PTiAN-02 Rev 1

Compound Indexing with the PTi-210 Module

Objective

Create an application that demonstrates Compound Indexes using the PTi-210 module and PowerTools Studio.

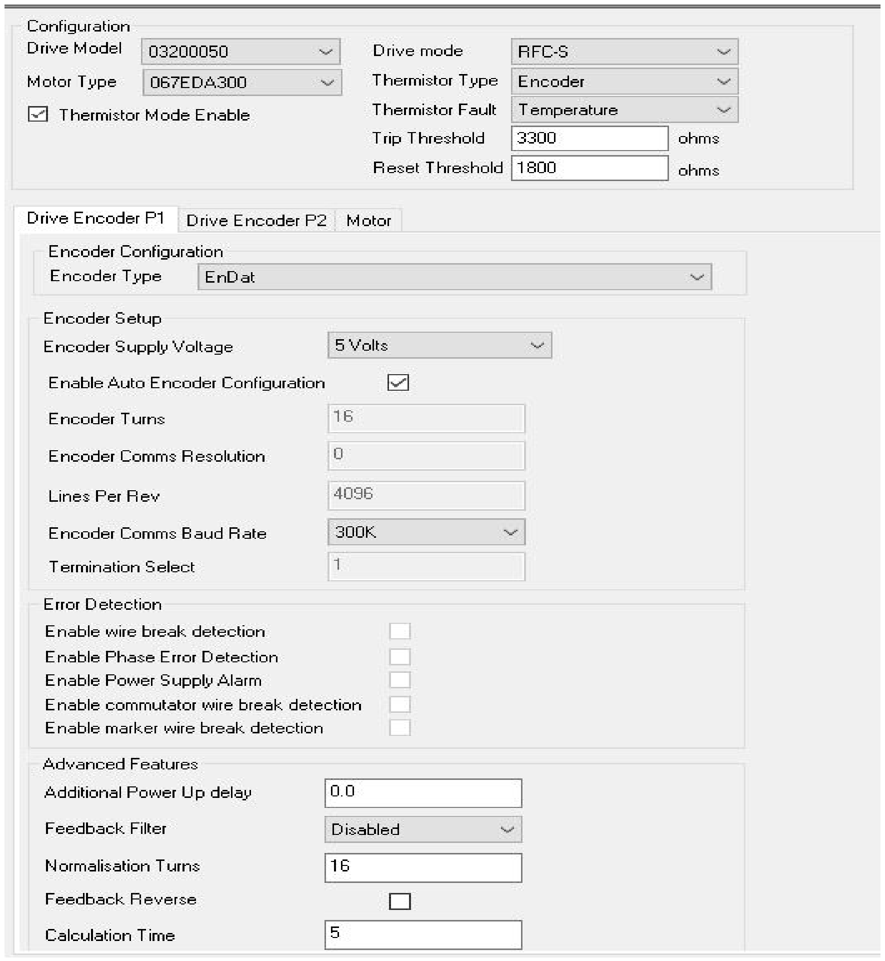
Solution Summary

Indexes are a means of moving the motor a specified distance or to a specified position. Each index being at zero velocity and ramps to a specified target velocity, and then decelerates back to zero speed. In some applications, the user desires to link two or more indexes together without coming to a stop in-between. The PTi-210 module accomplishes this task using Compound Indexes.

A user program must be created to "Compound" the desired indexes together. Doing so gives the motion a totally different profile than if the indexes are started using the standard "Index Initiate". The following Application Tool will demonstrate the usage of Compound Indexing.

Step 1 – Select Drive/Motor

Using PowerTools Studio, expand the Hardware branch on the hierarchy, and select the Motor/Encoder branch. Doing so will display the Motor/Encoder view as seen in Figure 1 below. This view is used to select and configure the drive and motor to be used in the application. If these parameters are not configured properly, the Unidrive M70x/DigitaxHD will not be able to control the motor correctly.



Configuration

Drive Model	03200050	Drive mode	RFC-S
Motor Type	067EDA300	Thermistor Type	Encoder
☑ Thermistor Mode Enable		Thermistor Fault	Temperature
		Trip Threshold	3300 ohms
		Reset Threshold	1800 ohms

Drive Encoder P1 | Drive Encoder P2 | Motor

Encoder Configuration

Encoder Type	EnDat

Encoder Setup

Encoder Supply Voltage	5 Volts
Enable Auto Encoder Configuration	☑
Encoder Turns	16
Encoder Comms Resolution	0
Lines Per Rev	4096
Encoder Comms Baud Rate	300K
Termination Select	1

Error Detection

Enable wire break detection	☐
Enable Phase Error Detection	☐
Enable Power Supply Alarm	☐
Enable commutator wire break detection	☐
Enable marker wire break detection	☐

Advanced Features

Additional Power Up delay	0.0
Feedback Filter	Disabled
Normalisation Turns	16
Feedback Reverse	☐
Calculation Time	5

Figure 1 – Drive/Encoder View

Drive Type – Select the drive model number that matches the drive you are working with from the list box.

Motor Type – Select the servo motor model that matches the motor you are using from the list box.

Drive Mode – Select what type of application the drive will be configured for. Available selections are RFC-S for servo motors and RFC-A for induction motors. Based on the setting of this parameter, information on the Motor tab will change.

Thermistor Type - If the Thermistor mode is enabled, this option allows the thermistor type to be selected.

Thermistor Fault - This option sets the P1 Thermistor Fault Detection parameter (3.123) on the drive.

Trip Threshold - The value of this field is used to set the P1 Thermistor Trip Threshold parameter (3.120) on the drive.

Reset Threshold - The value of this field is used to set the P1 Thermistor Reset Threshold parameter (3.121) on the drive.

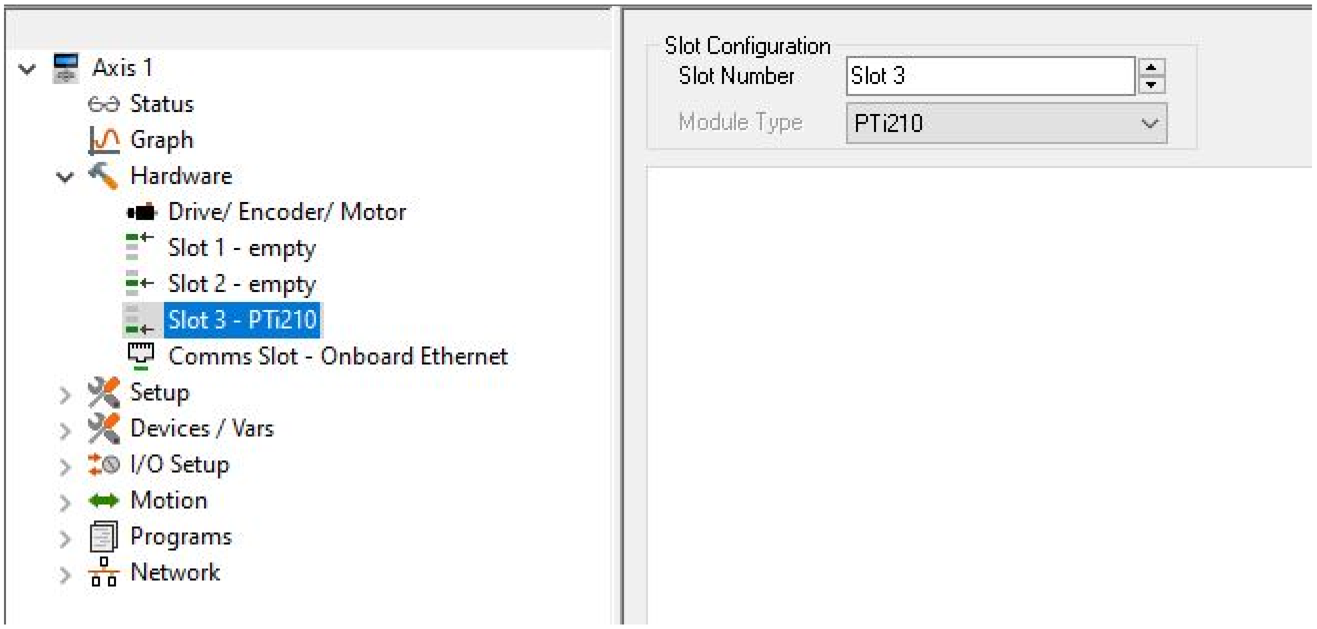
Drive Encoder P1 – This tab configures the settings for the encoder if connected to P1 on the drive. The settings are determined by the type of encoder. Refer to your motor/encoder documentation to determine the type of encoder being used.

Drive Encoder P2 – This tab configures the settings for the encoder if connected to P2 on the drive. The settings are determined by the type of encoder. Refer to your motor/encoder documentation to determine the type of encoder being used.

Motor – This tab allows the user to use the motor from the DDF file or manually adjust the motor parameters.

Step 2 – **Configure the Option Module Slots**

next step is to define what type of Solutions Module is fitted in each of the Unidrive M70x/DigitaxHD option slots. Figure 2 below is an example of a Slot Setup view. For this application, the only module necessary is a PTi-210 module. For this example, the Slots 1 and 2 are empty, and the PTi-210 is populated in Slot 3.

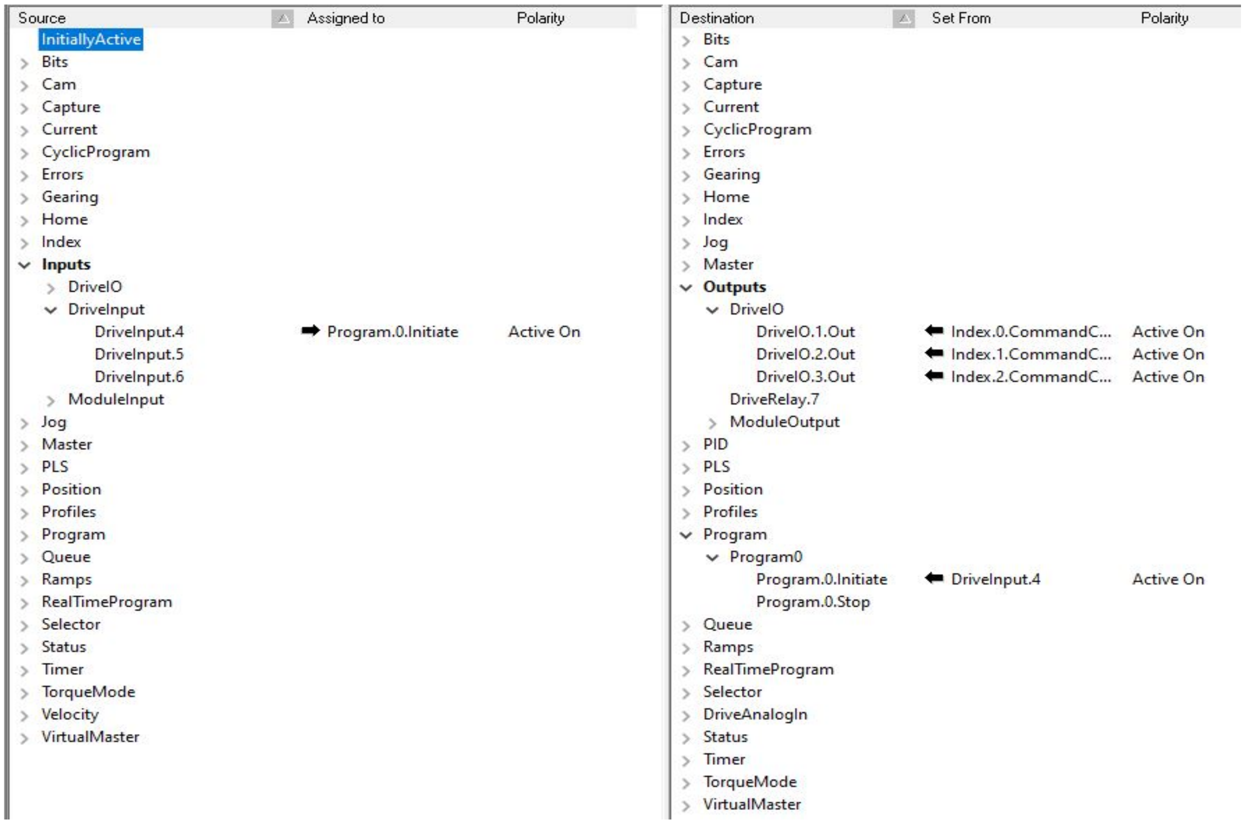


Slot Number: Slot 3
Module Type: PTi210

Figure 2 – Slot 3 View

Step 3 – Configure the Assignments

In order to use the Compound Indexing feature, the user must create a user program. Therefore, a method to initiate the user program is required. Figure 3 and 4 below shows the Assignments view with the necessary I/O configured.



Source	Assigned to	Polarity
InitiallyActive		
Bits		
Cam		
Capture		
Current		
CyclicProgram		
Errors		
Gearing		
Home		
Index		
Inputs		
DriveIO		
DriveInput		
DriveInput.4	Program.0.Initiate	Active On
DriveInput.5		
DriveInput.6		
ModuleInput		
Jog		
Master		
PLS		
Position		
Profiles		
Program		
Queue		
Ramps		
RealTimeProgram		
Selector		
Status		
Timer		
TorqueMode		
Velocity		
VirtualMaster		

Destination	Set From	Polarity
Bits		
Cam		
Capture		
Current		
CyclicProgram		
Errors		
Gearing		
Home		
Index		
Jog		
Master		
Outputs		
DriveIO		
DriveIO.1.Out	Index.0.CommandC...	Active On
DriveIO.2.Out	Index.1.CommandC...	Active On
DriveIO.3.Out	Index.2.CommandC...	Active On
DriveRelay.7		
ModuleOutput		
PID		
PLS		
Position		
Profiles		
Program		
Program0		
Program.0.Initiate	DriveInput.4	Active On
Program.0.Stop		
Queue		
Ramps		
RealTimeProgram		
Selector		
DriveAnalogIn		
Status		
Timer		
TorqueMode		
VirtualMaster		

Figure 3 – Assignments used in example

Following is a description of each of the assignments created:

DriveInput.4 – Program.0.Initiate

This assignment is used to start the user program sequence. When a rising edge of DriveInput.4 is detected, it causes Program.0.Initiate to activate, and therefore the program begins.

Index.0.CommandComplete – DriveIO.1.Out

This assignment is used to activate a digital output when Index 0 is complete.

Index.1.CommandComplete – DriveIO.2.Out

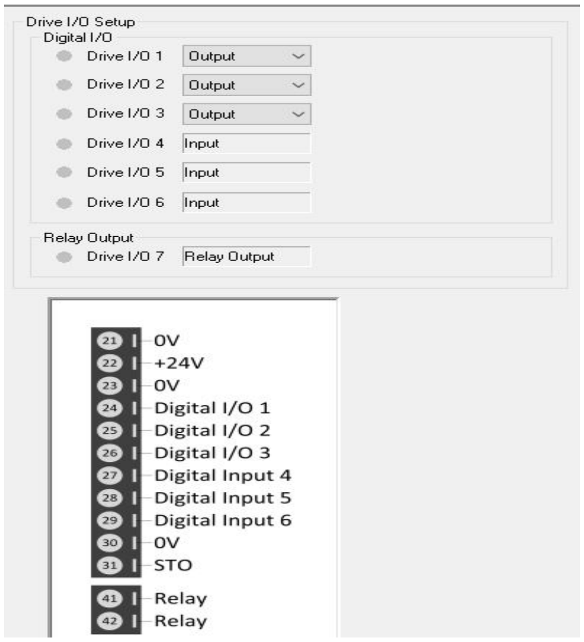
This assignment is used to activate a digital output when Index 1 is complete.

Index.2.CommandComplete – DriveIO.3.Out

This assignment is used to activate a digital output when Index 2 is complete.

Step 4 – Configure the I/O

This application example uses the first three I/O points on the Unidrive M700 as digital outputs. Therefore, we must configure those I/O as outputs. This is done on the Drive I/O Setup view found under the I/O Setup group in the PowerTools Studio hierarchy. Figure 4 below shows an example of this screen with Drive I/O 1-3 configured as Outputs.



Drive I/O Setup

Digital I/O

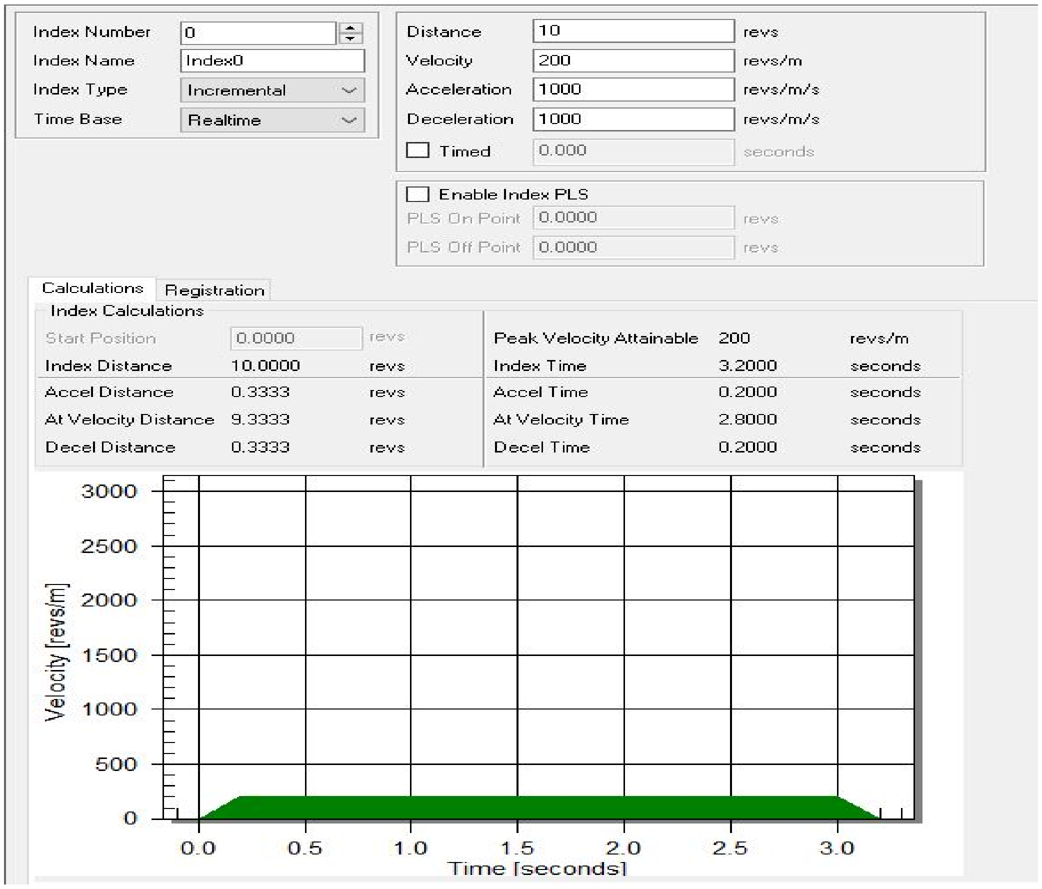
Drive I/O 1	Output
Drive I/O 2	Output
Drive I/O 3	Output
Drive I/O 4	Input
Drive I/O 5	Input
Drive I/O 6	Input

Relay Output

Drive I/O 7	Relay Output

Figure 4 – Drive I/O Setup View

Step 5 – Configure the Indexes



Index Number	0
Index Name	Index0
Index Type	Incremental
Time Base	Realtime

Distance	10	revs
Velocity	200	revs/m
Acceleration	1000	revs/m/s
Deceleration	1000	revs/m/s
☐ Timed	0.000	seconds

☐ Enable Index PLS

PLS On Point	0.0000	revs
PLS Off Point	0.0000	revs

Calculations | Registration

Index Calculations

Start Position	0.0000	revs	Peak Velocity Attainable	200	revs/m
Index Distance	10.0000	revs	Index Time	3.2000	seconds
Accel Distance	0.3333	revs	Accel Time	0.2000	seconds
At Velocity Distance	9.3333	revs	At Velocity Time	2.8000	seconds
Decel Distance	0.3333	revs	Decel Time	0.2000	seconds

Figure 5 – Index 0 View

This view is used to configure the motion performance of the index profile. Following is a description of each of the Index parameters:

Index Number – The PTi-210 can include up to 55 index profiles in a single application. Use the up and down arrows next to the Index Number to scroll through the available indexes.

Index Name – The Index Name parameter allows the user to give a descriptive name to the index profile. The Name may be up to twelve characters in length. Alpha and Numeric characters may be used, however, no spaces are allowed (if spaces are used they will automatically be converted to underscores).

Index Type – Use this list box to select the desired Index Type. The Index Type directly impacts how the index profile works. Following is a brief description of each of the available index types.

Absolute – An Absolute Index is used to move the motor to a specific position. After completing an Absolute Index, the motor will always be in the same position regardless of the starting position of the motor. The direction that the motor moves during an Absolute Index is dependent upon its position when the index is initiated.

If an Absolute Index is initiated a second time, just after completing the first index the motor will not move because it is already at its specified absolute position.

Correction – Correction Indexes are used to follow a dynamic fieldbus or analog value that changes the distance of the index prior to and during the index motion. Correction indexes use incremental distance values. The distance value can be updated via fieldbus, by simply writing to the index distance parameter. If the analog input's Destination Variable is set to an Index distance parameter, the index's distance value will be updated by the Analog to Position scaling found in the Analog Input view.

Incremental – An Incremental Index is used to make the motor travel a specified distance each time the index is initiated. The final position after the Index is completed is entirely dependent on the starting position before the Index is initiated.

If an Incremental Index is initiated a second time, it will move the same distance each time.

Position Tracker – Position Tracker indexes are used to follow a dynamic fieldbus or analog value that changes the end point of the index prior to and during the index motion. Position Tracker indexes use absolute position values. The position value can be updated via Fieldbus, by simply writing to the index position parameter. If the analog input's Destination is set to an Index number, the index's position value will be updated by the Analog to Position scaling found in the Analog Input view.

Registration – A Registration Index functions much the same as a Home profile. The index runs at a specified velocity until a registration signal activates. Once the signal activates, the index either beings to decelerate immediately, or it continues at velocity for a specified offset distance. The Index.#.SensorTrigger is the event used to activate the Registration Sensor for the given index.

Rotary Plus – A Rotary Plus Index is used when Rotary Rollover is active. The Rotary Plus Index is similar to an Absolute Index, but it is forced to go in the positive direction to get to its programmed position. The programmed position for a Rotary Plus Index must be within the Rotary Rollover range (Posn < Rotary Rollover).

Rotary Minus – A Rotary Minus Index is used when Rotary Rollover is active. The Rotary Minus Index is similar to an Absolute Index, but it is forced to go in the negative direction to get to its programmed position. The programmed position for a Rotary Minus Index must be within the Rotary Rollover range (Posn < Rotary Rollover).

Distance – The Distance parameter defines how far (in user units) the motor will move when the index is initiated. The sign of this parameter determines the direction that the motor will move (negative distance causes negative movement, positive distance causes positive movement).

When working with Absolute, Position Track Once, Position Track Cont., Rotary Plus, or Rotary Minus Index types, this value will change to say “Position” instead of “Distance”. This is because these index types all cause the motor move to a desired position (with respect to zero) instead of a desired distance.

In the case of a Registration Index type, this value specifies the maximum distance that the motor will travel in search of a Registration Sensor. If no registration sensor is seen, the motor will come to a stop exactly this distance away from the starting position.

Velocity – This parameter defines the target speed for the Index profile. The motor may or may not be able to reach the programmed velocity depending on the Accel/Decel ramps specified and/or the available current. If the motor cannot reach the specified velocity, then following error will accumulate while the index is in progress.

Acceleration – This parameter defines the ramp used to reach the target velocity when the Index is initiated. The motor may or may not be able to achieve the specified acceleration depending on available system current. If the motor cannot achieve the specified acceleration, then following error will accumulate while the acceleration ramp is in progress.

Deceleration – This parameter defines the ramp used to decelerate to zero velocity when the index is complete. The motor may or may not be able to achieve the specified deceleration depending on available system current or regeneration capabilities. If the motor cannot achieve the specified deceleration, then following error will accumulate while the deceleration ramp is in progress. It is also possible that that drive may trip due to excessive shunt/regen currents.

Timed Index Enable – In some applications, the user may already know the given time in which they need an index to complete. Therefore, rather than forcing the user to calculate the velocity, accel, and decel to result in the desired index time, a Timed Index allows the user to instead enter the desired Index Time directly. When Timed Index Enable is activated, the firmware automatically calculates the Velocity, Acceleration, and Deceleration to complete in the specified time.

All indexes can be configured as Timed Indexes except for the Registration Index type. This is due to the fact that it is unknown when the Registration Sensor will activate, therefore it is impossible to calculate the necessary values.

When the user activates the Timed Index Enable, the Velocity, Acceleration, and Deceleration parameters change to Max. Velocity, Max. Accel, and Max. Decel. In this case, these parameters are used as maximums in the automatic calculation. Therefore, when the firmware calculates the values to complete in the specified time, the values used will never exceed these maximum values. If the maximum values are not large enough to allow the index to complete in the specified Time, then the index will not exceed the maximum values, and a signal called Index.ProfileLimited will activate indicating that the index did not complete in the specified Time.

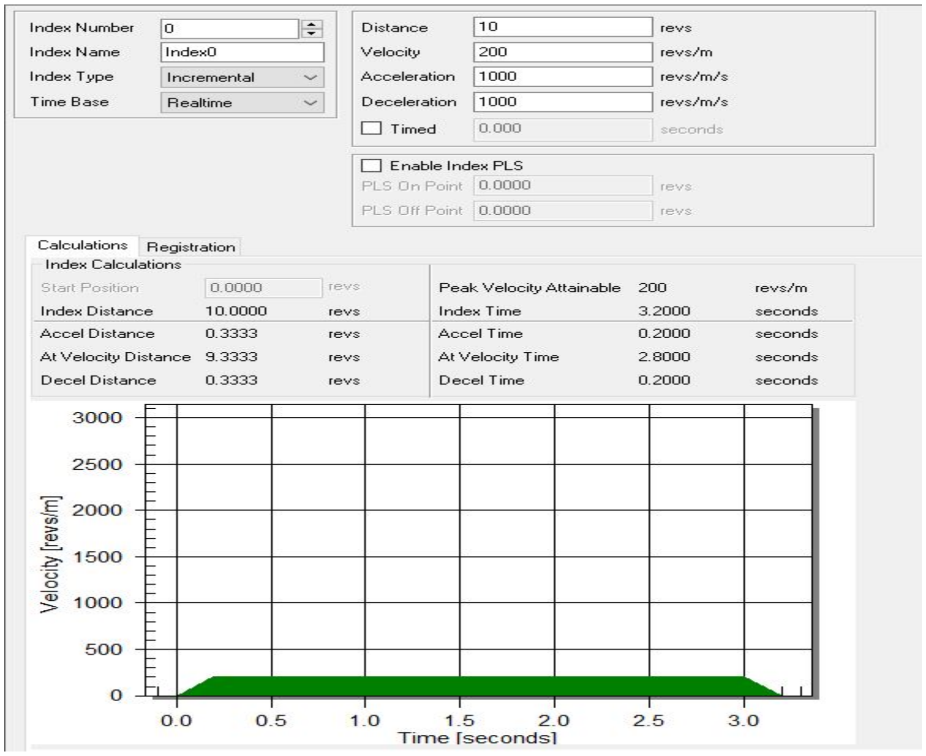
Index Time – When the Timed Index Enable signal is active, the user can specify the amount of time in which an Index should complete. Maximum resolution for this parameter is 0.001 seconds (or 1 msec). In the case of a synchronized index, the units for this parameter are master distance. Therefore, the user could specify the amount of master distance travel in which the index distance must be completed.

Index PLS Enable – Each Index profile has a built-in Programmable Limit Switch (PLS). In order to make the PLS functional, the Index PLS Enable checkbox must be activated. Doing so will cause the Index.#.PLSStatus event to activate when the motor reaches the specified On Point (distance from the starting position of the index), and to deactivate when the motor reaches the Off Point.

Index PLS On Point – If the Index PLS Enable is active, when the given index is initiated, the Index.#.PLSStatus event will activate when the motor reaches this distance from the start of the index. If this value is greater than the index distance, then the Index PLS will never activate.

Index PLS Off Point – If the Index PLS Enable is active, when the given index is initiated, the Index.#.PLSStatus event will deactivate when the motor reaches this distance from the start of the index. If this value is greater than the index distance, then the Index PLS will never deactivate.

Now that we have covered the details of the Index profile, let's create the three Indexes for use in this application example. We will create three indexes with widely varying velocities so that when we run the example index, it is obvious when the motor transitions from one index to the next.



Index Number	0	Distance	10	revs	
Index Name	Index0	Velocity	200	revs/m	
Index Type	Incremental	Acceleration	1000	revs/m/s	
Time Base	Realtime	Deceleration	1000	revs/m/s	
		☐ Timed	0.000	seconds	

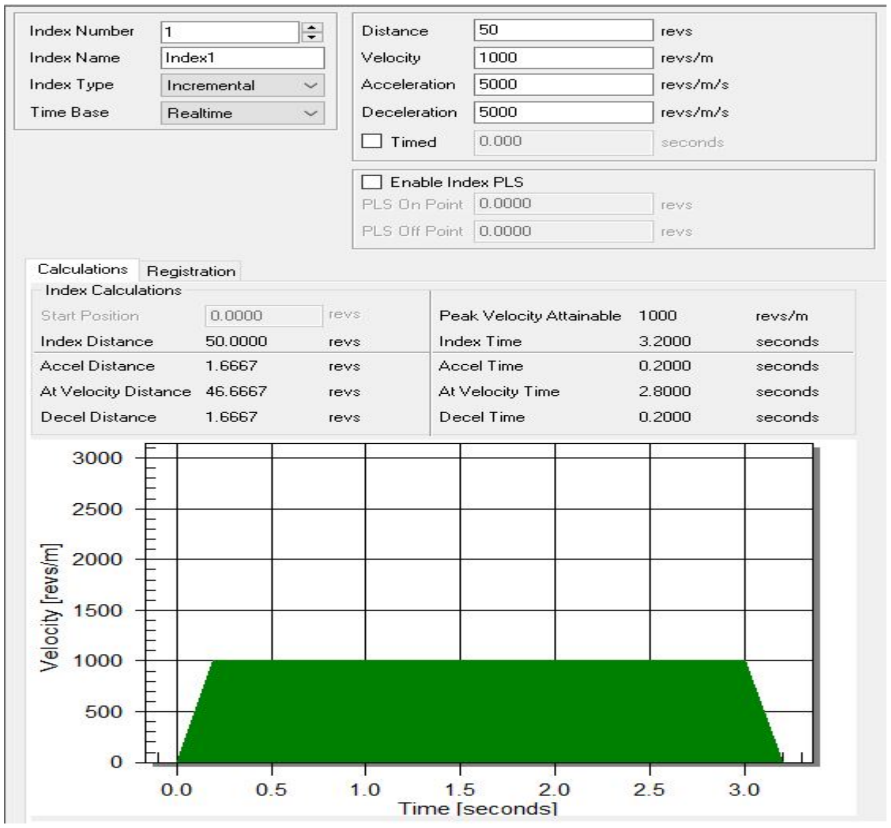
☐ Enable Index PLS
PLS On Point 0.0000 revs
PLS Off Point 0.0000 revs

Calculations | Registration

Index Calculations

Start Position	0.0000	revs	Peak Velocity Attainable	200	revs/m
Index Distance	10.0000	revs	Index Time	3.2000	seconds
Accel Distance	0.3333	revs	Accel Time	0.2000	seconds
At Velocity Distance	9.3333	revs	At Velocity Time	2.8000	seconds
Decel Distance	0.3333	revs	Decel Time	0.2000	seconds

Figure 6 – Index 0 (200 RPM)



Index Number	1	Distance	50	revs	
Index Name	Index1	Velocity	1000	revs/m	
Index Type	Incremental	Acceleration	5000	revs/m/s	
Time Base	Realtime	Deceleration	5000	revs/m/s	
		☐ Timed	0.000	seconds	

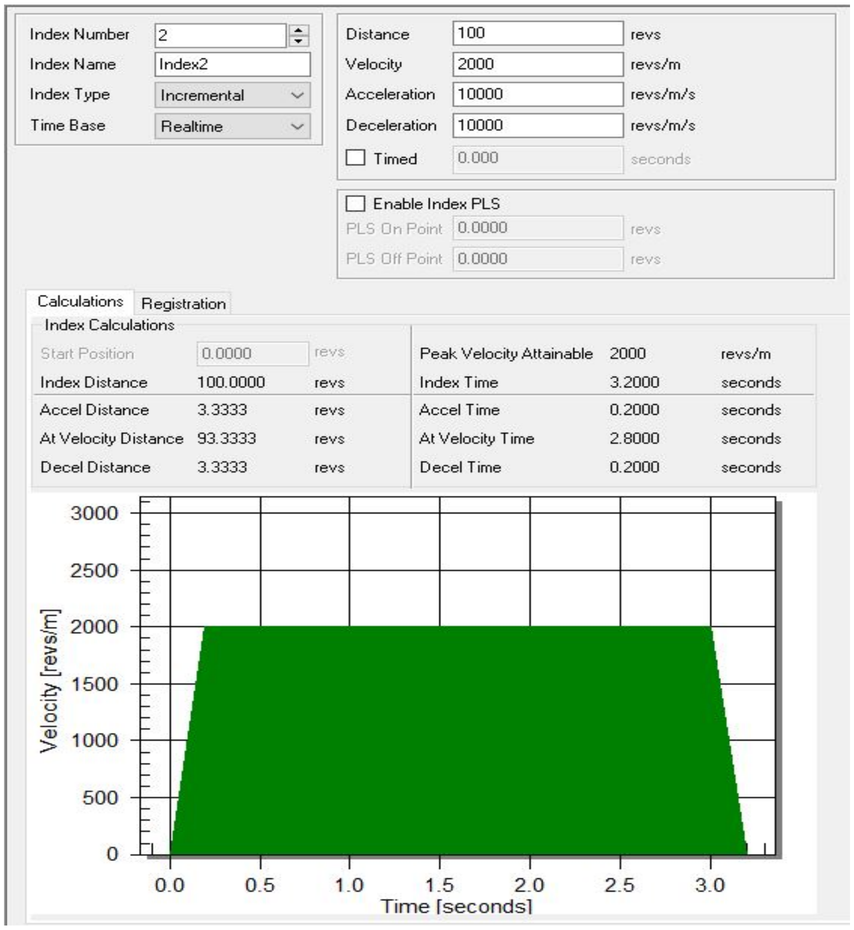
☐ Enable Index PLS
PLS On Point 0.0000 revs
PLS Off Point 0.0000 revs

Calculations | Registration

Index Calculations

Start Position	0.0000	revs	Peak Velocity Attainable	1000	revs/m
Index Distance	50.0000	revs	Index Time	3.2000	seconds
Accel Distance	1.6667	revs	Accel Time	0.2000	seconds
At Velocity Distance	46.6667	revs	At Velocity Time	2.8000	seconds
Decel Distance	1.6667	revs	Decel Time	0.2000	seconds

Figure 7 – Index 1 (1000 RPM)



Index Number	2		Distance	100	revs
Index Name	Index2		Velocity	2000	revs/m
Index Type	Incremental		Acceleration	10000	revs/m/s
Time Base	Realtime		Deceleration	10000	revs/m/s
			☐ Timed	0.000	seconds

☐ Enable Index PLS

PLS On Point	0.0000	revs
PLS Off Point	0.0000	revs

Calculations | Registration

Index Calculations

Start Position	0.0000	revs	Peak Velocity Attainable	2000	revs/m
Index Distance	100.0000	revs	Index Time	3.2000	seconds
Accel Distance	3.3333	revs	Accel Time	0.2000	seconds
At Velocity Distance	93.3333	revs	At Velocity Time	2.8000	seconds
Decel Distance	3.3333	revs	Decel Time	0.2000	seconds

Figure 8 – Index 2 (2000 RPM)

Now that the Index profiles are created we must arrange them in the program to get the desired compound profile.

Step 6 – Create the Program

Now we must create a user program to Compound the three Indexes together.

First to again demonstrate the difference between the Index.#.Initiate and Index.#.CompoundInitiate instructions, we will first create a program than runs the three indexes using the standard Index.#.Initiate instruction. Following that, we will initiate the same three Indexes using the Index.#.CompoundInitiate instruction.

Example NOT Using Compound Initiate

```
DriveIO.1.Out = OFF
DriveIO.2.Out = OFF
DriveIO.3.Out = OFF

Index.0.Initiate
Index.1.Initiate
Index.2.Initiate

End
```

If we initiate this program, the motion profile would look like that as shown in Figure 9 below.

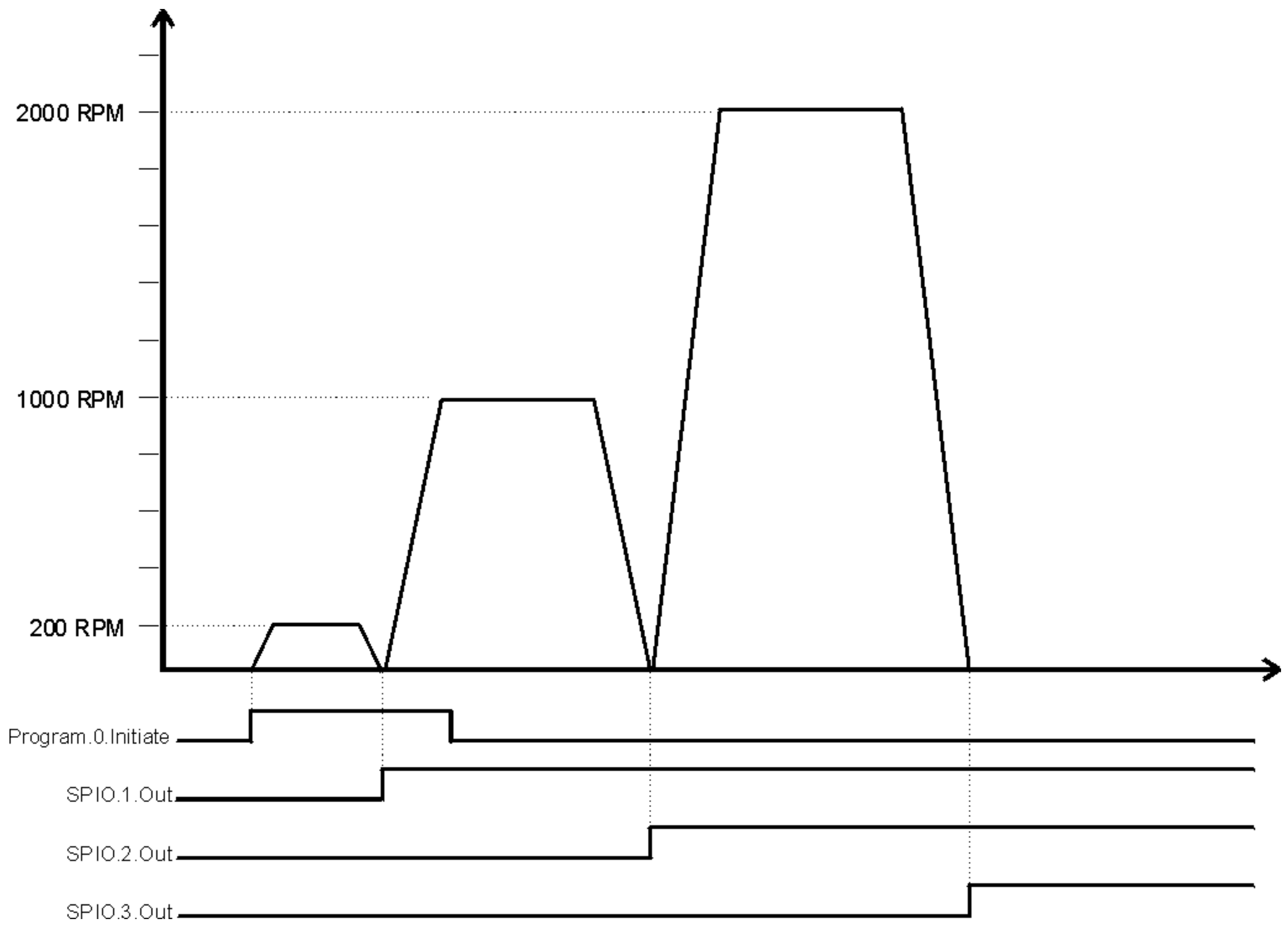


Figure 9 – Program NOT using Compound Initiate

Same Example this time WITH Compound Initiate

```
DriveIO.1.Out = OFF
DriveIO.2.Out = OFF
DriveIO.3.Out = OFF

Index.0.CompoundInitia
te
Index.1.CompoundInitia
te Index.2.Initiate
Wait                          For
Index.AnyCommandComplete End
```

Now if we initiate the program, the motion profile would look like that as shown in Figure 10 below.

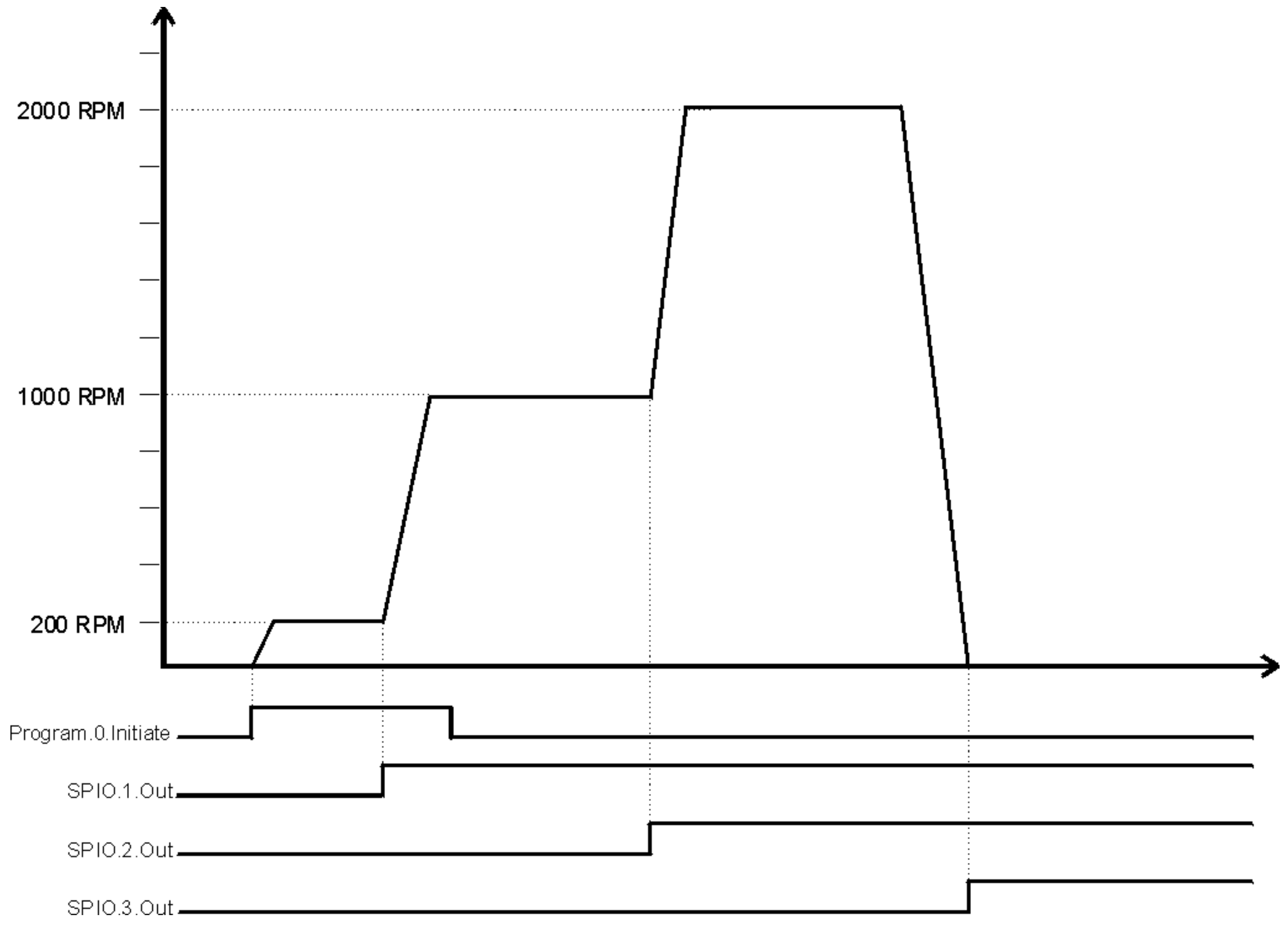


Figure 10 – Program WITH Compound Initiate

Program0 – Actual Program Code

```
'------------------------------------------------------------------
'Program 0                                                        -
'Description: Compound Indexing Example
- 'Filename(s): EZAT2.PTi                                          -
'Revision 1 – 3/22/2021, Created using PowerTools Studio 1.1
                                 - 'Min. Software Level Required:
                                 PowerTools Studio 1.1 or higher
                                 - 'Min. Firmware Required:
V01.01.00.20                     -
'------------------------------------------------------------------
'This program is initiated using DriveInput.4’
DriveIO.1.Out = OFF
DriveIO.2.Out = OFF
DriveIO.3.Out = OFF

Index.0.CompoundInitiate
Index.1.CompoundInitiate
Index.2.Initiate
Wait For Index.AnyCommandComplete
End
```

Description of Program Code Used

To initiate the program, simply activate DriveInput.4. See the Step 3 of this App Tool for further info on the Assignments.

The first instructions in the program are used to deactivate the three digital Outputs on the Unidrive M700 (if they happened to be on). The three outputs are just used to demonstrate functionality of the CommandComplete signal and are completely optional for this application.

The next instruction initiates Index 0 using the CompoundInitiate instruction. When the CompoundInitiate is used, that implies that the given index will end at the target velocity instead of stopping at zero velocity. The Index is initiated, and then the program proceeds to the next instruction in search of another index. Instructions can be inserted in-between the indexes, however, any instructions in-between must be processed before the first index is complete. If the first index completes its' distance before the next index in the program is found, then the motor will come to a stop using instantaneous deceleration (this will sound like a "clunk" in motion).

The next instruction of the program is Index.1.CompoundInitiate which means that Index 0 will transition directly into Index 1 without stopping, and Index 1 will also complete its distance at the target velocity (will not decelerate to zero speed). The program will suspend on this instruction until Index 0 is complete.

The next instruction is a standard Index Initiate. Therefore, Index 1 will transition into Index 2 without stopping, and then Index 2 will complete its distance by decelerating to a stop. The program will again suspend on this instruction until Index 1 is complete. The final index in a Compound string must always be a normal Index.#.Initiate instead of a CompoundInitiate.

The program then waits for Index 2 to be complete, and then the program ends. To repeat the program, simply activate DriveInput.4 again.

Important Notes

Following are a few important rules that apply when using Compound Indexing:

- The primary index always completes its specified distance and ends at the target velocity

It is also important to understand what ramps are used when transitioning from one index to another during Compound Indexes.

- The Acceleration of the secondary index is always used to reach the secondary indexes' target velocity (regardless of whether the secondary index is faster or slower than the primary index.)

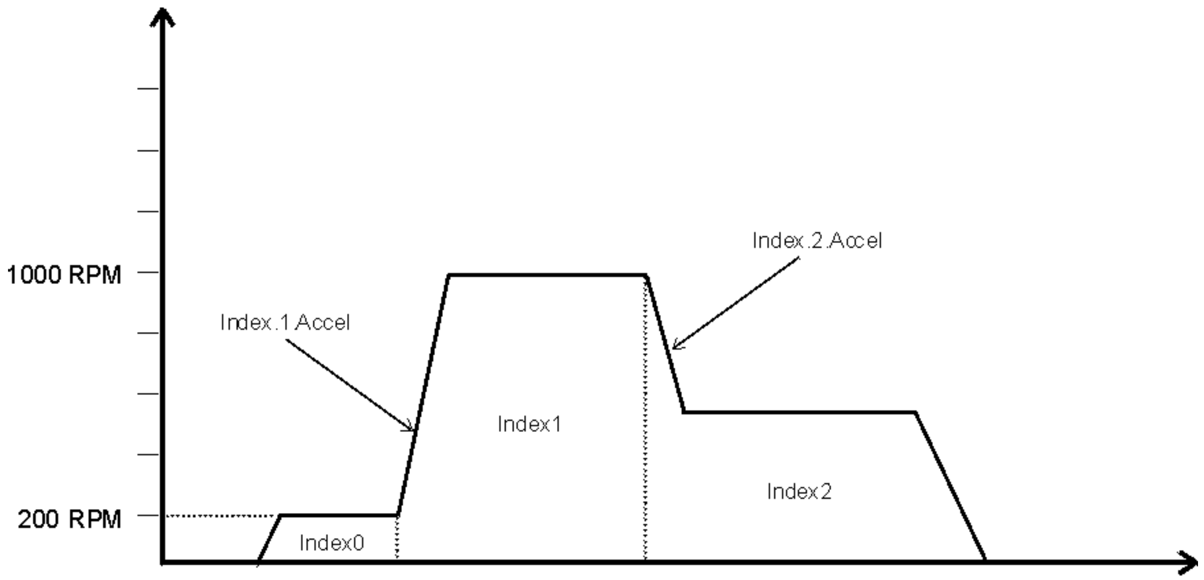


Figure 11 – Ramps Explanation for Compound Indexing

Important Notes - Continued

Because the ramps are always used when compounding from one index to another, if the ramps is not aggressive enough to reach the target velocity (or zero speed) in the specified index distance, the motor will decelerate at the specified ramp, and then back-up to the specified distance. Figure 12 below shows an example of this behavior.

Index 0 = 50 Revs @ 2000 RPM

Index 1 = 1 Rev @ 100 RPM, Acceleration = 5000 Revs/Min/sec

```
Index.0.CompoundI
nitiate
Index.1.Initiate
```

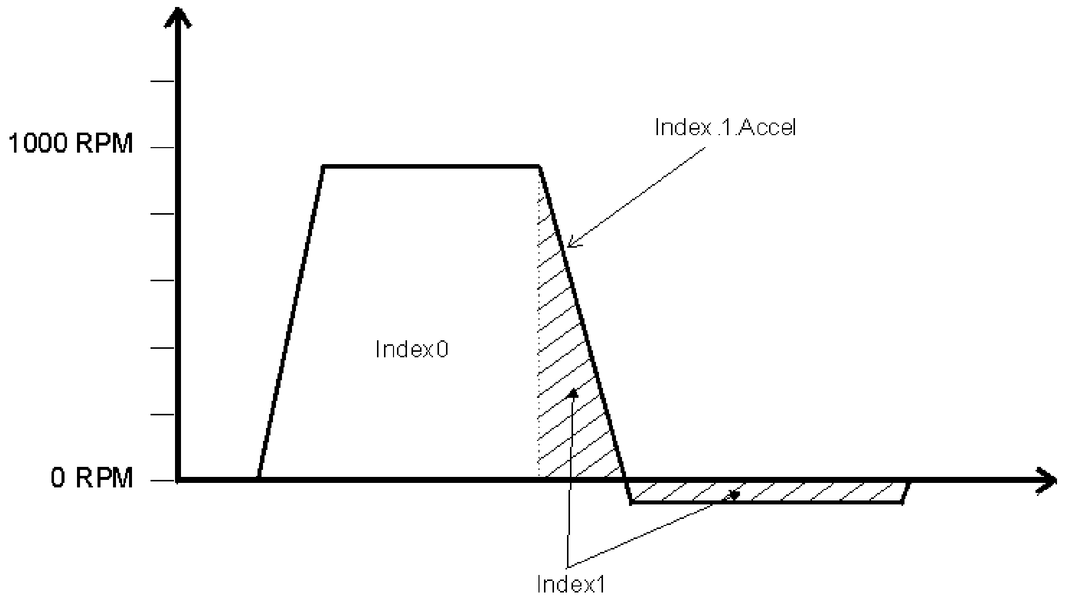


Figure 12 – Backup if Accel Ramp is not aggressive enough