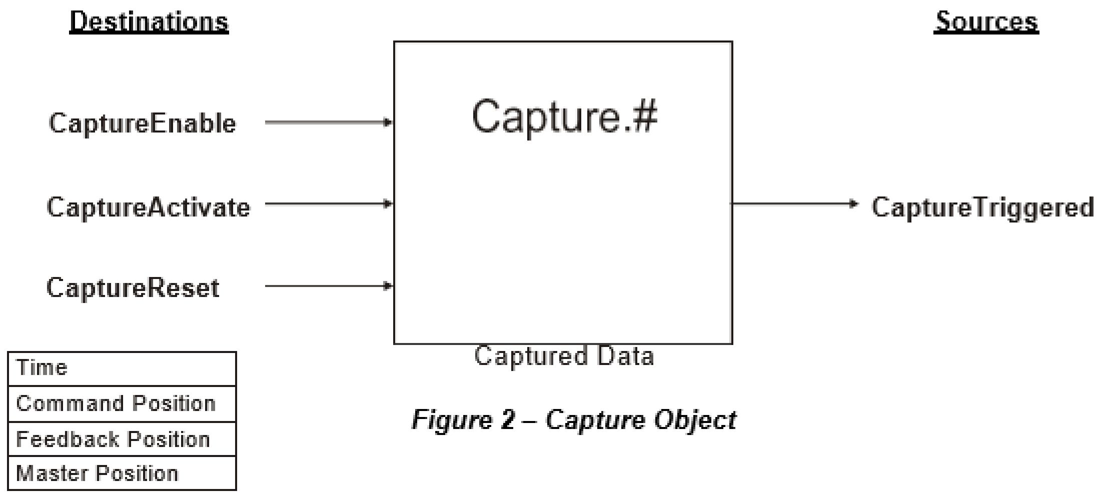


Figure 2 – Capture Object

The sources and destinations associated with the capture object can either be accessed through the Assignments screen or through a user program. This application will access the capture sources and destinations using both of these methods to operate the capture object.

In order to capture the position of the master encoder when a registration mark passes the sensor, the PTi-210 module digital input that the sensor is wired to must be assigned to the CaptureActivate destination found on the PowerTools Studio Assignments screen. Figure 9 shows all of the assignments used in the application. A User Program will deal with enabling and resetting the capture object.

When the capture is enabled, the first rising-edge of the CaptureActivate signal will engage the capture, and the Time, Motor Command Position, Motor Feedback Position, and the Master Position will be captured in less than 2

μsec. Once the capture has taken place, the CaptureTriggered function will turn on indicating that data has been captured and it is available for use in the application. CaptureReset must then be activated to prepare the system for the next capture, at which point CaptureTriggered signal turns off automatically.

Below is a timing diagram that details how the functions associated with the capture object operate.

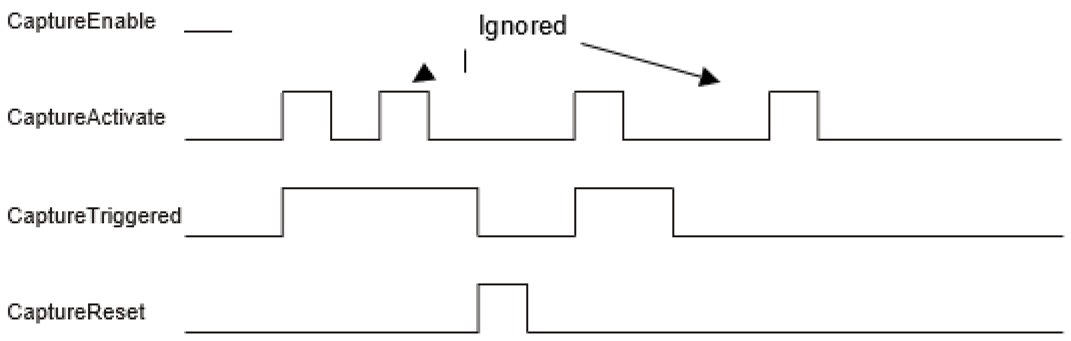


Figure 3 – Capture Timing

Queue Object

The Queue is used in applications where multiple products exist between the incoming product sensor and the location where the process takes place (i.e. rotary knife, applying labels, vision inspection, part rejection, etc.).

Up to eight Queue objects can be used simultaneously to control all of the processes in your application. Figure 4 below is a diagram of the Queue Object.

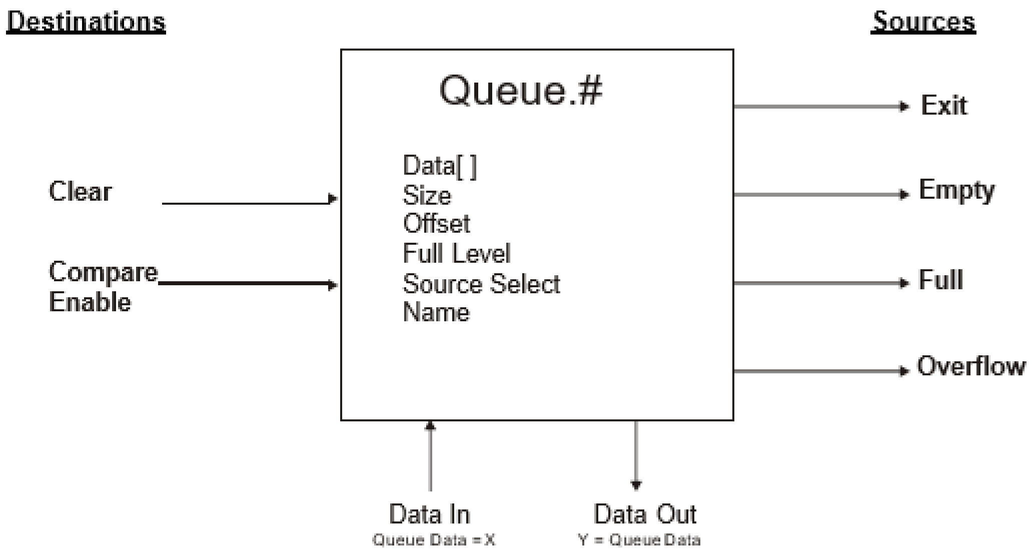


Figure 4 – Queue Object

The Queue is used to store the captured master position for each registration mark that passes the product sensor/registration sensor. Similar to the Capture Object, the Queue sources and destinations can be controlled through the assignments screen or in the User Program. For the case of this Rotary Knife example, the Queue is only used to store captured master positions and is not used in its full capacity. This example does not use the Offset, Compare, and Exit capabilities of the Queue.

Note: Because the rotary knife application uses captured data to initiate the index to make the cut, the material must not move relative to the master encoder once it has passed the incoming registration sensor. If the material does move with respect to the master encoder, the cut will be made at the wrong time (or wrong position).

The following parameters are used to configure the Queue object:

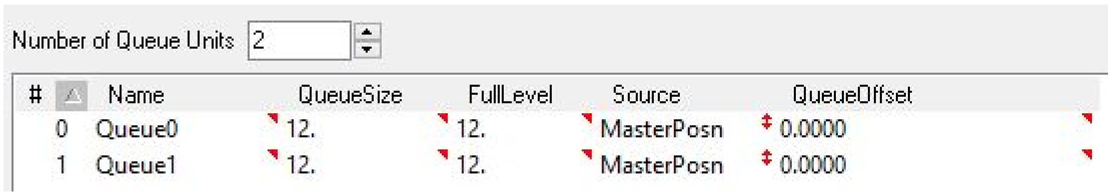
Name – Assign a name to each specific queue

Queue Size - This is the maximum number of registration marks/products that can be stored in the Queue at a time. If the number of products entered into the queue reaches this value, the QueueOverflow flag will activate. This value should be greater than the maximum number of registration marks/products that can possibly be between the sensor and the rotary knife cut point.

Queue Offset - The Queue Offset is added to the data put into the queue (captured position of master encoder in this example). The sum of the two values is called the QueueExitPosn. When the Source parameter is equal to the QueueExitPosn, the QueueExit function will activate. In the case of this example, the offset parameter is not used.

Full Level - This is a flag that notifies the user that a certain number of pieces of data are stored in the queue. Use this flag to notify the user that the queue is running at a certain "capacity".

Source - The Source determines which parameter to compare to the QueueExitPosn to activate the QueueExit function. If set to FeedbackPosn, the QueueExitPosn is compared to the Motor Position Feedback parameter. If set to MasterPosn , then QueueExitPosn is compared to the Master Feedback Position parameter, and if set to CommandPosn, then QueueExitPosn is compared to the Motor Commanded Position. When the Source and the QueueExitPosn are equal, the QueueExit function activates. The QueueExit feature is not used for this application.



Number of Queue Units 2

#	Name	QueueSize	FullLevel	Source	QueueOffset
0	Queue0	12.	12.	MasterPosn	0.0000
1	Queue1	12.	12.	MasterPosn	0.0000

Figure 5 – Example Queuing Setup Screen

Following is a detailed view of the internal workings of the queue object. It shows how each of the queue parameters is used and how each source and destination functions.

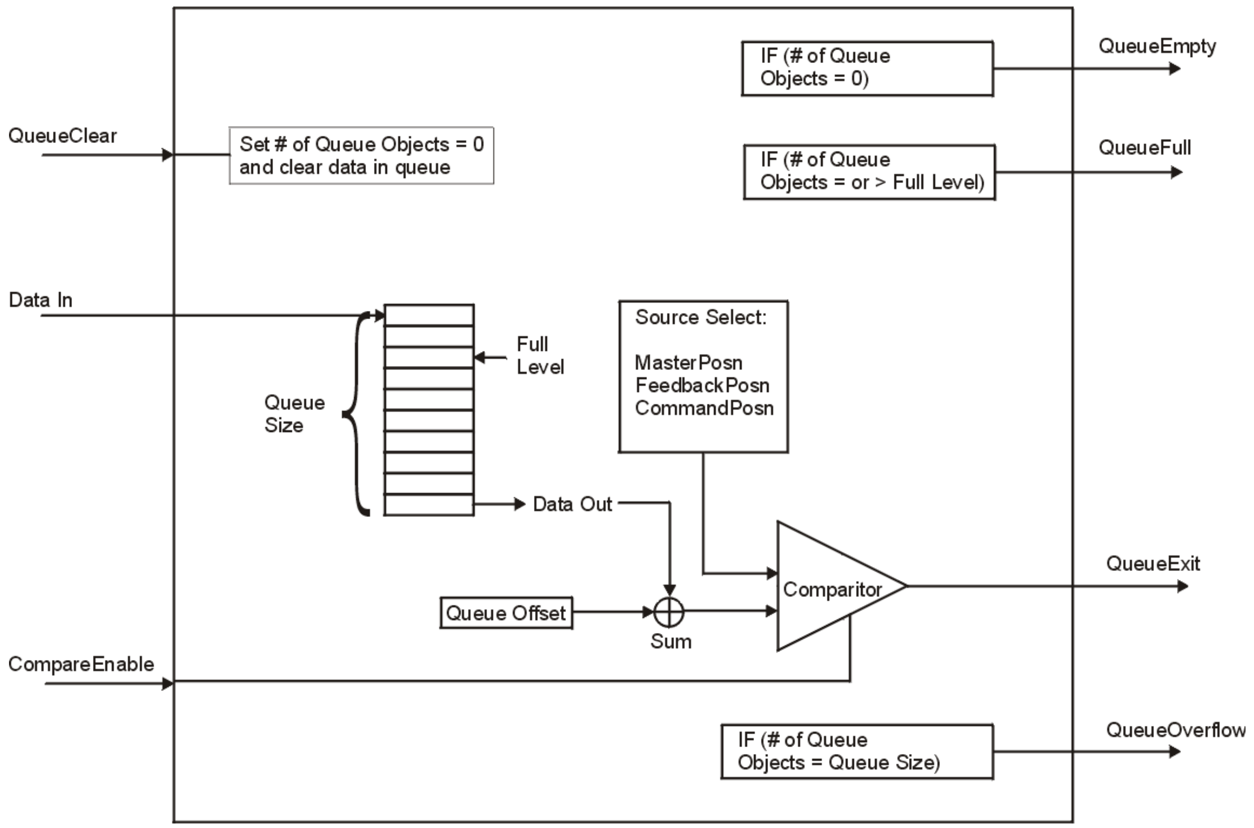
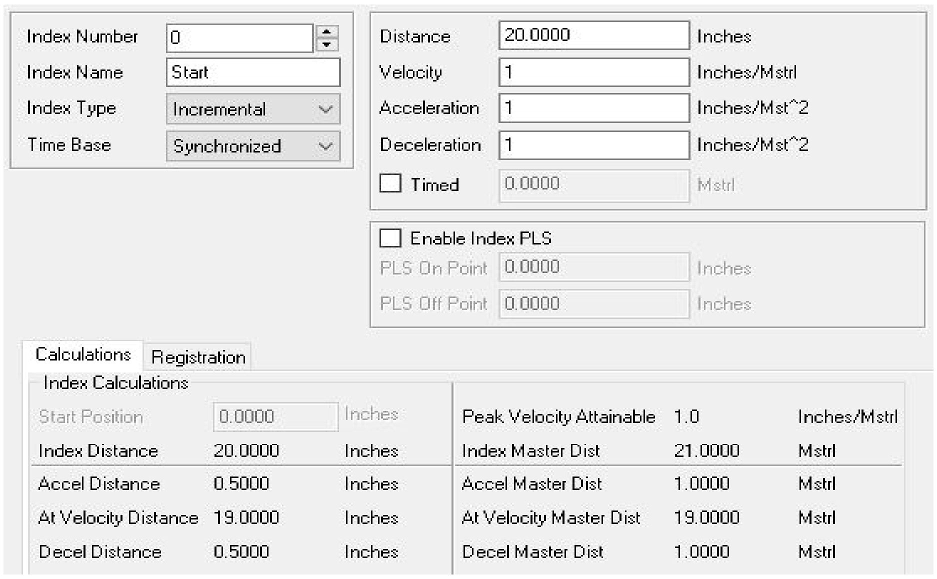


Figure 6 – Detailed View of Queue Object

Synchronized Indexes

Synchronized indexes are used to command the motor to move a specified distance at a velocity referenced to an external encoder. To configure an index as synchronized, the Timebase parameter must be changed to Synchronized as shown in Figure 7 below. When using synchronized indexes, the units for velocity, acceleration, and deceleration change to be units of Follower Distance Units / Master Distance Unit. If the distance units for the master and follower are the same (i.e. inches, mm, deg, revs, etc) then the velocity of a sync index is programmed as a ratio. A velocity of 1.0 would mean that the follower travels one of its units per one master distance unit traveled. If the master encoder is stopped, then the index does not advance.



Index Number	0
Index Name	Start
Index Type	Incremental
Time Base	Synchronized

Distance	20.0000	Inches
Velocity	1	Inches/Mstrl
Acceleration	1	Inches/Mst^2
Deceleration	1	Inches/Mst^2
☐ Timed	0.0000	Mstrl

☐ Enable Index PLS		
PLS On Point	0.0000	Inches
PLS Off Point	0.0000	Inches

Calculations | Registration

Index Calculations

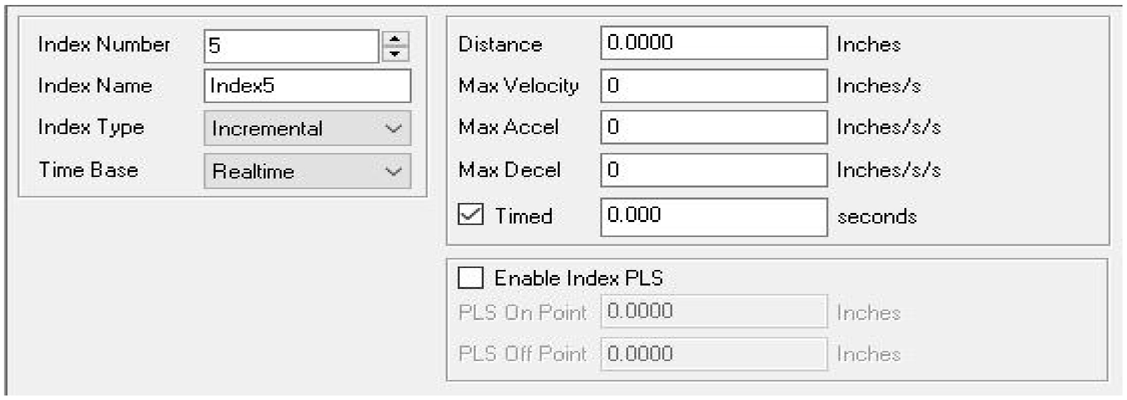
Start Position	0.0000	Inches	Peak Velocity Attainable	1.0	Inches/Mstrl
Index Distance	20.0000	Inches	Index Master Dist	21.0000	Mstrl
Accel Distance	0.5000	Inches	Accel Master Dist	1.0000	Mstrl
At Velocity Distance	19.0000	Inches	At Velocity Master Dist	19.0000	Mstrl
Decel Distance	0.5000	Inches	Decel Master Dist	1.0000	Mstrl

Figure 7 – Synchronized Index Setup

For more information on Synchronized Indexes, refer to App Tool #18 (Synch Jogs, Indexes, and Dwells).

Timed Indexes

Timed indexes allow the user to specify the amount of time in which an index must complete its specified distance (Units of Seconds, resolution of 1msec). Rather than forcing the user to try to calculate the velocity, accel, and decel, when they already know the desired time, the user can configure a Timed Index. To create a timed index, simply check the Timed Index checkbox on the index setup screen. Figure 8 below shows an index setup screen with the checkbox enabled.



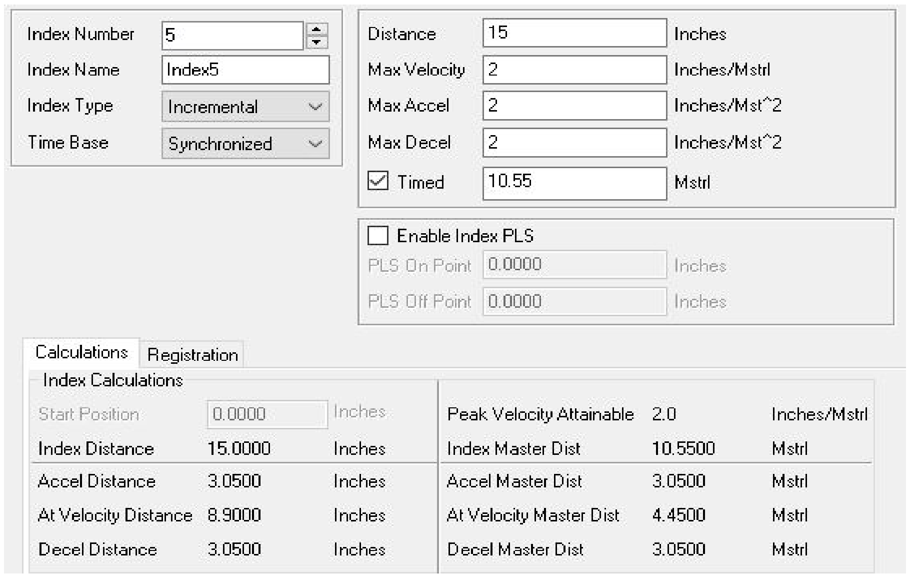
Index Number	5	
Index Name	Index5	
Index Type	Incremental	
Time Base	Realtime	

Distance	0.0000	Inches
Max Velocity	0	Inches/s
Max Accel	0	Inches/s/s
Max Decel	0	Inches/s/s
✓ Timed	0.000	seconds
☐ Enable Index PLS		
PLS On Point	0.0000	Inches
PLS Off Point	0.0000	Inches

Figure 8 – Timed Index Setup (Realtime)

When the Timed Index checkbox is enabled, the firmware does an internal calculation to determine what velocity, accel, and decel are necessary to complete the index distance in the time specified. Notice that when the Timed Index checkbox is enabled, the Velocity, Accel, and Decel are preceded by "Max.". This means that the user-entered parameters are used as maximums for the internal calculations. This is done so that the user can specify absolute limits to how fast the motor will move, and how aggressive the acc/dec ramps can be. If the user enters maximum values such that the index cannot complete in the specified time, a function called "Index.ProfileLimited" will activate. This signifies that the index will not complete in the entered time. To reset the function, the "Index.ResetProfileLimited" function is used.

If the Time Base for the Timed Index is set to Synchronized, the user enters the Index Time in units of Master Distance Units instead of Seconds (See Figure 9 below). By doing this, the user can specify the amount of master distance in which to complete the index. This will be a vital feature for the Rotary Knife application since the material being tracked by the master encoder will define how fast the rotary knife must move.



Index Number	5		Distance	15	Inches
Index Name	Index5		Max Velocity	2	Inches/Mstrl
Index Type	Incremental		Max Accel	2	Inches/Mst^2
Time Base	Synchronized		Max Decel	2	Inches/Mst^2
			☑ Timed	10.55	Mstrl

☐ Enable Index PLS

PLS On Point	0.0000	Inches
PLS Off Point	0.0000	Inches

Calculations | Registration

Index Calculations

Start Position	0.0000	Inches	Peak Velocity Attainable	2.0	Inches/Mstrl
Index Distance	15.0000	Inches	Index Master Dist	10.5500	Mstrl
Accel Distance	3.0500	Inches	Accel Master Dist	3.0500	Mstrl
At Velocity Distance	8.9000	Inches	At Velocity Master Dist	4.4500	Mstrl
Decel Distance	3.0500	Inches	Decel Master Dist	3.0500	Mstrl

Figure 9 – Timed Index Setup (Synchronized)

All Index Types other than Registration indexes can be configured as a timed index. Timed Indexes CAN NOT be compounded into our out of.

Compound Indexes

Compound Indexes allow the user to perform multiple indexes in a row without coming to a stop in-between each index. In order to compound indexes together, the indexes must be initiated in a program using the "Index.#.CompoundInitiate" instruction as opposed to the standard "Index.#.Initiate" instruction. An example of compounding Index 1 into Index 2 and then into Index 3 is shown below.

```
Index.1.CompoundInitia
te
Index.2.CompoundInitia
te Index.3.Initiate
```

Notice that the last index in the group uses only the standard "Index.#.Initiate" instruction. This signifies that Index 3 will not compound into another index, and that it will end at zero velocity.

When compounding from one index to another, the primary index will always end (complete its distance) at its programmed velocity. When it has covered its programmed distance, it will then ramp from its programmed velocity to the next index programmed velocity. Whether the motor needs to accelerate or decelerate to the next index velocity, the ramp used to reach that velocity is always the acceleration ramp of the secondary index.

Figure 10 below shows what the motion will look like for the example program instructions shown above. Table 1 contains the three index profile parameters that define how each index functions individually.

	Index 1	**Index 2**	**Index 3**
Dist (Revs)	10	10	15
Velocity (RPM)	1000	500	2000
Acceleration (RPM/sec)	10000	5000	10000
Deceleration (RPM/sec)	~~10000~~	~~5000~~	10000

Table 1 – Compound Index parameter values

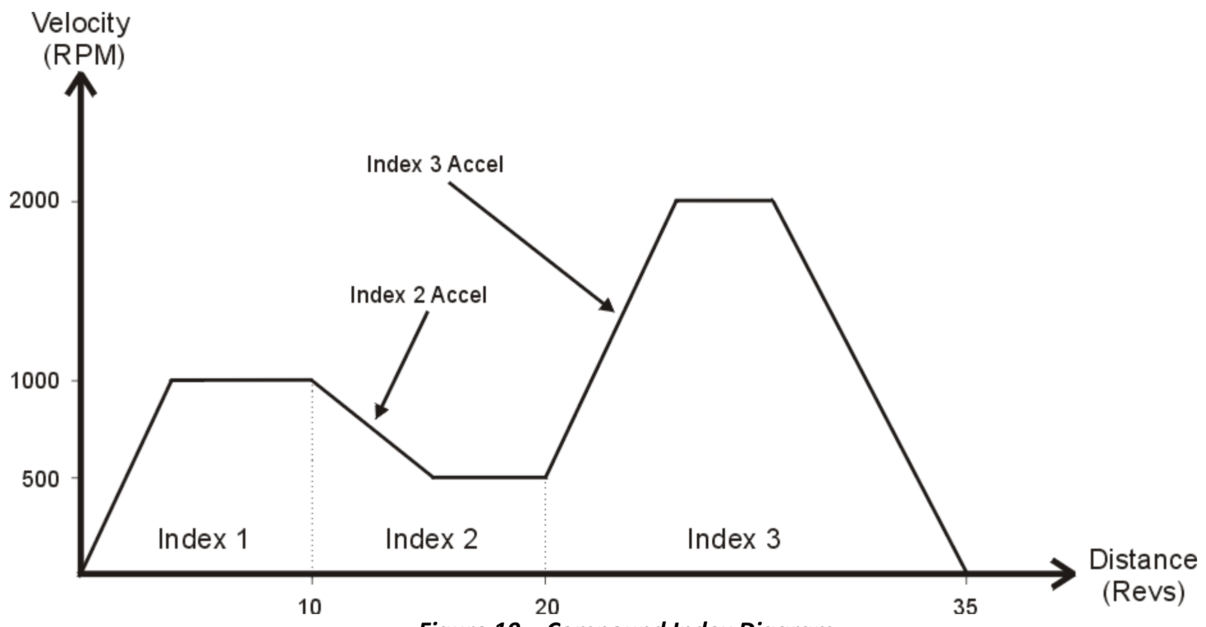


Figure 10 – Compound Index Diagram

Index 1 and Index 2 Decel values are crossed out in Table 1 above because when compounding into another index, the deceleration is never used.

The user programs for the rotary knife application rely heavily on compound indexes to maintain the positional relationship between the knife and the material that it is cutting.

Graphic with key dimensions and definitions

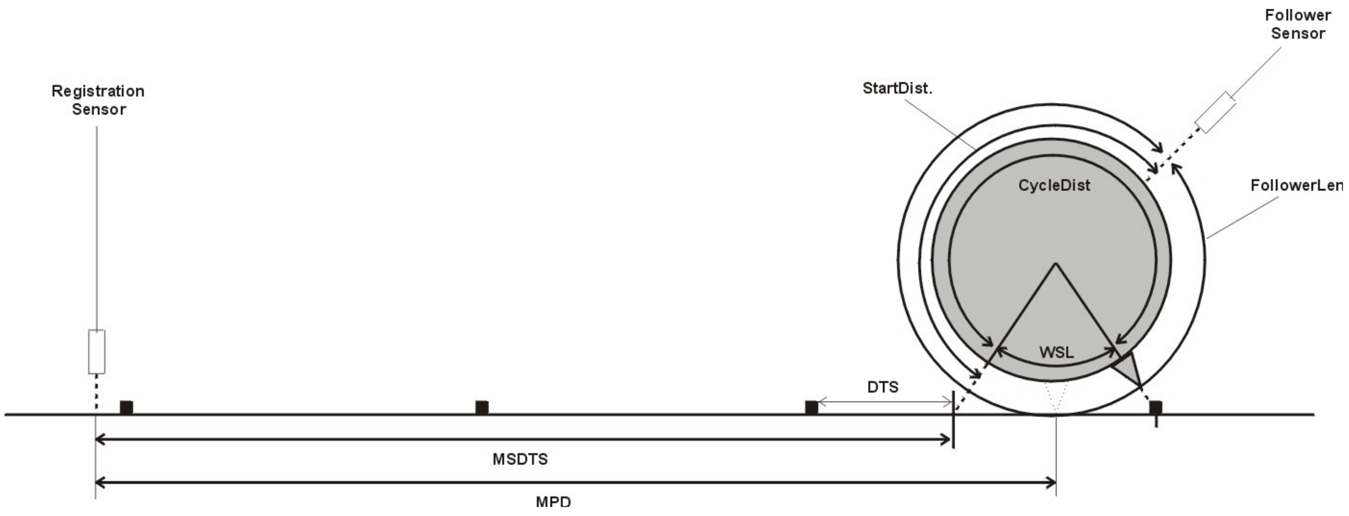


Figure 11 – Rotary Knife graphic with key dimensions

Definition of Terms

The following terms will be referred to throughout this application note. It is important to fully understand these terms.

MPD (Master Phase Distance) – The Master Phase Distance is the distance from the registration sensor to the center of the working segment. The units of the parameter are in master user units.

MSDTS (Master Sensor Distance to Synch) – MSDTS is the distance from the registration sensor to the beginning of the working segment. The units of the parameter are in master distance units. This parameter is automatically calculated in Program 0 based on other entered values.

DTS (Distance to Synch) – Distance to Synch is the distance from the next product/registration mark to the beginning of the working segment. It is called Distance to Synch because the follower must be synchronized to the master at the Working Ratio during the Working Segment. The units of the parameter are in master distance units.

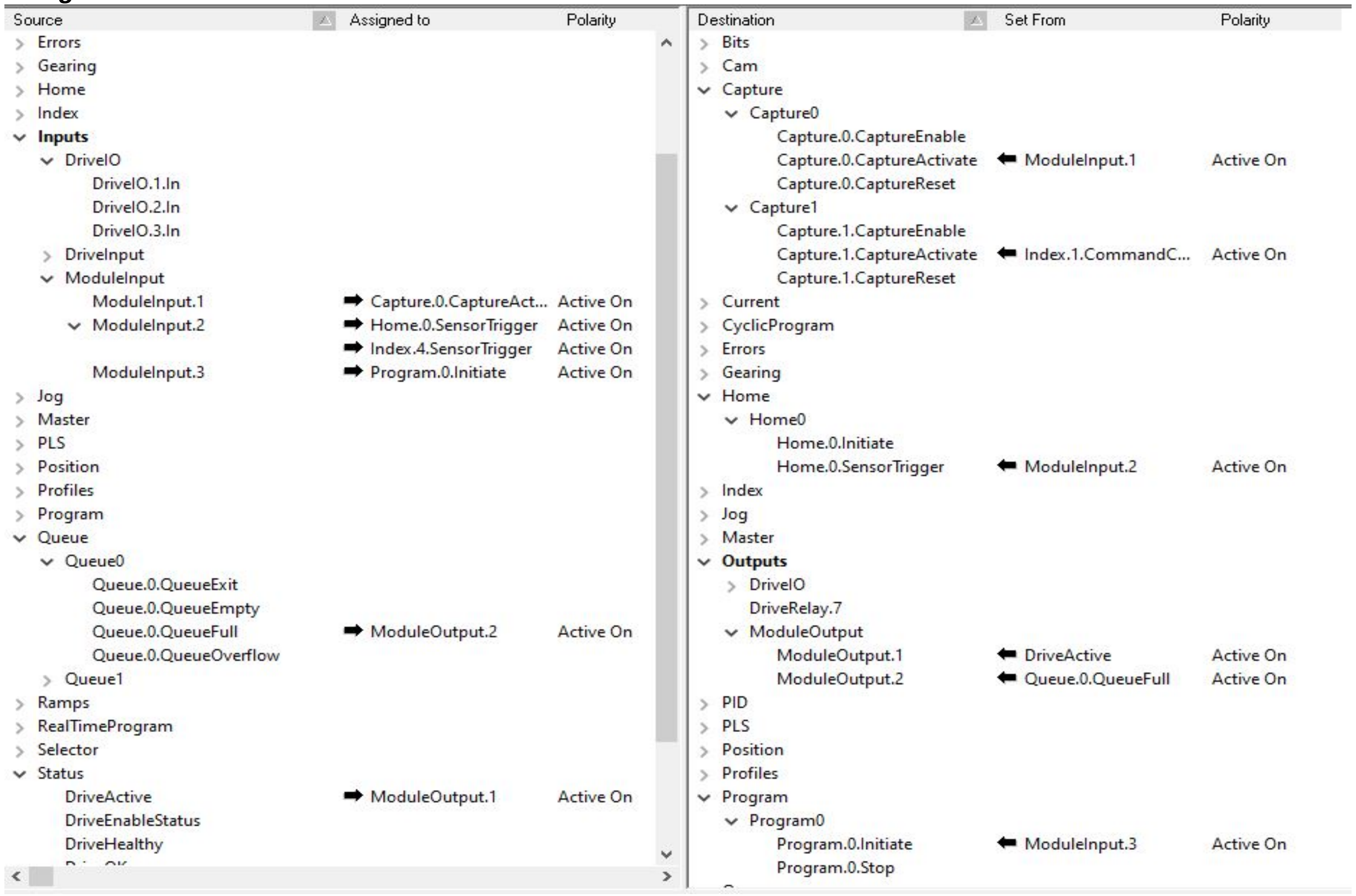
WSL (Working Segment Length) – The Working Segment Length is the follower distance that the follower will be synchronized to the master encoder at the Working Ratio. The Working Segment Length must be at least as long as the distance that the knife is in contact with the material. This avoids pulling on the material and tearing it or holding the material back and causing build-up behind the knife. The units for this parameter are follower distance units.

StartDist (Starting Distance) – The Starting Distance is the follower distance from the Follower Sensor (Home Sensor) to the beginning of the working segment. The units for this parameter are follower distance units.

FollowerLen (Follower Length) – The Follower Length is the follower distance covered by one full revolution of the rotary knife. The units for this parameter are follower distance units.

CycleDist (Cycle Distance) – The Cycle Distance is the follower distance in which the correction takes place such that the knife is in position for the next cut cycle. The follower will typically move faster than the working ratio in this region but can move slower depending on the space between registration marks.

Assignments Used



Source	Assigned to	Polarity
Errors		
Gearing		
Home		
Index		
Inputs		
DriveIO		
DriveIO.1.In		
DriveIO.2.In		
DriveIO.3.In		
DriveInput		
ModuleInput		
ModuleInput.1	Capture.0.CaptureAct...	Active On
ModuleInput.2	Home.0.SensorTrigger	Active On
	Index.4.SensorTrigger	Active On
ModuleInput.3	Program.0.Initiate	Active On
Jog		
Master		
PLS		
Position		
Profiles		
Program		
Queue		
Queue0		
Queue.0.QueueExit		
Queue.0.QueueEmpty		
Queue.0.QueueFull	ModuleOutput.2	Active On
Queue.0.QueueOverflow		
Queue1		
Ramps		
RealTimeProgram		
Selector		
Status		
DriveActive	ModuleOutput.1	Active On
DriveEnableStatus		
DriveHealthy		

Destination	Set From	Polarity
Bits		
Cam		
Capture		
Capture0		
Capture.0.CaptureEnable		
Capture.0.CaptureActivate	ModuleInput.1	Active On
Capture.0.CaptureReset		
Capture1		
Capture.1.CaptureEnable		
Capture.1.CaptureActivate	Index.1.CommandC...	Active On
Capture.1.CaptureReset		
Current		
CyclicProgram		
Errors		
Gearing		
Home		
Home0		
Home.0.Initiate		
Home.0.SensorTrigger	ModuleInput.2	Active On
Index		
Jog		
Master		
Outputs		
DriveIO		
DriveRelay.7		
ModuleOutput		
ModuleOutput.1	DriveActive	Active On
ModuleOutput.2	Queue.0.QueueFull	Active On
PID		
PLS		
Position		
Profiles		
Program		
Program0		
Program.0.Initiate	ModuleInput.3	Active On
Program.0.Stop		

Figure 12 – Assignments Screen

Figure 12 above shows the assignments used in this application.

ModuleInput.3 is used to start Program 0 which will start the entire system.

ModuleInput.1 is the input that the registration sensor is wired to. It is assigned to Capture.0.Activate so that when a mark passes the registration sensor, Capture object 0 will capture data.

ModuleInput.2 is the input that the knife sensor is wired to. It is used primarily for homing the system and for the cycle dropout routine (Index 4). If this input activates during a home or cycle dropout, then the knife moves to a specified offset from that position.

ModuleOutput.2 is controlled by the Queue.0.QueueFull source. If Queue.0.QueueFull activates, that would mean that too many marks have been sensed in the distance between the registration sensor and the cut point. This could signify that the material is defective, or that the registration sensor is malfunctioning.

Index.1.CommandComplete is assigned to Capture.1.CaptureActivate. This is done so that we can capture the position of the master encoder when the working segment is complete. The purpose of this will be described in the Program 1 description below.

Many other assignments could be made to add to the control capability or diagnostics of the system. The assignments listed in this application tool are a bare minimum to make the system function.

Motion Profiles

We will now take a look at the motion profile to used to solve the application. We can break the total application into three basic segments: Startup, Working Segment, Correction, and Stop (or Cycle Dropout).

The first segment is the startup segment. This is needed to get the system from a stop into the first cycle. To do this, a combination of two indexes is used to place the knife blade at the beginning of the working segment precisely when the mark reaches the beginning of the working segment.

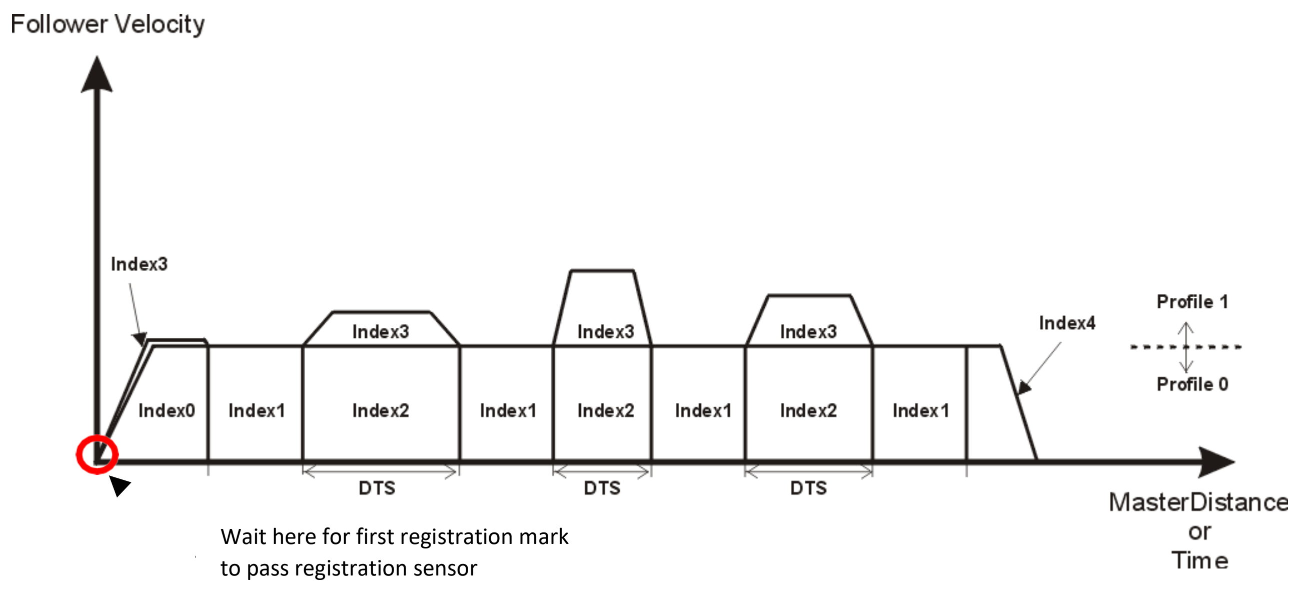


Figure 13 – Motion Profile (Waiting for Registration Mark)

The process starts after the system is homed, and the program waits until the next registration mark

Start Up Segment

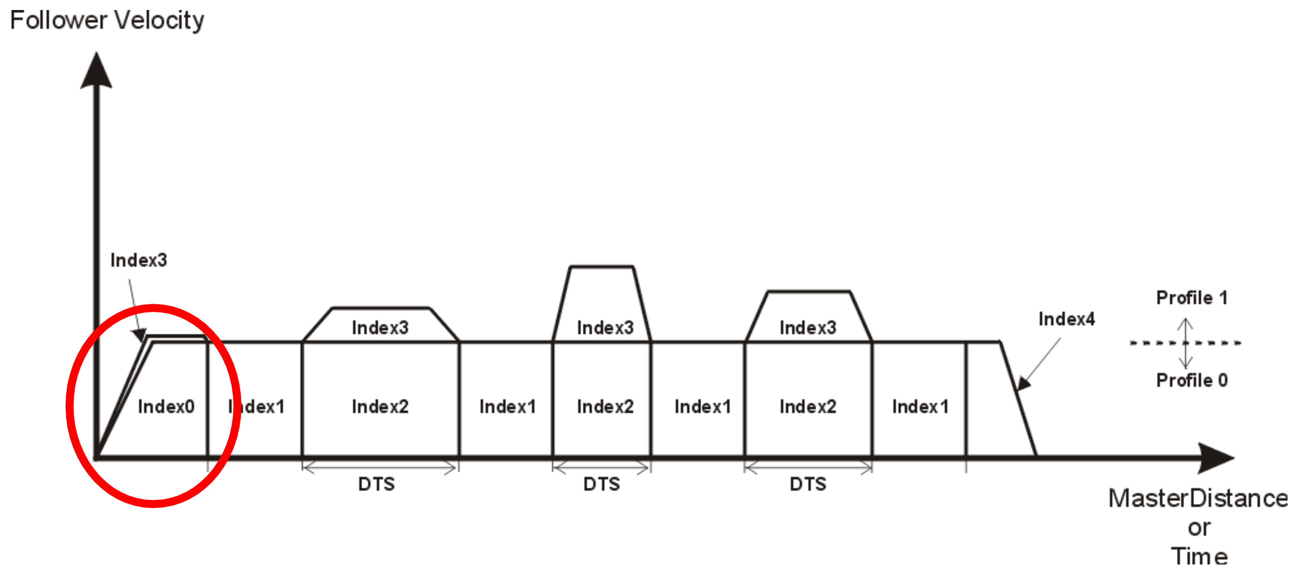


Figure 14 – Motion Profile (Start Up Segment)

Once the first registration mark passes the registration sensor, the program initiates Indexes 0 and 3 using the master position captured by the registration sensor as the starting point for the indexes. Index 0 is a synchronized index with a velocity of the working ratio. This index is compounded into Index 1 (the working segment index) so that when Index 0 is complete, it transitions into the working segment index (also with a velocity of the working ratio) without stopping or changing speeds. Since Index 0 runs at the working ratio (usually 1:1 with the master), if the distance from the registration sensor to the beginning of the working segment is not the same distance as the follower sensor to the beginning of the working segment, another profile must be added (or subtracted) to cause the knife to enter the working segment at exactly the right time. Our program uses Index 3 (the correction index) to achieve this. You will see later in the program description that Index 3 distance is the difference between distance from follower sensor to the beginning of the working segment and the distance from the registration sensor to the beginning of the working segment.

Working Segment

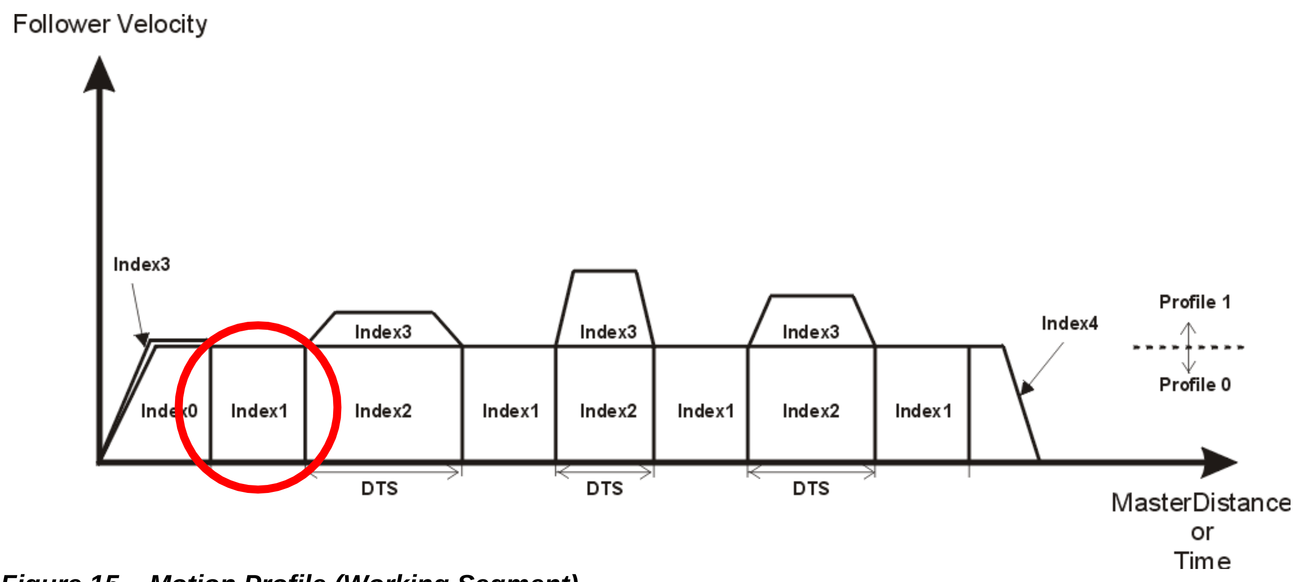


Figure 15 – Motion Profile (Working Segment)

When Indexes 0 and 3 complete, the knife will be at the beginning of the working segment. At this point, the knife will run at the working ratio for the working segment distance. This is the segment where the knife will actually perform the cut and be in contact with the material. Index 1 controls the working segment and is also a synchronized index to maintain the velocity referenced by the master encoder. The user at the beginning of Program 0 enters the working segment length into the variable Var.WSL. Index 1 compounds into Index 2 therefore, it will end at velocity and transfer directly into Index 2 without changing speeds.

Correction Segment

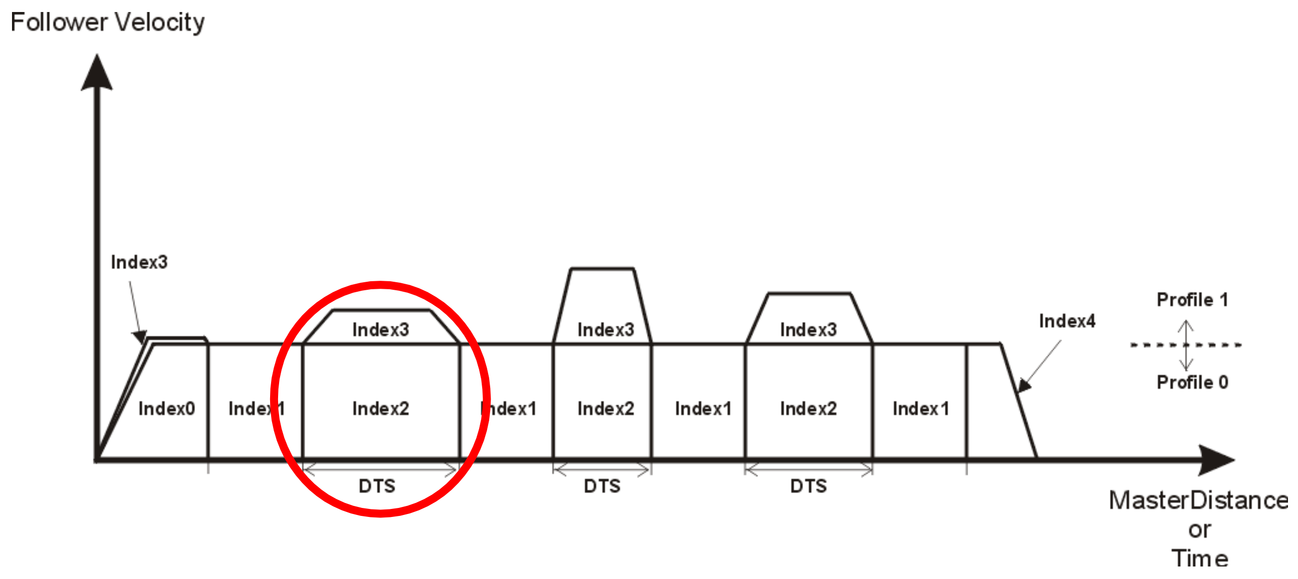


Figure 16 – Motion Profile (Correction Segment)

The next segment of the motion profile is the correction segment. The correction segment is made up of two indexes that run simultaneously on two different profiles; the cycle index, and the correction index. The cycle index, controlled by Index 2, is a synchronized index with a velocity of the working ratio. The working segment index (Index 1) compounds into Index 2 and therefore it starts At Velocity. The cycle index distance is calculated in Program 1 and is equal to Var.DTS. This means that its distance is the same as the distance from the next registration mark to the start of the working segment. Since the follower distance from the end of the working segment is most often different from the calculated value of DTS, some correction has to be done so that the knife gets into position by the time the next registration mark reaches the start of the working segment. The difference between DTS and the distance from the end of the working segment to the beginning of the working segment is loaded into the correction index (Index 3). Index 3 is a Synchronized – Timed index. The Time for Index 3 is set to DTS. Because it is a synchronized index, the time specified is actually a master distance. By setting the time to DTS, that means that the correction index will finish exactly when DTS master distance is passed and therefore it will finish exactly when the registration mark has reached the beginning of the working segment. Index 3 runs on Profile 1 so that it will run simultaneously with Index 2. The two index velocities and distances sum such that the total distance from the end of the working segment to the beginning of the working segment is covered in the proper time.

The key to this segment is that while the value of DTS changes, the correction segment will complete in different periods of time, but the total distance covered by Index 2 and Index 3 will always remain the same.

Cycle Dropout Segment

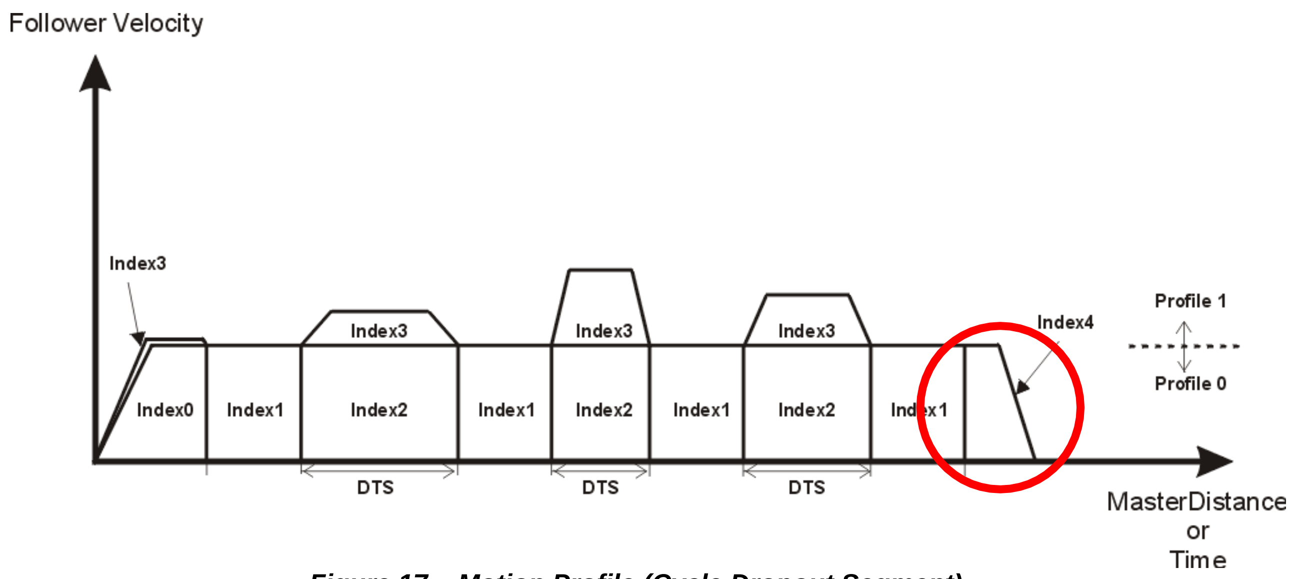
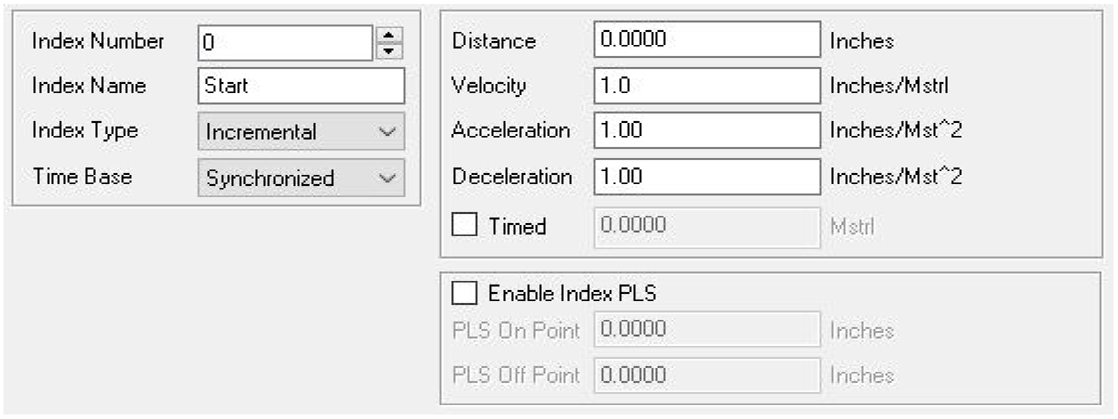


Figure 17 – Motion Profile (Cycle Dropout Segment)

This segment is used only when the user wishes to stop cycling or temporarily suspend cycling. A drive or module input is usually used to control when the system will drop out. This application tool handles this situation by using a registration index but could be done in many ways. Index 4 is a realtime registration index that moves until the knife registration sensor is seen. The registration sensor used for this index is the same sensor used for homing the knife. When the registration sensor is seen, the knife will come to a stop in the same position as when the homing sequence is completed. By moving to the same position as the homing sequence, we do not need to configure a special index for getting started again after performing a cycle dropout. We can simply use the same index that gets the sequence stated after a home routine.

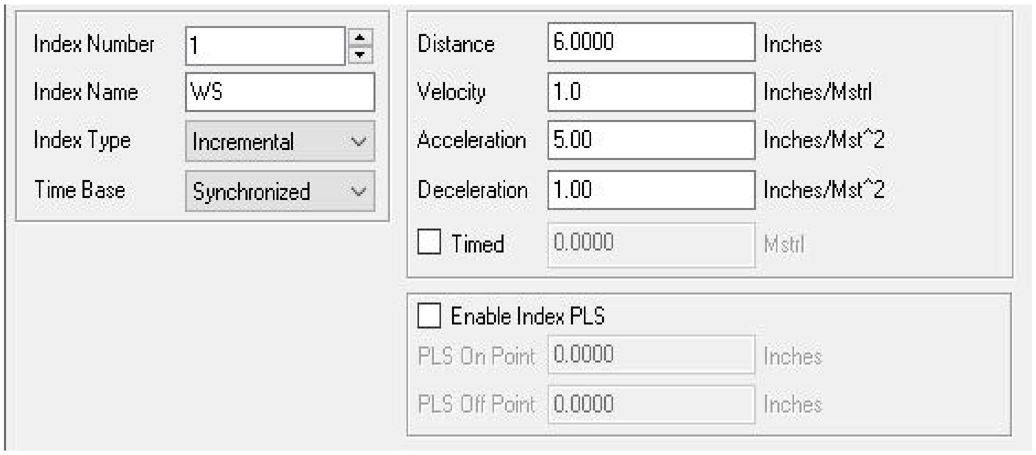
Index Parameter Values

Below are screen captures of each of the individual index setup screens. Parameters that show a value of 0.00 indicate that the user programs write to that parameter and therefore do not need to be setup on the index screen.



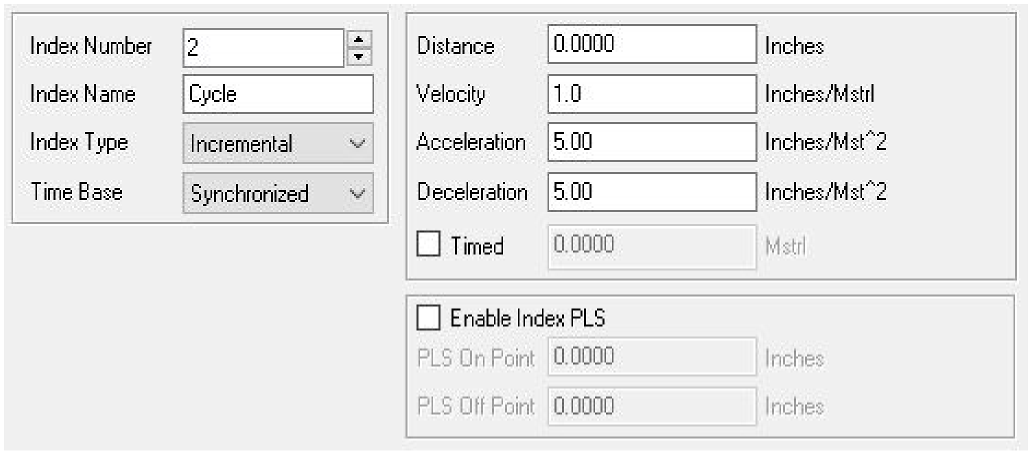
Index Number	0	Distance	0.0000	Inches
Index Name	Start	Velocity	1.0	Inches/Mstrl
Index Type	Incremental	Acceleration	1.00	Inches/Mst^2
Time Base	Synchronized	Deceleration	1.00	Inches/Mst^2
		☐ Timed	0.0000	Mstrl
		☐ Enable Index PLS		
		PLS On Point	0.0000	Inches
		PLS Off Point	0.0000	Inches

Figure 18 – Start Index Setup



Index Number	1	Distance	6.0000	Inches
Index Name	WS	Velocity	1.0	Inches/Mstrl
Index Type	Incremental	Acceleration	5.00	Inches/Mst^2
Time Base	Synchronized	Deceleration	1.00	Inches/Mst^2
		☐ Timed	0.0000	Mstrl
		☐ Enable Index PLS		
		PLS On Point	0.0000	Inches
		PLS Off Point	0.0000	Inches

Figure 19 – Working Segment Index Setup

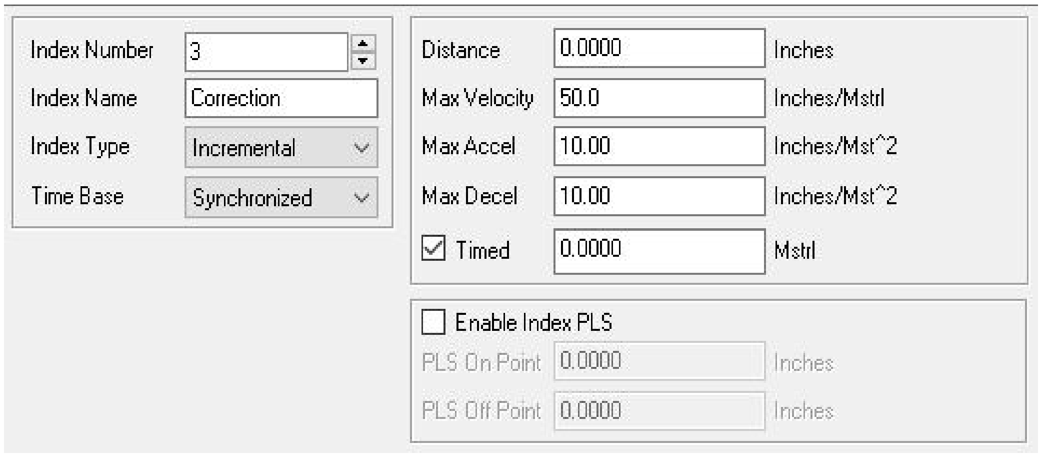


Index Number	2
Index Name	Cycle
Index Type	Incremental
Time Base	Synchronized

Distance	0.0000	Inches
Velocity	1.0	Inches/Mstrl
Acceleration	5.00	Inches/Mst^2
Deceleration	5.00	Inches/Mst^2
☐ Timed	0.0000	Mstrl

☐ Enable Index PLS		
PLS On Point	0.0000	Inches
PLS Off Point	0.0000	Inches

Figure 20 – Cycle Index Setup

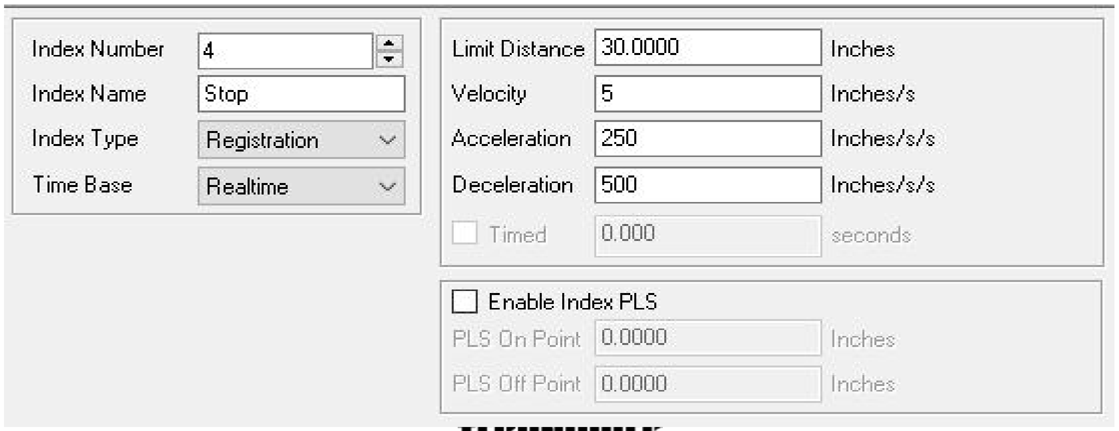


Index Number	3
Index Name	Correction
Index Type	Incremental
Time Base	Synchronized

Distance	0.0000	Inches
Max Velocity	50.0	Inches/Mstrl
Max Accel	10.00	Inches/Mst^2
Max Decel	10.00	Inches/Mst^2
☑ Timed	0.0000	Mstrl

☐ Enable Index PLS		
PLS On Point	0.0000	Inches
PLS Off Point	0.0000	Inches

Figure 21 – Correction Index Setup



Index Number	4
Index Name	Stop
Index Type	Registration
Time Base	Realtime

Limit Distance	30.0000	Inches
Velocity	5	Inches/s
Acceleration	250	Inches/s/s
Deceleration	500	Inches/s/s
☐ Timed	0.000	seconds

☐ Enable Index PLS		
PLS On Point	0.0000	Inches
PLS Off Point	0.0000	Inches

TECHNIQUES

```
Home.0.Initiate

Wait For Home.AnyCommandComplete



' User Entered Constants

Var.MPD =  'Enter distance from part sensor to center of working segment
here Var.WSL =     'Enter desired working segment length here

Var.WorkingRatio =  'Enter the desired working ratio here (usually
1.0) Var.FollowerLen =    'Enter complete follower cycle length here

Var.SensorDist =  'Enter distance from knife sensor to center of working segment here
Var.MinimumDTS = 'Enter the smallest DTS value in which the knife can make the
correction



' Initialize Calculations

Var.CycleDist = Var.FollowerLen - Var.WSL

Var.StartDist = Var.SensorDist - (Var.WSL/2.0000) -
Home.0.SpecifiedOffset Var.MSDTS = var.MPD - var.WSL/2.0000

Index.0.Vel                =
Var.WorkingRatio Index.1.Vel
=           Var.WorkingRatio
Index.2.Vel                =
Var.WorkingRatio
Index.1.Dist = Var.WSL



Queue.0.QueueClear = ON
Capture.0.CaptureClear
Capture.0.CaptureEnable =
ON Program.1.Initiate

Do While TRUE

    Wait for Capture.0.CaptureTriggered AND NOT
    Index.4.CommandInProgress Queue.0.DataIn =
    Capture.0.CapturedMasterPosition Capture.0.CaptureClear

Loop
```

Description of Program 0 Code

Program 0 begins by homing the knife to the follower sensor. Once the home is complete, Program 0 defines several constants used in mathematical equations by the program. The user must enter values for these parameters based on actual machine dimensions. If these values have not been entered, a red dot will appear next to these lines indicating that the application cannot run without them. Once values have been entered, the file should be downloaded to the PTi-210 using PowerTools Studio software.

Program 0 continues by performing several calculations to generate other values used by formulas in the program. These calculations use the constants entered by the user earlier in the program.

Once the calculations are complete, Program 0 prepares the internal Queuing and Capture objects for use in the application. Once the Queue is cleared, and the Capture is enabled and cleared, the application is ready to run. Program 0 initiates Program 1, which runs on Task 1. This means that Program 1 will run simultaneously with Program 0. The two programs will work together to perform the application.

Program 0 then goes into a continuous Do While loop. The loop will run forever (or until the Stop is used) because the argument is “TRUE” which will always evaluate as true, and the loop will repeat. Something other than TRUE could be used to control the looping construct if desired (i.e. DriveInput.4 =ON, etc.).

Once in the loop, the program will wait until Capture.0.CaptureTriggered is activated. This means that a registration mark has passed the registration sensor and the encoder position has been captured. We then take the CapturedMasterPosition and load it into Queue 0 using the Queue.0.DataIn instruction. Queue 0 will store the captured master encoder positions, which will be used by Program 1. Once the data is loaded into Queue 0, the program clears the capture object so that it is ready for the next registration mark to pass the sensor. This loop is repeated to keep the queue full of captured positions.

Program 1 – Actual Program Code

```
Do While TRUE
             Capture.1.CaptureClear
             Capture.1.CaptureEnable = ON

             Wait for NOT Queue.0.QueueEmpty

             Var.DTS = Var.MSDTS
             Var.FiltDTS = Var.DTS
```

```
Index.0.Dist = Var.DTS - 0.5000
Index.3.Dist = Var.StartDist - Var.DTS + 0.5000
Queue.0.QueueRemove
Index.3.IndexTime = Var.DTS

Index.0.CompoundInitiate Using Capture.0 On Profile.0
Index.3.InitiateUsing Capture.0 On Profile.1

Index.1.CompoundInitiate On Profile.0
Wait For Capture.1.CaptureTriggered

Do While ((NOT Queue.0.QueueEmpty) AND (NOT DriveInput.6))
       Var.DTS = Var.MSDTS - (Capture.1.CapturedMasterPosition - Queue.0.DataOut)
              Queue.0.QueueRemove
       Capture.1.CaptureClear

       If Var.DTS > Var.MinimumDTS Then
              DriveIO.2.Out = OFF
              Index.2.Dist = Var.DTS
              Index.3.Dist = Var.CycleDist - Var.DTS
              Index.3.IndexTime = Var.DTS
              Index.2.CompoundInitiate Using Capture.1 On Profile.0
              Index.3.Initiate Using Capture.1 On Profile.1
              Index.1.CompoundInitiate On Profile.0
              Var.FiltChnge = Var.DTS - Var.FiltDTS
              Var.FiltChnge = 0.1000 * Var.FiltChnge
              Var.FiltDTS = Var.FiltDTS + Var.FiltChnge
              Wait For Capture.1.CaptureTriggered
       Else
              DriveIO.2.Out = ON
              Index.2.Dist = Var.FiltDTS
              Index.3.Dist = Var.CycleDist - Var.FiltDTS
```

```
                        Index.3.IndexTime = Var.FiltDTS
                        Index.2.CompoundInitiate Using Capture.1 On Profile.0
                        Index.3.Initiate Using Capture.1 On Profile.1
                        Index.1.CompoundInitiate On Profile.0
                        Wait For Capture.1.CaptureTriggered
                        Queue.0.QueueRemove
                EndIf
        Loop

        Index.4.Initiate
        Wait For Index.4.CommandComplete
        Capture.1.CaptureClear
        Wait For NOT DriveInput.6
        Queue.0.QueueClear = ON
Loop
```

Description of Program 1 Code

Program 1 begins with a Do While TRUE loop so the program will remain running until a stop function is activated, or the drive is disabled. Capture Object 1 is then cleared and enabled so that it is ready to capture position data. Capture 1 is used to capture the master encoder position when the working segment index is complete.

Program 0 then waits until Queue 0 is not empty. The Queue will remain empty until the first registration mark has passed the registration sensor. When a mark passes the sensor we are ready to make the first cut.

Once the first mark passes the sensor, we need to make a series of calculations to determine how far, and how fast, the knife must move in order to cut on the registration mark. The first formula sets DTS equal to MSDTS. For the first registration mark, DTS is equal to the distance from the registration sensor to the beginning of the working segment. We then store the initiate value of DTS into another variable called FiltDTS. FiltDTS will continuously store a filtered value for DTS that is used in the event that a registration mark is missed, or two marks are too close together. The next two formulas calculate the distance for Index 0 and Index 3. Indexes 0 and 3 will be initiated and run simultaneously. Their velocities and distances will sum together to yield the knife cutting at exactly the right position and at the right velocity. Index 0 distance is equal to DTS minus the distance it takes for Index 0 to accelerate to velocity. Based on the values entered for this example, it will take the knife 0.5 Inches to get up to its programmed velocity (this value can be found on the Calculations tab in PowerTools Studio software). Index 3 distance is then calculated as the difference between the starting distance and Index 0. Therefore, Index 0 distance and Index 3 distance, sum to equal the starting distance. It is necessary to use two indexes to do this starting move because we must compound one of the indexes into another index at the working ratio. Since both Index 0 and Index 1 have a velocity of the working ratio, they are compounded together, and no velocity change takes place between them. Once both index distances have been calculated, the last piece of data is removed from the queue. We no longer need that data, because we have calculated all of the move profile parameters for that registration mark.

Program 1 then sets Index 3 Time to the value of DTS. Because Index 3 is a synchronized timed index, the user can specify the amount of master distance that the index must complete in. We want Index 3 to complete right at the beginning of the working segment. This means that the time must be set to the master distance between the registration sensor, and the beginning of the working segment. That value is DTS for the first registration mark.

Next, Program 1 initiates both Index 0 and Index 3. Notice that both index initiates have the “Using Capture.0” motion modifier after them. This means that they will both use the information captured in Capture 0 as the starting point for the index. Capture 0 has stored the exact time and position (within 1 usec) that the registration mark passed the registration sensor. By using this captured data as the starting point, we eliminate any time lost between when the mark passed the sensor, and when the index actually gets initiated. This feature keeps the system very accurate. Also notice that Index 0 runs on Profile 0, and Index 3 runs on Profile 1. Specifying what profile to perform the index on is done by using the “On Profile” motion modifier. By specifying two different profiles, the two indexes will run simultaneously. This means that the individual index velocities and distances will sum. The motor will run at Index 0 velocity + Index 3 velocity, and will move Index 0 distance + Index 3 distance. Index 0 compounds into Index 1 because the “CompoundInitiate” instruction is used instead of the standard “Initiate” instruction. Using the compound initiate causes Index 0 to end at its programmed velocity rather than at zero velocity. Index 0 compounds into Index 1 so that the two indexes run in series rather than stopping in between. Index 0 and Index 3 combine to form the start index cycle.

Program 1 then initiates Index 1, which is compounded into by Index 0. Since Index 0 and Index 1 have the same velocity, the transition between indexes will become seamless (without any acceleration or deceleration). Notice that Index 1 also uses the “Compound Initiate” instruction. Index 1 will compound back into Index 0 and will be a repeating cycle. Index 1 comprises the Working Segment, and therefore has a velocity of the Working Ratio and a distance of the Working Segment Length.

Next, the program waits until Capture 1 is triggered. Capture 1 is activated by the Index 1.CommandComplete signal (See Figure 6 above). When Index 1 is done, Capture 1 will activate, and the program will continue on. We use Capture 1 to capture the position of the master encoder at the end of the working segment. This value is used in a calculation to determine how much time (or master distance) we have for the knife to get around to start the working segment for the next registration mark.

Program 1 continues by entering a continuous loop. This loop will repeat continuously until the Stop input is activated, or DriveInput.6 is activated. DriveInput.6 acts as the cycle dropout function. This function allows the user to finish the remainder of the cycle they are currently working on, and then stop at a known position. If DriveInput.6 is inactive, then we check to make sure the queue is not empty.

If the queue **is** empty, that means that no more registration marks have passed the registration sensor since the initial first mark. If the queue is empty, then we continue cycling using a “filtered” value for length between marks. We use this value because in most situations, the manufacturer does not want the machine to stop just because there are no registration marks (or missing marks). Stopping and starting repeatedly can cause more wear-and-tear on the machine components. One could choose to stop in this situation, but for this example, we chose to use a filtered DTS value value. The filtered value that we use is an average value that is calculated by the program. So, if there are missing marks, we will continue to cut using the average gap length.

If the queue **is not** empty, then more registration marks have passed the sensor, and we must calculate how fast the knife must move to position itself for the next cut (next mark). Therefore, Program 1 must calculate a new DTS value (distance from mark to start of working segment). DTS is equal to MSDTS minus the master distance traveled during the last cycle (or the distance from the sensor to the start of the working segment minus the distance traveled during the last segment). Once this calculation is complete, we no longer need the captured data, so we remove the oldest piece of data from the queue using the QueueRemove instruction.

Next, we use the CaptureClear instruction to reset the Capture 1 object. This allows us to capture again at the end of the next working segment.

Program 1 then checks to see if the calculated value for DTS is so small that we cannot possibly get the knife into position in time for the next cut. This would mean that the registration marks were too close together.

If DTS **is** greater than MinimumDTS, then Program 1 calculates new values for Index 2 and Index 3 distance. First, DriveIO.2.Out is cleared in case it was activated from a previous cycle. Index 2, called the Cycle Index, is set equal to DTS. Since Index 2 is a synchronized index, also having a velocity of the working ratio, it will finish at the same time that the registration mark reaches the start of the working segment. The distance for Index 3, called the Correction Index, is calculated as the difference between the parameter CycleDist and the value DTS. This means that Index 2 and Index 3 distance will <u>always</u> sum to equal the CycleDist. This equation must be true otherwise the working segment would shift position in relation to the desired cut point. Index 3 Time is then set to DTS. Again, since Index 3 is a synchronized timed index, by setting the time parameter equal to DTS, it guarantees that the index is complete by the time the registration mark reaches the beginning of the working segment.

After calculating the index distances, Program 1 initiates Indexes 2 and 3 on separate profiles, both using data stored in Capture 1 as their starting point. The data stored in Capture 1 is the exact time that the working segment ended. Since there could be some time (and therefore master distance) lost during the index calculations above, we need to specify exactly when the indexes are/were supposed to start. The “Using Capture” instruction is used for this purpose. Notice that the “CompoundInitiate” instruction is used for Index 2 so that it smoothly transitions into the next working segment index without stopping in between.

The next instruction in Program 1 then initiates Index 1, the Working Segment Index, on the same profile as Index

2. Since Index 2 compounds into index 1, the motor will not stop in between. Again, the “CompoundInitiate” instruction is used so that Index 1 will smoothly transition into Index 2 on the next cycle.

Next, a new value for Filt DTS is calculated. The filtered value is calculated by finding the difference between the last real DTS value and the current FilteredDTS value. The difference is the multiplied by 0.1 to scale the maximum amount of change in one cycle. The scaled value is then added to the current FilteredDTS to obtain the new FilteredDTS value.

Once the filtered DTS calculations are done, the program waits until Capture 1 is triggered. This means the next working segment is complete and we need to repeat the loop.

If DTS **is not** greater than MinimumDTS, then we activate DriveIO.2.Out to signify that a cut will be made in the wrong position. An external device could read the status of this output and decide to stop the machine, keep track of how many times the output activates, or do nothing. This output is completely optional and can be used as a diagnostic tool to determine if the material is bad or for any other purpose. The same steps for calculating Index 2 and 3 distances and initiating those indexes are followed as in the “If DTS **is** greater than MinimumDTS” section above, but instead of using the actual DTS value, the filtered DTS value is used in the calculations. This prevents the system from damaging itself due to excessive accel/decels. An additional item is then removed from the queue to prevent a situation where the knife gets completely out of phase, causing every calculated DTS to be less than MinimumDTS and therefore preventing the knife from ever getting back into phase.

If DriveInput.6 is active at the beginning of the next time through the loop, the current cycle will complete and then instead of compounding into Index 2 (the cycle index), the program compounds into Index 4 (the Stop Index).

Index 4 is a registration index that will travel until the home sensor is activated and then stop.

The program then waits for the CycleDropout signal to deactivate, and then clears the Queue object to remove all of the data points stored while the CycleDropout was active. Once the Queue is cleared, we loop to the very top of the program and start the entire cycle over again.