

Rotary Knife with Blended Timed Indexes

Objective

Develop an application solution to accurately cut/perforate/seal registered material using a rotary knife using Timed Blended and Compound index initiate commands.

Overview

Blended and Compound Timed indexes enable rotary knife applications to be done in a simpler manner, which reduces the programming required. It does away with the necessity of using two profiles. 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 EP-P 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 EP-P drive 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.

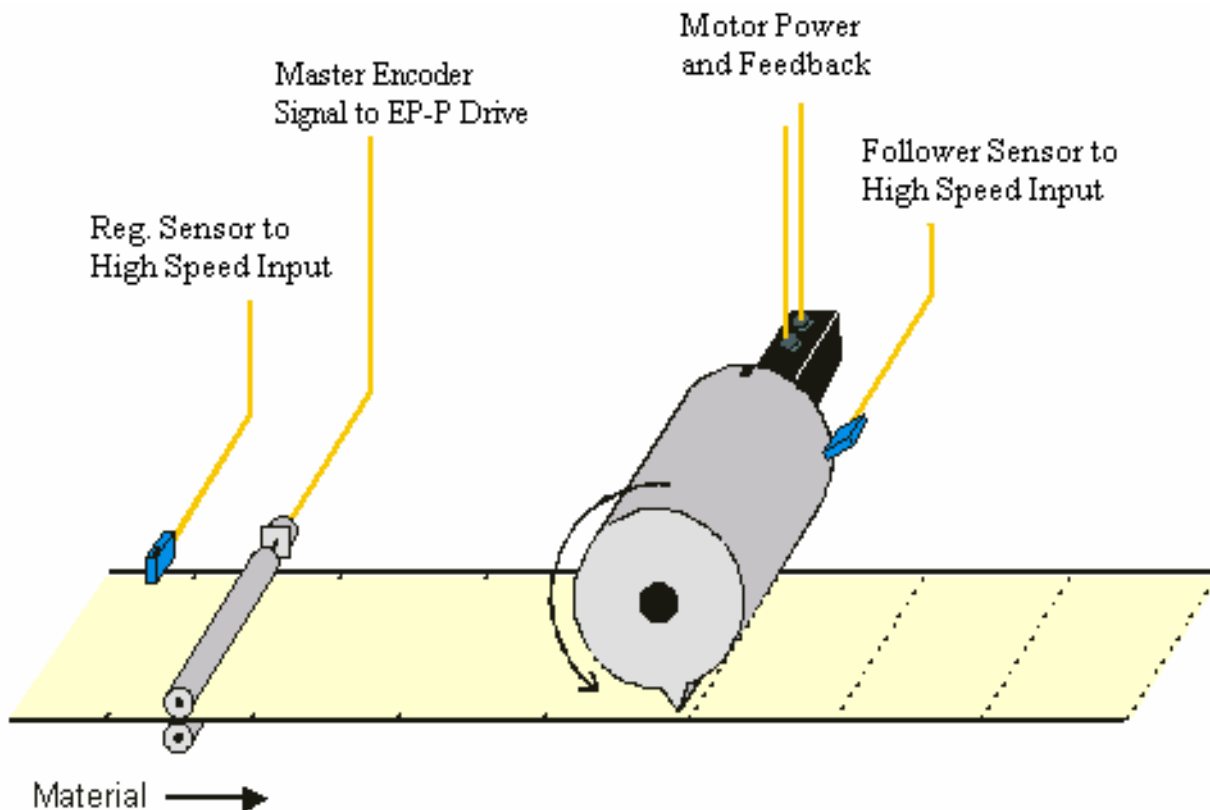


Figure 1 – Example Rotary Knife System

Capture Object

The Capture Object allows the EP-P drive 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 highspeed digital inputs located on the EP-P drive.

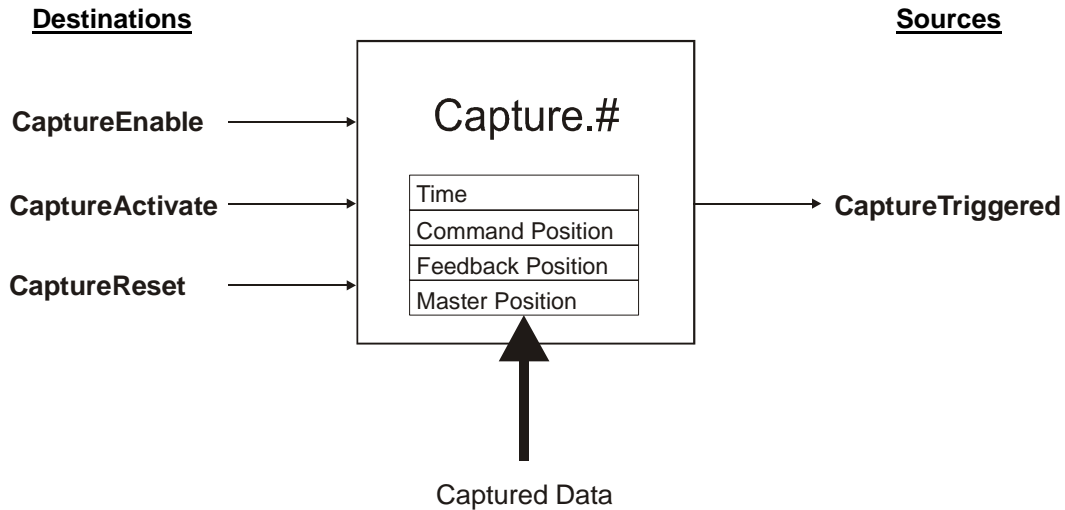


Figure 2 – Capture Object

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 use both methods to operate the capture object.

In order to capture the position of the master encoder when a registration mark passes the sensor, the EP-P drive Input that the sensor is wired to must be assigned to the CaptureActivate destination found on the PowerTools Pro 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 μ sec. 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 turns off.

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

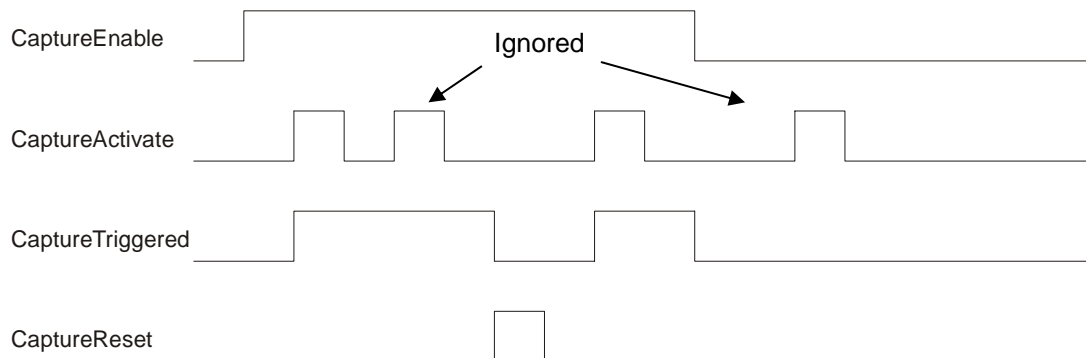


Figure 3 – Capture Timing

Queue Object

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.

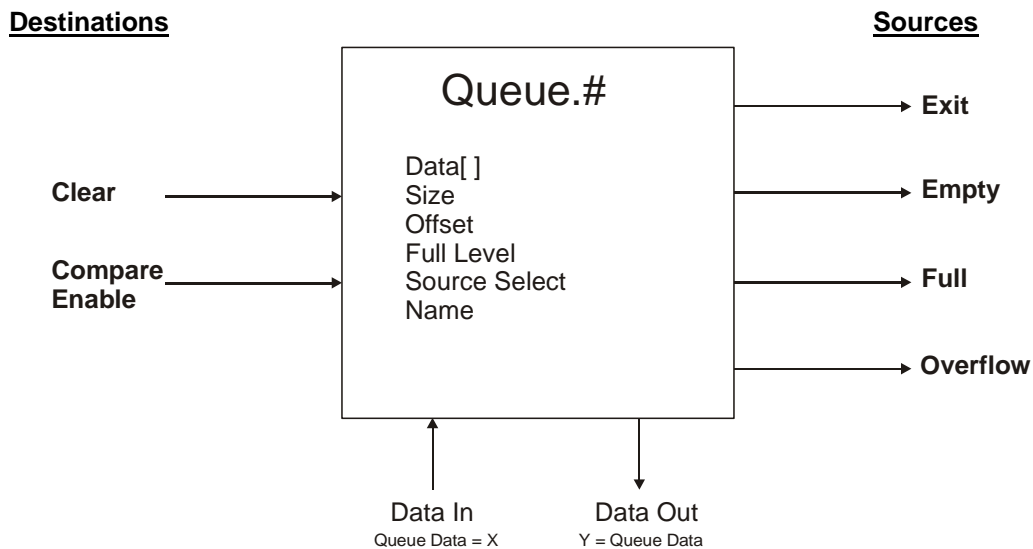


Figure 4 – Queue Object

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.

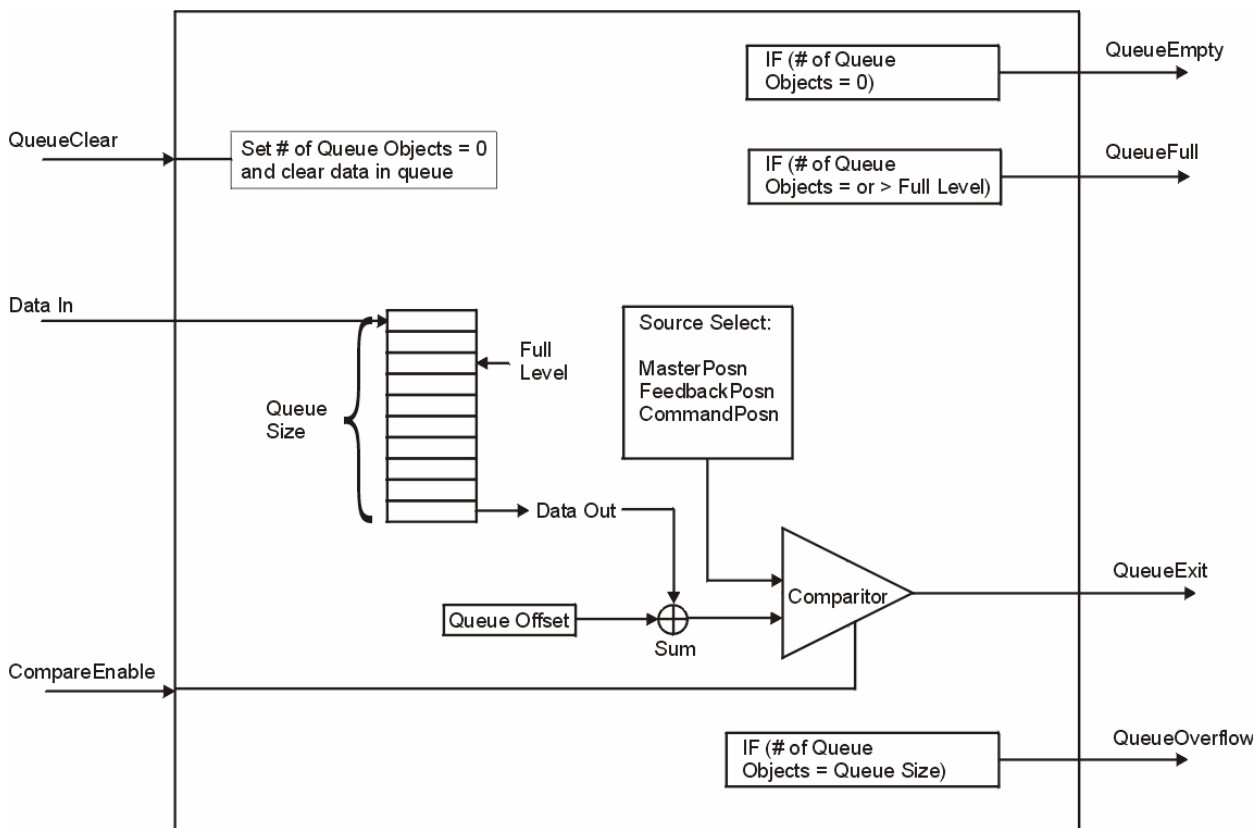
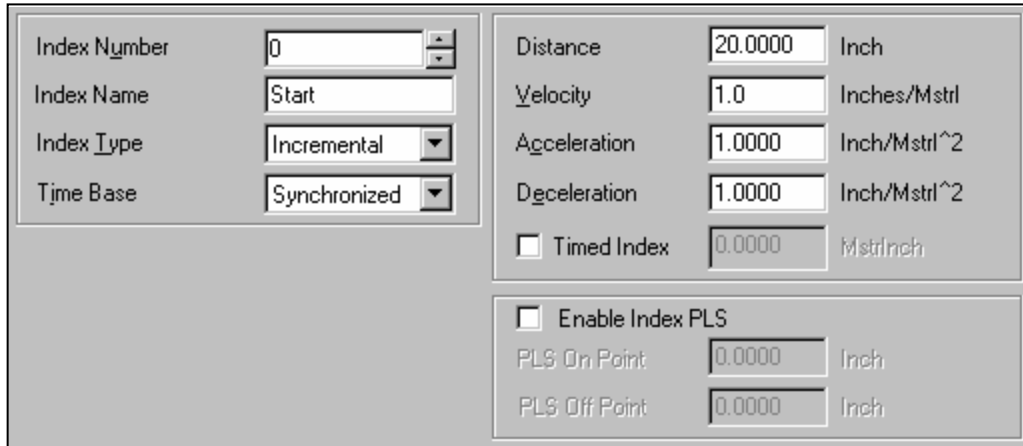


Figure 6 – Detailed View of Queue Object

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.



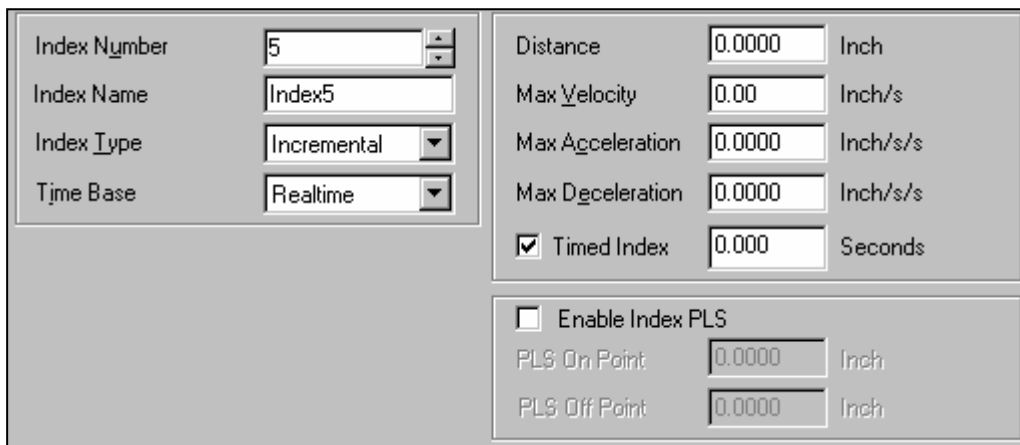
Index Number	0	Distance	20.0000	Inch
Index Name	Start	Velocity	1.0	Inches/Mstrl
Index Type	Incremental	Acceleration	1.0000	Inch/Mstrl^2
Time Base	Synchronized	Deceleration	1.0000	Inch/Mstrl^2
		<input type="checkbox"/> Timed Index	0.0000	Mstrlnch
		<input type="checkbox"/> Enable Index PLS		
		PLS On Point	0.0000	Inch
		PLS Off Point	0.0000	Inch

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	Distance	0.0000	Inch
Index Name	Index5	Max Velocity	0.00	Inch/s
Index Type	Incremental	Max Acceleration	0.0000	Inch/s/s
Time Base	Realtime	Max Deceleration	0.0000	Inch/s/s
		<input checked="" type="checkbox"/> Timed Index	0.000	Seconds
		<input type="checkbox"/> Enable Index PLS		
		PLS On Point	0.0000	Inch
		PLS Off Point	0.0000	Inch

Figure 8 – Timed Index Setup (Realtime)

When the checkbox is enabled, the firmware does an internal calculation to determine what velocity, accel, and decel are necessary to complete the entered 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 instead of Seconds (See Figure 9 below). Doing so allows the user to specify the amount of travel on the master axis in which to complete the index distance on the follower axis. 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.

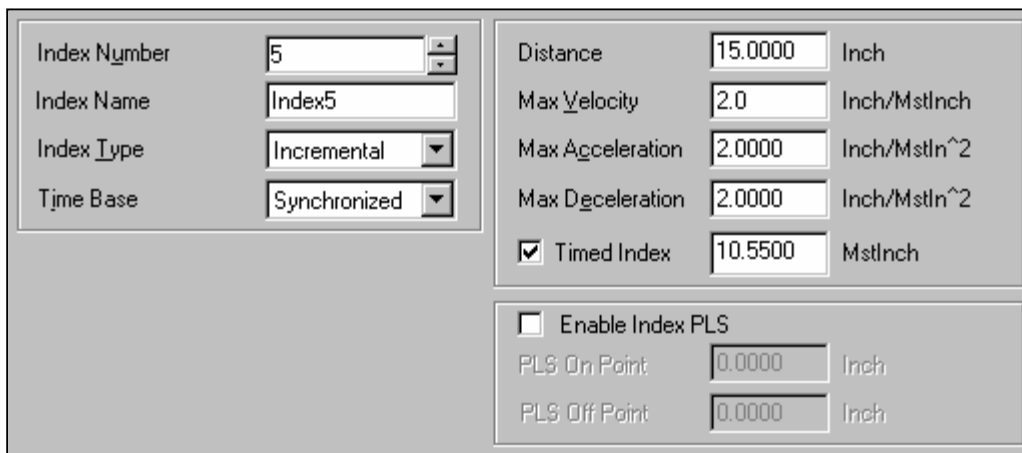


Figure 9 – Timed Index Setup (Synchronized)

Compound Indexes

Compound Indexes allow the user to perform two or more 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.CompoundInitiate
Index.2.CompoundInitiate
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 own programmed velocity. When the index has covered its programmed distance, the motor will then ramp from its current velocity to the next index programmed velocity without stopping. 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

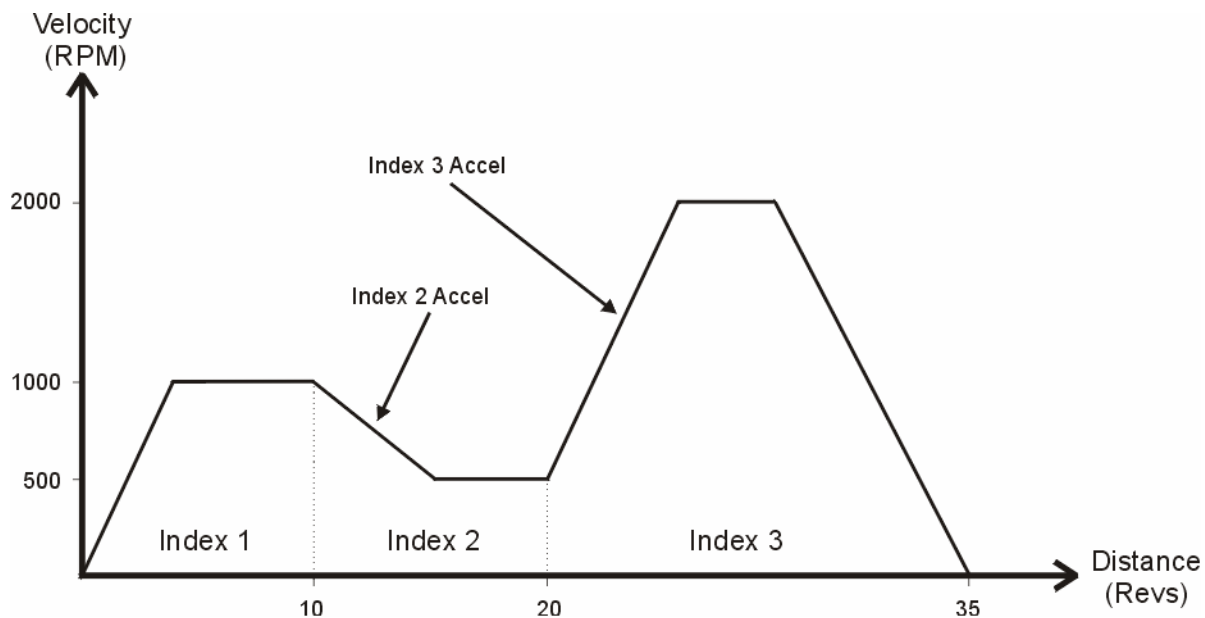


Figure 10 – Compound Index Diagram

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. Even though the deceleration ramp is not used in this case, the user should still enter a deceleration value so that if the same index is used as part of a normal Index.#.Initiate instruction, the motor will correctly decelerate to a stop without a “bump” in motion.

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

Compound Timed Indexes

When Compounding a Timed Index, the timed index will always begin and end at the velocity at which the timed index starts. If the timed index is compound initiated while the motor is stopped, the motor will come to a stop at the end of the timed index even though it is compounded into another index. However, if the timed index is compounded into (from another index), then the timed index will begin and end at the velocity of the previous index.

Figure 11 below shows two examples of using a Compound Timed Index. Notice how Index.1 begins and ends at the velocity of Index.0 since it is a Compounded Timed Index. In the first example the motor must speed up to complete the programmed distance in the allotted time, while in the second example the motor must slow down. Notice that the Velocity of the Timed Index (Index 1) is the same for both examples. So why does the motor speed up in one case and slow down in the other? Remember that the velocity value of a Timed Index is simply a Maximum value used internally to calculate the correct velocity to complete the user-entered distance in the correct time. Note that the Index Distance is greater in the first example and smaller in the second example, which explains the need to speed up and slow down respectively.

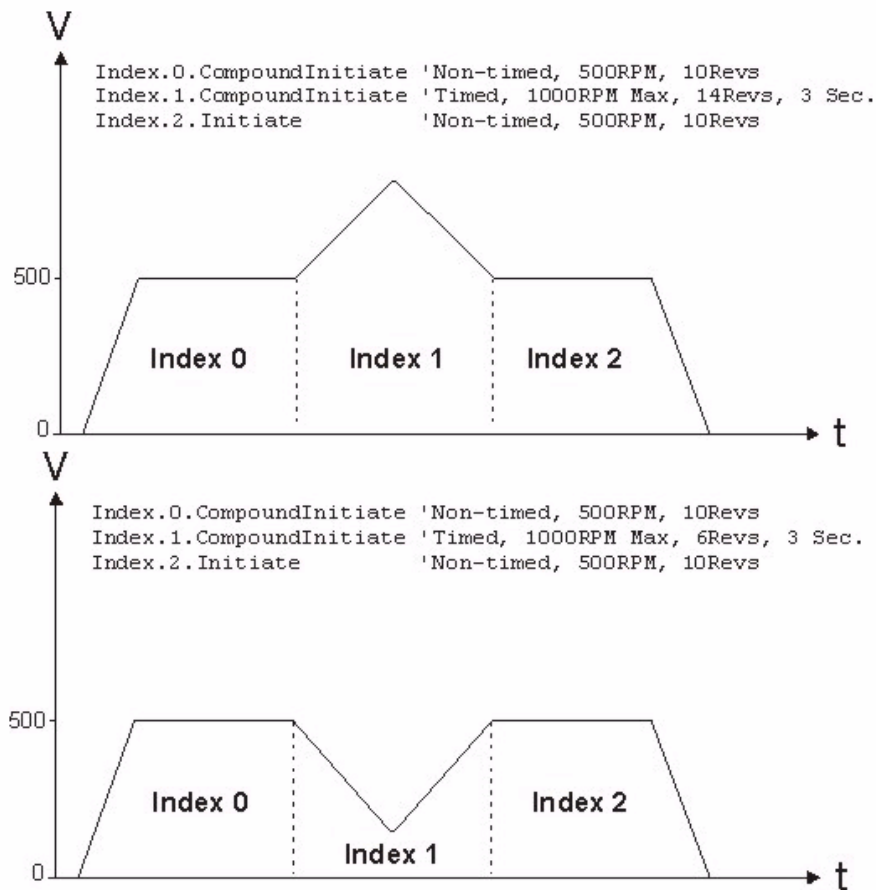


Figure 11 – Compound Timed Index Diagram

The Compound Timed Index is very useful in the rotary knife application to perform the “correction” index because it will always start and end at the velocity of the “Working Segment” index.

Blended Timed Indexes

A Blended Timed Indexes works much the same as a Compound Timed Index, except that a Blended Timed Index can have different beginning and ending velocities.

Figure 12 below shows two examples of using a Blended Timed Index. Notice how the Index 0 which is a Blended Timed Index begins a zero velocity but ends at the velocity of Index 1. Index 1 is also a Blended Timed Index, but notice how it behaves differently in the two examples. The velocity of Index 1 is the same in both examples, so why do they behave differently??? Remember that in a Timed Index, the velocity specified by the user is only a maximum value used internally to calculate the correct velocity to complete the user-entered distance in the correct time. Note that the Index Distance of the first example is much larger than the Distance of the second example. If the Index 1 Distance was set even greater than the first example, the motor might need to increase in speed after Index 0 and then decelerate back down to the Index 2 velocity to complete the distance in the allotted time. As the distance of Index 1 gets smaller and smaller, the profile looks more like that of the second example. Remember though that a Blended Timed Index will never cause the velocity to change sign (go negative in this case). Instead of calculating a negative velocity, the system instead would generate the Index.ProfileLimited signal and the index would take longer to complete than the specified Index Time.

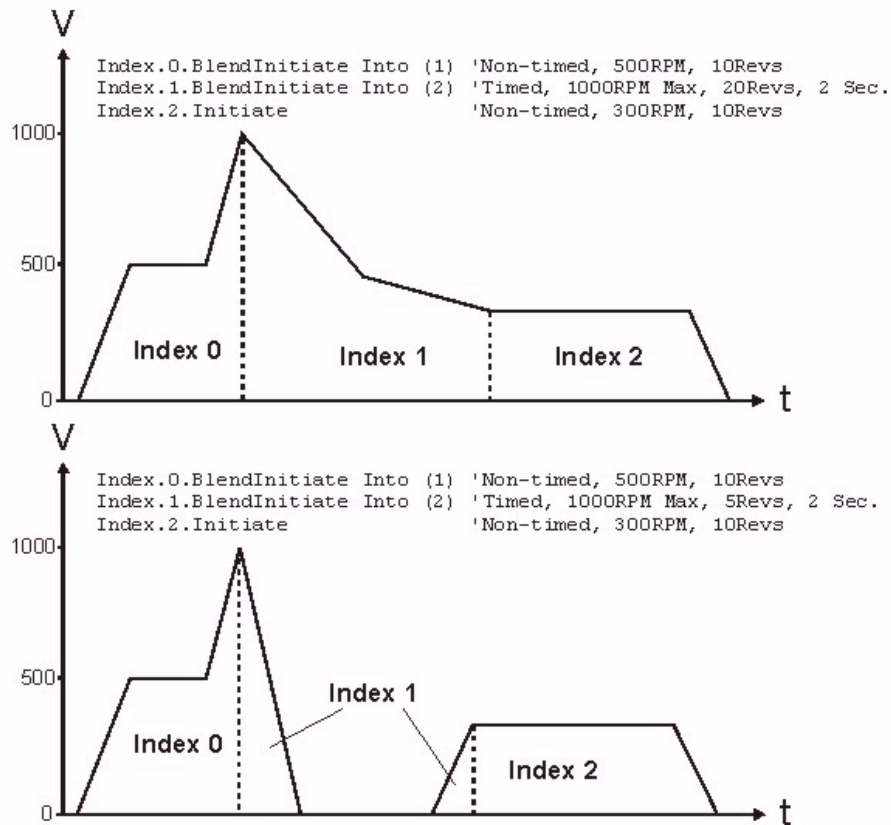


Figure 12 – Blended Timed Index Diagram

The Blended Timed Index is useful in the rotary knife application to perform the initial move from a stopped state up to the velocity of the working segment in a specified master distance.

Graphic with key dimensions and definitions

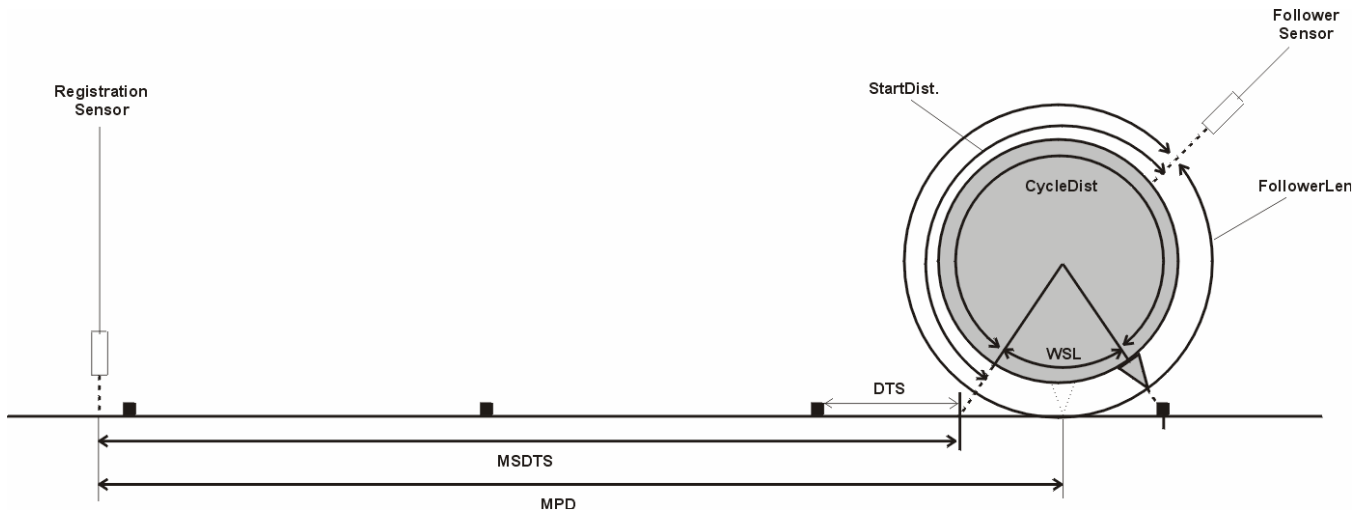


Figure 13 – Rotary Knife graphic with key dimensions

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	Destination	Set From	Polarity
Inputs			Outputs		
DriveInput.1	➔ Capture.0.CaptureActiv...	Active On	DriveOutput.2	← Index.1.CommandInProgre...	Active On
DriveInput.2	➔ Home.0.SensorTrigger	Active On	Capture		
DriveInput.3	➔ Index.3.SensorTrigger	Active On	Capture0		
Index	➔ Program.0.Initiate	Active On	Capture.0.CaptureActivate	← DriveInput.1	Active On
WS			Capture1		
Index.1.CommandComplete	➔ Capture.1.CaptureActiv...	Active On	Capture.1.CaptureActivate	← Index.1.CommandComplete	Active On
Index.1.CommandInProgress	➔ DriveOutput.2	Active On	Program		
			Program0		
			Program.0.Initiate	← DriveInput.3	Active On
			Index		
			StopI		
			Index.3.SensorTrigger	← DriveInput.2	Active On
			Home		
			Home0		
			Home.0.SensorTrigger	← DriveInput.2	Active On

Figure 14 – Assignments Screen

Figure 14 above shows the assignments used in this application.

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

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

DriveInput.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.

DriveOutput.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 achieve 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.

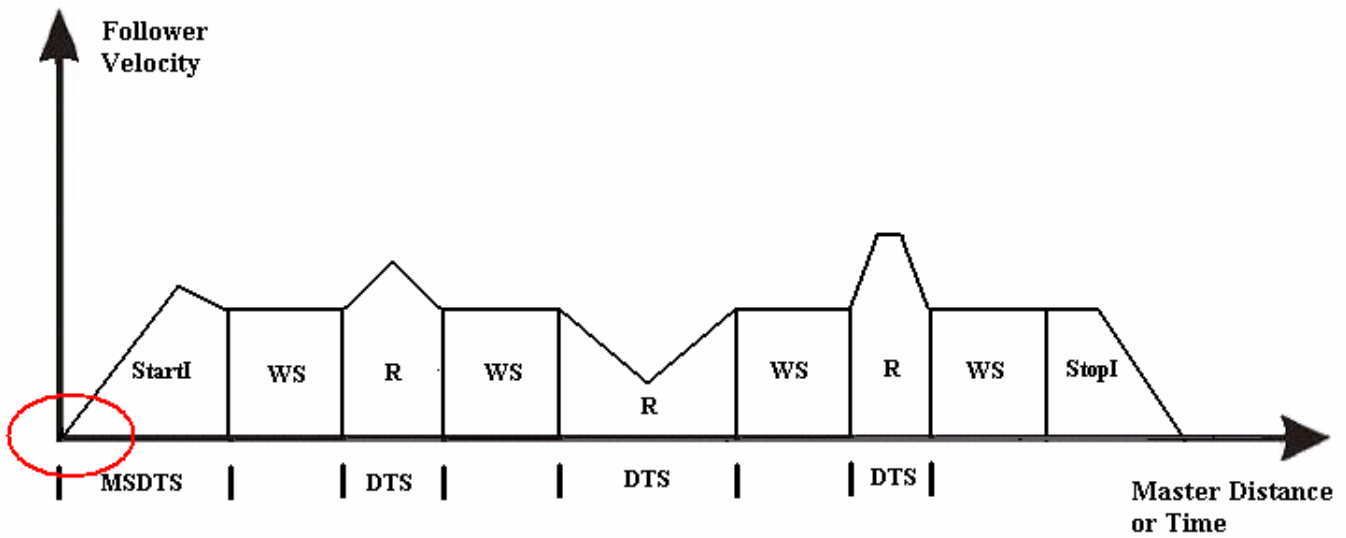


Figure 15 – Motion Profile (Waiting for Registration Mark)

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

Start Up Segment

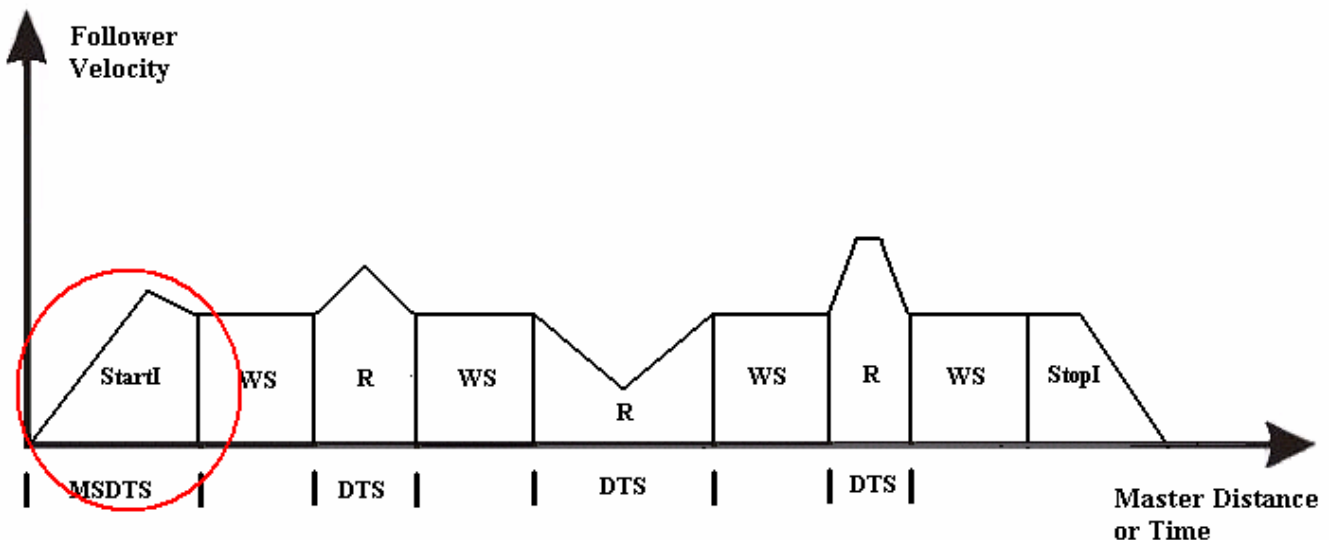


Figure 16 – Motion Profile (Start Up Segment)

Once the first registration mark passes the registration sensor, the program initiates Index StartI using the master position captured by the registration sensor as the starting point for the index. StartI is a synchronized blended timed index, which will complete at the same time as the first part reaches the working segment. Since it blends

into the working segment index, it will complete its velocity at that of the working segment, which is equal to the working ratio. The distance for this index to travel is equal to the distance from where the home completes to the start of the working segment.

Working Segment

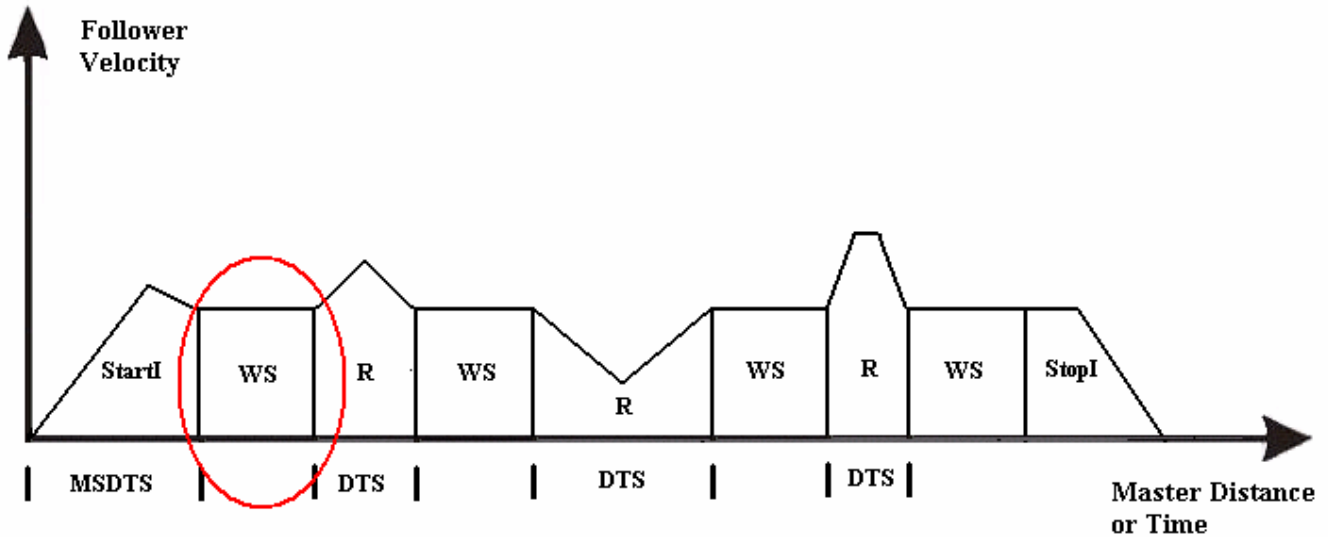


Figure 17 – Motion Profile (Working Segment)

When the start index completes, 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 “WS” 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.

Correction Segment

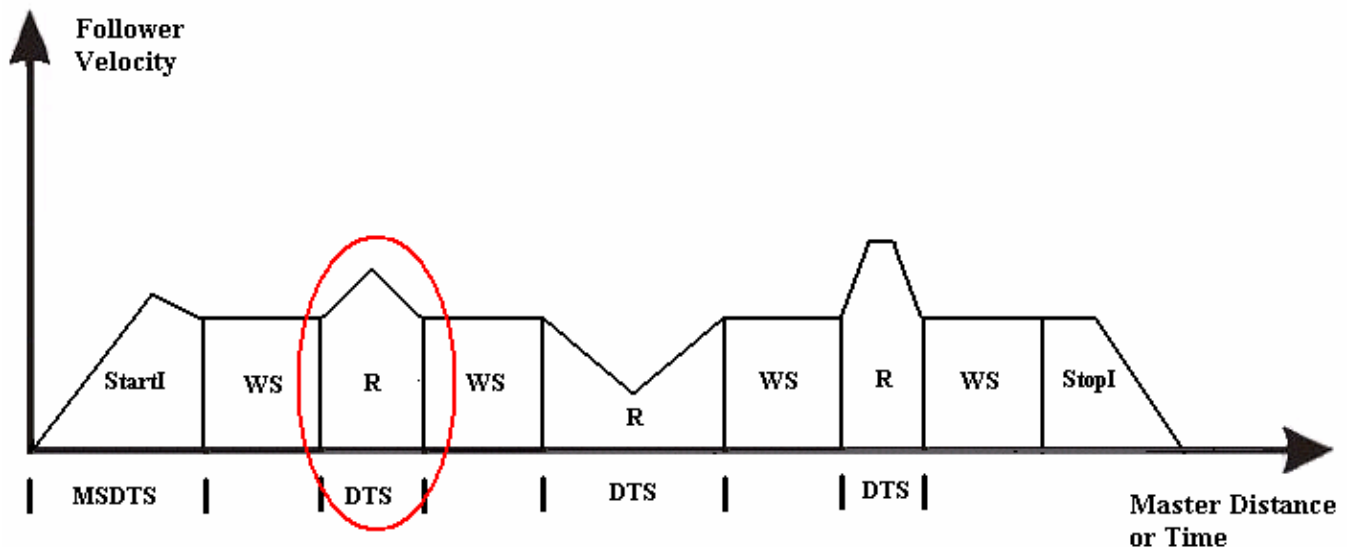


Figure 18 – Motion Profile (Correction Segment)

The next segment of the motion profile is the correction segment. The correction segment is made a compound timed index. Its distance is always equal to a full rotation of the knife minus the working segment length. The master time for this index is the only parameter that changes. The Time for Index "Recovery" is set to DTS. Because it is a synchronized index, the time specified is actually a master distance. Setting the time to DTS 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.

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 the recovery index will always remain the same.

Cycle Dropout Segment

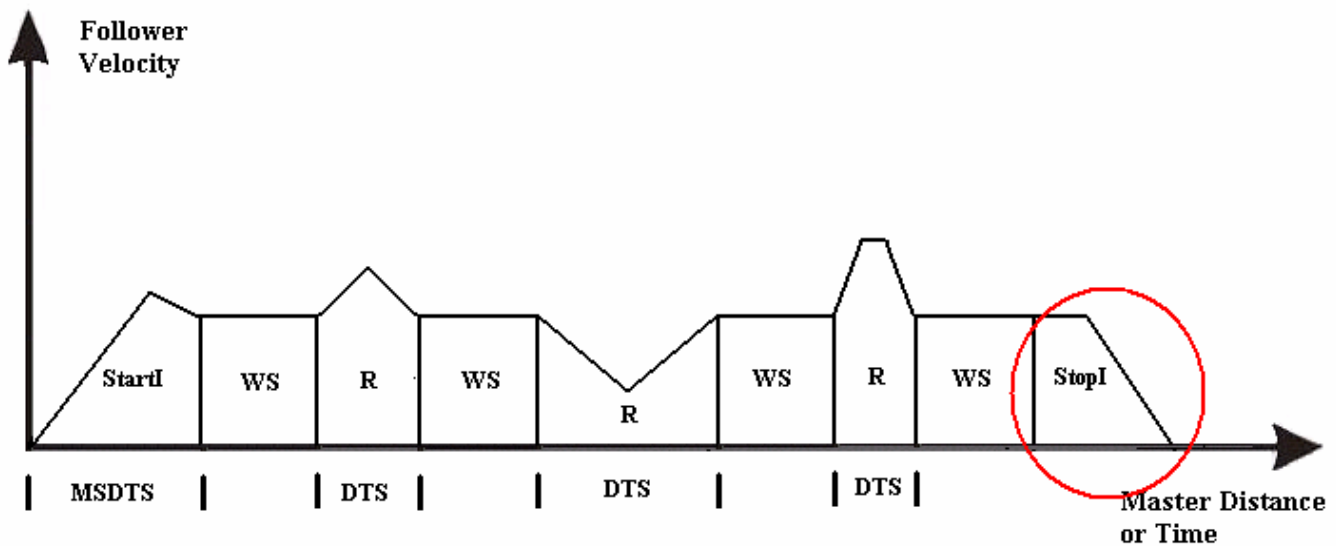


Figure 19 – Motion Profile (Cycle Dropout Segment)

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 "StopI" 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 started 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	Inches
Index Name	Start1	Max Velocity	100.0	Inches/Mstrl
Index Type	Incremental	Max Accel	2.0000	Inches/Mst ²
Time Base	Synchronized	Max Decel	2.0000	Inches/Mst ²
		<input checked="" type="checkbox"/> Timed	0	Mstrlnch

Figure 20 – Start Index Setup

Index Number	1	Distance	4.0000	Inches
Index Name	WS	Velocity	1.0	Inches/Mstrl
Index Type	Incremental	Acceleration	5.0000	Inches/Mst ²
Time Base	Synchronized	Deceleration	1.0000	Inches/Mst ²
		<input type="checkbox"/> Timed	0.0000	Mstrlnch

Figure 21 – Working Segment Index Setup

Index Number	2	Distance	0.0000	Inches
Index Name	Recovery	Max Velocity	50.0	Inches/Mstrl
Index Type	Incremental	Max Accel	10.0000	Inches/Mst ²
Time Base	Synchronized	Max Decel	10.0000	Inches/Mst ²
		<input checked="" type="checkbox"/> Timed	0.0000	Mstrlnch

Figure 22 – Correction Index Setup

Index Number	3	Limit Distance	30.0000	Inches
Index Name	Stop1	Velocity	5.00	Inches/s
Index Type	Registration	Acceleration	10.0000	Inches/s/s
Time Base	Realtime	Deceleration	10.0000	Inches/s/s
		<input checked="" type="checkbox"/> Timed	177.020	seconds

Figure 23 – Stop (or Cycle Dropout) Index Setup

Program 0 – Actual Program Code

```

Home.0.Initiate 'Home0,Sensor,SpecifiedOffset=0.1290Inches,Vel=5.00Inches/s

DriveOutput.1 = ON

' 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 value of DTS which the knife can make a correction

' CycleDist: Distance the knife has to make during the complete revolution minus
' the working segment.
Var.CycleDist = Var.FollowerLen - Var.WSL

' StartDist: Distance the knife has to travel from the homed position to the
' working segment.
Var.StartDist = Var.SensorDist - (Var.WSL/2.0000) - Home.0.SpecifiedOffset

' MSDTS: Master distance from the sensor to the start of the working segment.
Var.MSDTS = var.MPD - Index.8.Dist - var.WSL/2.0000

' Initialize working segments velocity and distance according to
' ratio and working segment length.
Index.WS.Vel = Var.WorkingRatio
Index.WS.Dist = Var.WSL

' Set up the start index.
Index.StartI.IndexTime = Var.MSDTS
Index.StartI.Dist = Var.StartDist

' Set up the correction index.
Index.Recovery.Dist = Var.CycleDist 'Calculate correction distance for time-
based index on profile.1

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 EP-P drive using PowerTools Pro software.

Program 0 continues by performing several calculations to generate other values used by formulas in the program. There are certain values that are known from the start.

- Start Index
- Working Segment Index
- Recovery Distance Index

The start index must cover the distance the knife must cover from the home sensor to the working segment. It must do this in the master distance required to go from the part sensor to the start of the working segment.

The working segment index moves at the ratio of the desired follower speed to that of the master. It covers the determined distance of the working segment.

The distance that the recovery index must cover is equal to a complete revolution of the knife minus the width of the working segment. During operation, the only value that will need to be changed for this index is the master time required, which varies based on the spacing of the parts.

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.5 =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
  Queue.0.QueueRemove

  Index.StartI.BlendInitiate into (1) Using Capture.0
  Index.WS.CompoundInitiate
  Wait for Capture.1.CaptureTriggered

Do While ((NOT Queue.0.QueueEmpty) AND (NOT DriveInput.4))

  Var.DTS = Var.MSDTS - (Capture.1.CapturedMasterPosition - Queue.0.DataOut)
  Queue.0.QueueRemove
  Capture.1.CaptureClear

  If (Var.DTS > Var.MinimumDTS) Then
    Index.Recovery.IndexTime = Var.DTS

    Index.Recovery.CompoundInitiate
    DriveOutput.1 = OFF
    Index.WS.CompoundInitiate
    Wait for Capture.1.CaptureTriggered
  Else
    Index.Recovery.IndexTime = Var.MSDTS
    Index.Recovery.CompoundInitiate
    DriveOutput.1 = ON

    Index.WS.CompoundInitiate
    Wait for Capture.1.CaptureTriggered
    Queue.0.QueueRemove
  EndIf
Loop

  Index.StopI.Initiate
  Wait For Index.StopI.CommandComplete
  Capture.1.CaptureClear
  Wait For NOT DriveInput.4
  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 1 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, the program 1 initiates Index.StartI. Notice that the index initiates has the "Using Capture.0" motion modifier after it. This means that it will 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.

Index.StartI is a blended timed index which will do distance recovery and since it is set to blend into the index of the working segment, it will finish at the same velocity as Index.WS. Since it is a timed index, it will complete this motion in the same time as it takes for the part to move from the part sensor to the working segment. Note that the actual accel and decel ramps of this index can be adjusted as long as the values selected do not prevent the

index from being completed in the allotted time.

Program 1 then initiates Index.WS as a compound index. The working segment is a compound index that is set as a non-timed index. Index.WS 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.WS.CommandComplete signal (See Figure 6 above). When Index.WS 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.4 is activated. DriveInput.4 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.4 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 default 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 the MSDTS value. So, if there are missing marks, we will continue to cut using the default 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 simply loads this value as the index time of Index.Recovery. Note that the previous algorithm using two profiles required several calculations. With the timed compound index, the new index time the only parameter that must be loaded. Also note that since there is no longer any need for the "Using Capture" command.

After initiating the recovery index, the program simply 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 DriveOutput.1 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 recovery index is loaded with the default distance and initiated as a compound timed index as was done in the successful state. 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.4 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.Stop1 (the Stop Index). Index.Stop1 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.