User and Programming Guide

Motion Made Easy

PTi210 Module

Part Number: 0478-0616-07 Issue 7

Motion Made Easy
User and Programming Guide

CONTROL TECHNIQUES

Part Number: 0478-0616-07
Issue: 7
Date: August, 2024

Part Number: 0478-0616-07

Issue: 7

Date: August 2024

Version: V01.09.XX.XX

This document has been prepared to conform to the current released version of the product. Because of our extensive development efforts and our desire to further improve and enhance the product, inconsistencies may exist between the product and documentation in some instances. Call your customer support representative if you encounter an inconsistency.

Document Conventions

Manual conventions have been established to help you learn to use this manual quickly and easily. As much as possible, these conventions correspond to those found in other Microsoft® Windows® compatible software documentation.

Menu names and commands are printed in bold type: the **File** menu.

Dialog box names begin with uppercase letters and are printed in bold type: the **Axis Limits** dialog box.

Dialog box field names are in quotes: “Field Name”.

Button names are bold-italic: ***OK*** button.

Source code is printed in Courier font: `Case ERMS.`

In addition, you will find the following typographic conventions throughout this manual.

This	Represents
bold	Characters that you must type exactly as they appear. For example, if you are directed to type **a:setup**, you should type all the bold characters exactly as they are printed.
italic	Placeholders for information you must provide. For example, if you are directed to type *filename*, you should type the actual name for a file instead of the word shown in italic type.
ALL CAPITALS	Directory names, file names, key names, and acronyms.
SMALL CAPS	Non-printable ASCII control characters.
KEY1+KEY2 example: (Alt+F)	A plus sign (+) between key names means to press and hold down the first key while you press the second key.
KEY1,KEY2 example: (Alt,F)	A comma (,) between key names means to press and release the keys one after the other.

“Warning” indicates a potentially hazardous situation that, if not avoided, could result in death or serious injury.

“Caution” indicates a potentially hazardous situation that, if not avoided, may result in minor or moderate injury.

NOTE

For the purpose of this manual and product, “Note” indicates essential information about the product or the respective part of the manual.

Throughout this manual, the word “drive” refers to either the Unidrive M700/M701/M702 or the Digitax HD.

Safety Instructions

General Warning

Failure to follow safe installation guidelines can cause death or serious injury. The voltages used in the product can cause severe electric shock and/or burns and could be lethal. Extreme care is necessary at all times when working with or adjacent to the product. The installation must comply with all relevant safety legislation in the country of use.

Qualified Person

For the purpose of this manual and product, a “qualified person” is one who is familiar with the installation, construction and operation of the equipment and the hazards involved. In addition, this individual has the following qualifications:

- Is trained and authorized to energize, de-energize, clear and ground and tag circuits and equipment in accordance with established safety practices.
- Is trained in the proper care and use of protective equipment in accordance with established safety practices.
- Is trained in rendering first aid.

Reference Materials

The following related manuals and user guides may be useful with your particular system.

- *Unidrive M700/M701/M702 Control User Guide*
- *Unidrive M70X Parameter Reference Guide*
- *SI-I/O User Guide User Guide*
- *SI-I/O 24 Plus User Guide*
- *SI-DeviceNet User Guide*
- *SI-Profibus User Guide*
- *SI-Universal Encoder User Guide*
- *Digitax HD Control User Guide*
- *Digitax HD Parameter Reference Guide*

Table of Contents

1 Safety Information

The PTi210 module and its associated drive are intended as components for professional incorporation into complete equipment or systems. If installed incorrectly the drive may present a safety hazard. The drive uses high voltages and currents, carries a high level of stored electrical energy and is used to control mechanical equipment that can cause injury.

Close attention is required to the electrical installation, commissioning and maintenance must be carried out by personnel who have the necessary training and experience, They must read this safety information and User Guide carefully.

Careful consideration must be given to the functions of the drive and solutions module, which might result in a hazard, either through their intended functions e.g. auto-start or through incorrect operation due to a fault or trip e.g. stop/start, forward/reverse, maximum speed, loss of communications link.

In any application where a malfunction of the drive or solutions module could lead to damage, loss or injury, a risk analysis must be carried out and where necessary further measures taken to reduce the risk. To ensure mechanical safety additional safety devices such as electro-mechanical interlocks may be required. The drive must not be used in a safety critical application without high-integrity protection against hazards arising from a malfunction.

General Information

The manufacturer accepts no liability for any consequences resulting from inappropriate, negligent of incorrect installation or adjustment of the optional operation parameters of the equipment or from mismatching the variable speed drive (drive) with the motor.

The contents of this guide are believed to be correct at the time of printing. In the interests of a commitment to a policy of continuous development and improvement, the manufacturer reserves the right to change the specification of the product or its performance, of the contents of this guide, without notice.

Drive software version

This product is supplied with the latest version of user-interface and machine control software. If this product is to be used on a new or existing system with other drives, there may be some differences between their software and the software in this product. These differences may cause this product to function differently. This may also apply to drives returned from a Control Techniques Service Centre. If there is any doubt, contact a Control Techniques Drive Centre.

Safety of Machinery

Within the European Union all machinery in which this product is used must comply with Directive 89/392/EEC, Safety of Machinery.

The product has been designed and tested to a high standard, and failures are very unlikely. However the level of integrity offered by the product's control function – for example stop/start, forward/reverse and maximum speed – is not sufficient for use in safety-critical applications without additional independent channels of protection. All applications where malfunction could cause injury or loss of life must be subject to a risk assessment, and further protection provided where needed.

General warning

Failure to follow safe installation guidelines can cause death or serious injury. The voltages used in this unit can cause severe electric shock and/or burns, and could be lethal. Extreme care is necessary at all times when working with or adjacent to this equipment. The installation must comply with all relevant safety legislation in the country of use.

AC supply isolation device

The AC supply must be removed from the drive using an approved isolation device or disconnect before any servicing work is performed, other than adjustments to the settings or parameters specified in the manual. The drive contains capacitors which remain charged to a potentially lethal voltage after the supply has been removed. Allow at least 10 minutes after removing the supply before carrying out any work which may involve contact with electrical connections to the drive.

Products connected by plug and socket

A special hazard may exist where the drive is incorporated into a product which is connected to the AC supply by a plug and socket. When unplugged, the pins of the plug may be connected to the drive input, which is only separated from the charge stored in the bus capacitor by semiconductor devices. To avoid any possibility of electric shock from the pins, if they are accessible, a means must be provided for automatically disconnecting the plug from the drive (e.g., a latching contactor).

Grounding (Earthing, equipotential bonding)

The drive must be grounded by a conductor sufficient to carry all possible fault current in the event of a fault. The ground connections shown in the manual must be followed.

Fuses

Fuses protection must be provided at the input in accordance with the instructions in the manual.

Isolation of control circuits

The installer must ensure that the external control circuits are isolated from human contact by at least one layer of insulation rated for use at the applied AC supply voltage.

2 Introduction

Modern variable speed drives such as the Unidrive M/Digitax HD offer a multitude of built-in features such as ramp control, speed control, PID Loops, and even simple position control. However, the drive's functionality may not be suitable for some high performance motion applications. When it comes to more complex applications, the user must resort to using external equipment such as PLCs to control the drive from a system point of view.

However, the flexibility of the Unidrive M/Digitax HD can be substantially increased by using a PTi210 option module. A PTi210 option module is a 1½ axis controller with a dedicated processor that allows the user to write their own application specific software.
The Unidrive M/Digitax HD drive also offers powerful networking capabilities in addition to the PTi210 so that many drives (and other equipment) can be linked together to communicate process wide information thus offering a complete application solution.

2.1 Installing the PTi210 Module in a Unidrive M

The PTi210 module is an option module that can be fitted to any one of the three expansion slots in the Unidrive M. Figure 2-1 shows the three slot positions.

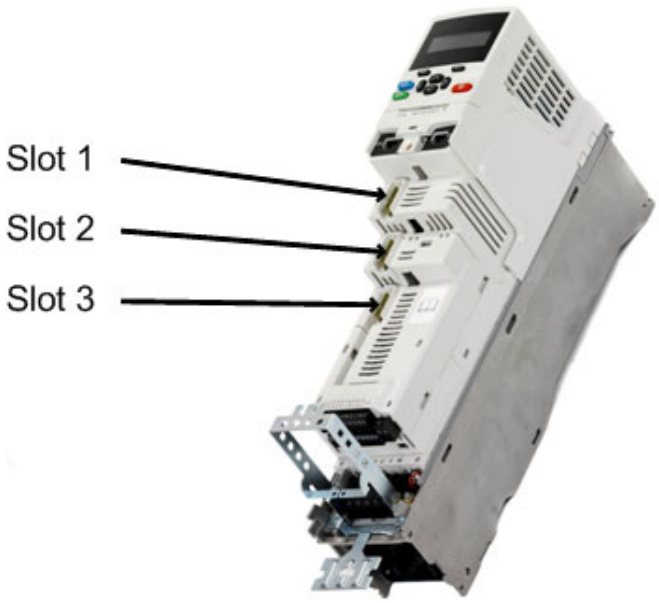


Figure 2-1: Unidrive M Slot Diagram

The PTi210 module is powered from the Unidrive M internal power supply.

When using PowerTools Studio to program the PTi210 module, the user must indicate which options slot the module is fitted in. By default, PowerTools Studio will select Slot 3 for the PTi210 module.

2.2 Installing the PTi210 Module in a Digitax HD

The PTi210 Module is an option module that can be fitted to any one of the two expansion slots in the Digitax HD, see Figure 2-2. The PTi210 module is powered from the Digitax HD internal power supply.

By default, PowerTools Studio will select slot 1 for the PTi210 module.

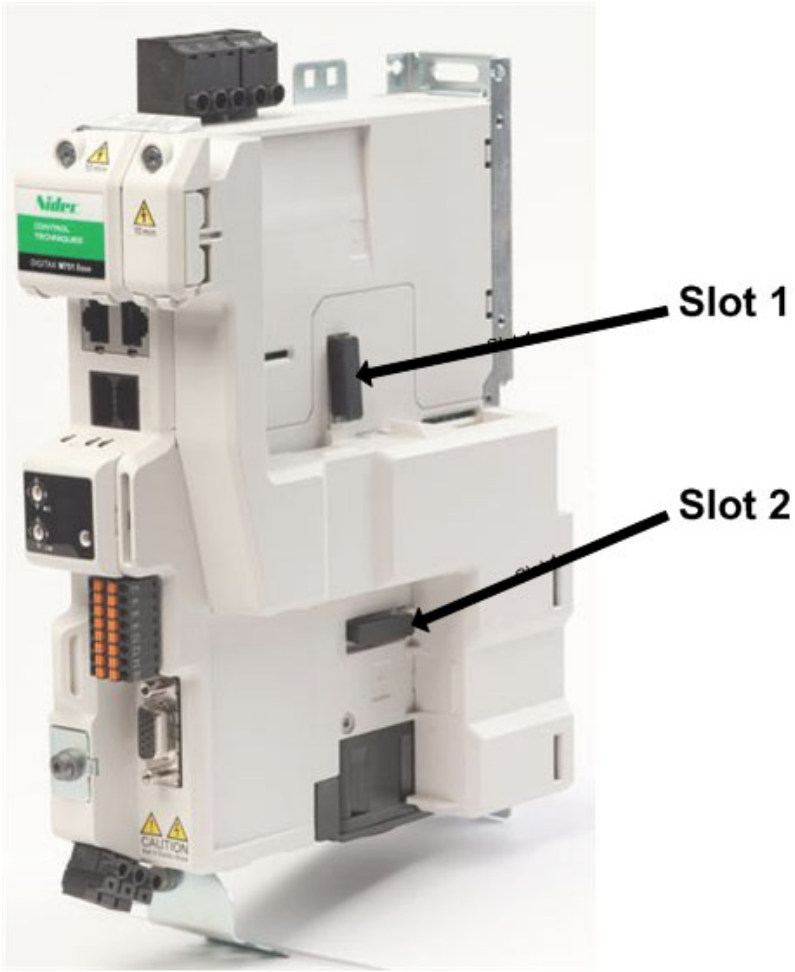


Figure 2-2: Digitax HD Slot Diagram

2.3 Development Software

Applications for the PTi210 module are developed by the user using PowerTools Studio software. PowerTools Studio is an easy-to use, Windows® based setup and diagnostics tool. It provides you with the ability to create, edit and maintain your system setup. You can download or upload your setup data to or from a device. You can also save it to a file on your PC or print it for review or permanent storage.

PowerTools Studio is designed to be the easiest-to-use software available for the 1½ axis motion controllers.

PowerTools Studio will run on Windows® 7, Windows® 8, and Windows® 10, and Windows® 11 operating systems.

2.3.1 Features

- "Hierarchy Tree" for quick navigation to any setup view
- Simple I/O function assignments
- Powerful online diagnostic capabilities
- Fill-in-the-blank motion profile parameters
- Programming

2.3.2 Minimum drive firmware version

To fully utilise all the features available within PowerTools Studio, it is recommended that the minimum drive firmware version V01.21.00.00 is installed in the Unidrive M/Digitax HD drive.

2.4 System Integration Module Compatibility Chart

The table below shows what System Integration option modules are compatible with the PTi210 and whether or not they are configurable within PowerTools Studio.

Module		Category	Configured by PowerTools Studio
Name	ID		
SI-Encoder	105	Position Feedback	No[1]
SI-Universal Encoder	106	Position Feedback	Yes
SI-I/O 24 Plus	108	I/O Expansion and Position Feedback	Yes
SI-I/O	209	I/O Expansion	Yes
SI-Applications Plus	304	Applications	No[1]
SI-Applications Compact	304	Applications	No[1]
MCi210	310	Applications	No[1]
MCi200	311	Applications	No[1]
PTi210	320	Applications	Yes
SI-Interbus 500 kBd	404	Fieldbus	No
SI-Interbus 2 MBd	414	Fieldbus	No
Factory Fit Ethernet (V2) (Unidrive M)	430	Fieldbus	Yes[2,5]
SI-EtherCAT	431	Fieldbus	No[1,6]
SI-PROFINET RT	432	Fieldbus	No[1]
SI-Ethernet	433	Fieldbus	No[1,4]
SI-PROFINET V2	434	Fieldbus	No[1]
Factory Fit EtherCAT (Digitax HD M753)	435	Fieldbus	No[1]
SI-POWERLINK	436	Fieldbus	No[1,6]
Factory Fit Ethernet (Digitax HD M750)	437	Fieldbus	Yes[2]
SI-BACnet IP	439	Fieldbus	No[1]
SI-PROFIBUS	443	Fieldbus	Yes[3]
SI-DeviceNet	447	Fieldbus	Yes[3]
SI-CANopen	448	Fieldbus	No[1]

[1] – No setup screen is provided by PowerTools Studio to configure the option module but the option module parameters are accessible from within the PTi210 user program.

[2] – Indicates that a setup screen is provided to allow the cyclic I/O mappings to be configured using RTMoE, the RTMoE mappings will be configured using the Easy Mode parameters in menu 10 of the options module. (The Easy Mode links will be overridden by the Advanced Cyclic Links when downloaded from Machine Control Studio).

[3] – Indicates that a setup screen is provided to allow the cyclic I/O mappings to be configured.

[4] – PowerTools Studio will not communicate with the drive via the SI-Ethernet module.

[5] – Minimum firmware version V02.07.00.00.

[6] – These modules are designed to control the drive operation and, if fitted with a PTi210 module, will prevent the PTi210 from controlling the drive properly. It is recommended in these circumstances to disable the drive control for these modules by setting Pr **1M.033** = On.

2.5 Parameter Reference Convention

Throughout this document the following parameter reference conventions are followed:

- **mm.ppp**, where **mm** represents the drive menu number and **ppp** represents the parameter number within the menu
- **1M**.**ppp**, where **1M** represents the option module main setup menu as shown in the following table:
- **2M.ppp**, where **2M** represents the option module secondary setup menu as shown in the following table:

Slot #	Main Setup Menu (1M.ppp)	Secondary Setup Menu[2] (2M.ppp)
1	15	25
2	16	26
3[1]	17	27

[1] – Not available on Digitax HD.

[2] – Only available on certain option modules (e.g. SI-I/O 24 Plus)

3 Installation

This section of the manual will cover basic installation information.

Before installing or removing a System Integration option module in any drive, ensure the AC supply has been disconnected for at lease 10 minutes and refer to *Safety Information* on page 1. If using a DC bus supply ensure this is fully discharged before working on any drive or System Integration option module.

3.1 Mechanical Installation

Please refer to the Installation Sheet that comes with the PTi210 module for details on installing the module into the relevant drive.

3.2 Slot Selection

The PTi210 module may be placed in any available option slot on the Unidrive M/Digitax HD. The user must indicate which slot the PTi210 module is fitted in using PowerTools Studio configuration software. The default slot number for Unidrive M70X is Slot 3 and for Digitax HD is Slot 1 in the configuration software.

3.3 Electrical Connections

The PTi210 module has a single terminal block allowing screwless terminal access to the digital I/O.

The terminals are numbered from Terminal 1 to 3 on the upper row (left to right) and Terminals 4 to 6 on the bottom row (left to right). The different terminal functions are listed in the table below.

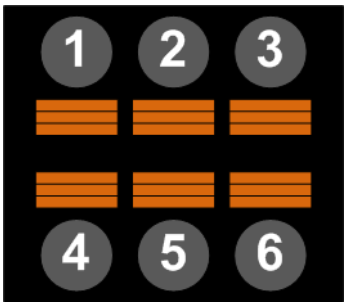


Figure 3-1: PTi210 Digital I/O Connections

Terminal #	Function	Description
1	DI1	Digital Input 1 (High Speed Capture)
2	DI2	Digital Input 2 (High Speed Capture)
3	DI3	Digital Input 3
4	DO1	Digital Output 1
5	DO2	Digital Output 2
6	0 V Common	0 V Common

3.4 Digital I/O Connections

The PTi210 module is equipped with 3 digital inputs, two of which are designed for high speed capture inputs and 2 digital outputs. The I/O are electrically sourcing I/O. All I/O utilize positive logic meaning that they are active when a positive voltage is applied (15 Vdc to 30 Vdc).

The digital I/O can be used to control different functions in the PTi210 module. The digital I/O are updated at the Trajectory Update Rate. The Trajectory Update Rate can be found on the Setup view in PowerTools Studio (see *Setup View* on page 83 for more information on the Trajectory Update Rate).

The first two digital inputs (Digital Input 1 and Digital Input 2) of the PTi210 module are also unique (as compared to Unidrive M/ Digitax HD digital I/O and SI-I/O) because they can be used in the high speed capture process. Even though they are only updated once every Trajectory Update, the PTi210 module processor knows when they activate to within 1 microsecond. Therefore, when Capture is used, they can be accurate to 1 microsecond (see *Capture View* on page 103 for more information on the Capture object).

Figure 3-2 below shows a wiring diagram for the digital I/O on the PTi210 module.

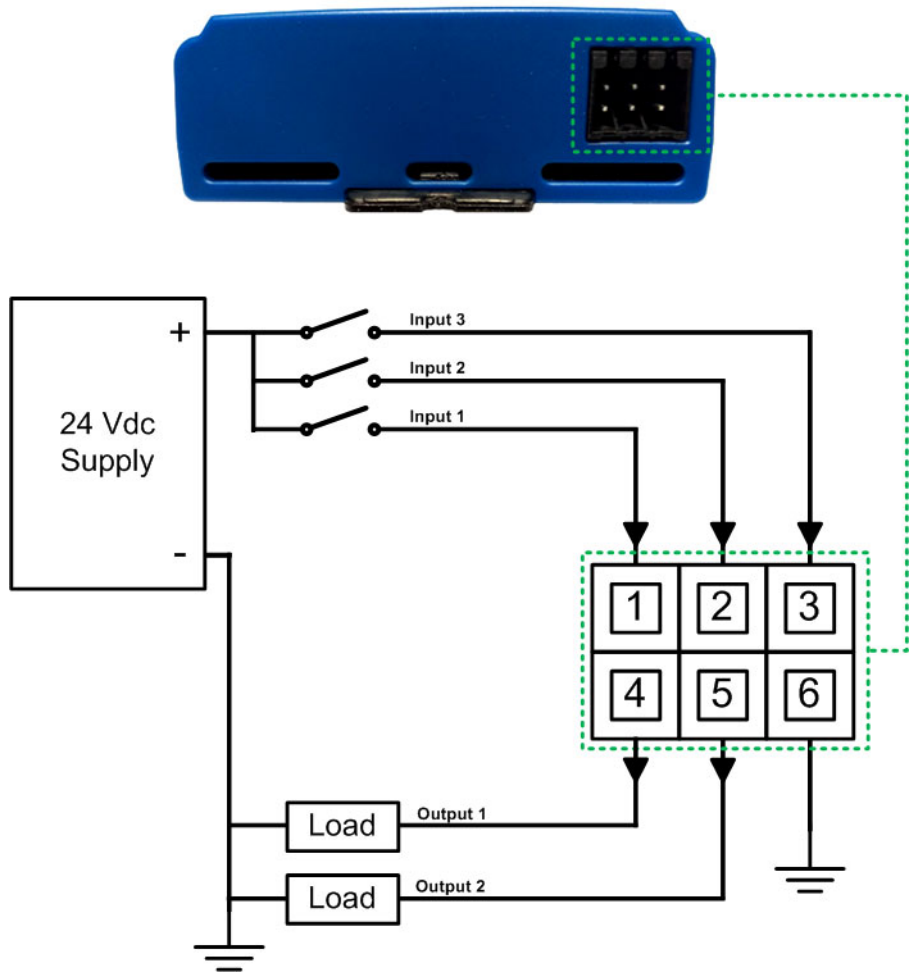


Figure 3-2: PTi210 I/O Wiring Diagram

3.5 Digital I/O Specifications

I/O Type:	Sourcing
Input Turn on Voltage:	15 Vdc +/-0.5 Vdc
Input Voltage Range:	0 Vdc to +24 Vdc
Max Input Voltage:	+/- 30 Vdc
Output Voltage:	Depends on the 24 Vdc Supply
Max Output Current:	20 mA Total for both Outputs

WARNING

The digital I/O circuits are isolated from the power circuits in the drive by basic insulation (single insulation) only. The installer must ensure that the external control circuits are insulated from human contact by at least one layer of insulation (supplementary insulation) rated for use at the AC supply voltage.

WARNING

If the digital inputs or outputs are to be connected to other circuits classified as Safety Extra Low Voltage (SELV) (e.g. to a personal computer), an additional isolating barrier must be included in order to maintain the SELV classification.

3.6 Motor Encoder Feedback Connections

The following table can be used to connect the encoder feedback signals for various different encoder types to the Unidrive M or Digitax HD. For further installation information, please refer to the relevant drive documentation.

Encoder Type Pr 03.038	Connections														
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
AB (0)	A	A\	B	B\	Z	Z\							+V	0V	Th
FD (1)[1]	F	F\	D	D\	Z	Z\									
FR (2)[1]	F	F\	R	R\	Z	Z\									
AB Servo (3)	A	A\	B	B\	Z	Z\	U	U\	V	V\	W	W\			
FD Servo (4)[1]	F	F\	D	D\	Z	Z\	U	U\	V	V\	W	W\			
FR Servo (5)[1]	F	F\	R	R\	Z	Z\	U	U\	V	V\	W	W\			
SC (6)	A (Cos)	A\ (Cos)	B (Sin)	B\ (Sin)	Z	Z\									
SC Hiperface (7)	Cos	Cosref	Sin	Sinref	DATA	DATA\									
EnDat (8)	DATA	DATA\	CLK	CLK\	Freeze	Freeze\									
SC EnDat (9)	A	A\	B	B\	DATA	DATA\					CLK	CLK\			
SSI (10)	DATA	DATA\	CLK	CLK\	Freeze	Freeze\									
SC SSI (11)	A (Cos)	A\ (Cos)	B (Sin)	B\ (Sin)	DATA	DATA\					CLK	CLK\			
SC Servo (12)	A (Cos)	A\ (Cos\)	B (Sin)	B\ (Sin\)	Z	Z\	U	U\	V	V\	W	W\			
BiSS (13)	DATA	DATA\	CLK	CLK\	Freeze	Freeze\									
Resolver (14)	Cos H	Cos L	Sin H	Sin L	Ref H	Ref L									
SC SC (15)	A (Cos)	A\ (Cos)	B (Sin)	B\ (Sin)	Z	Z\	C*1	C*1	D*2	D*2	Freeze2	Freeze2\			
Commutation Only (16)							U	U\	V	V\	W	W\			
SC BiSS (17)	A	A\	B	B\	DATA	DATA\					CLK	CLK\			
EnDat Alt (18)					DATA	DATA\					CLK	CLK\			
SSI Alt (19)					DATA	DATA\					CLK	CLK\			
BiSS Alt (20)					DATA	DATA\					CLK	CLK\			

[1] - Currently not supported in PowerTools Studio

3.7 Simple Servo Motor Phasing Test

When connecting a non-standard servo motor to the Unidrive M or Digitax HD, it is necessary to know the wiring configuration of the motor. At times, all of the necessary wiring documentation for connecting the motor is not readily available from the motor manufacturer. In that case, it may be possible to follow the simple servo motor phasing test described below. This will help to determine if the motor phases (U, V, and W) are wired correctly along with the encoder commutation and channel signals. If the procedure described below is followed, and you still have problems, please refer to the Unidrive M or Digitax HD *User Guide* for further wiring information.

Begin by entering the motor peak current, continuous current, number of poles, encoder lines per rev., etc. Then follow the steps below.

Step 1: Verify wiring of encoder channels per the documentation. Define CW rotation of the motor shaft, from the flange side, with increasing counts.

To verify this, do the following:

A Disable the drive

B Navigate the keypad to display parameter Pr **03.029**

C Turn the shaft clockwise and verify that the encoder counts increase from 0 to 65535

D If the counts decrease, the encoder A and B channels need to be swapped

E Repeat A through C of Step 1.

Step 2: Verify wiring of motor power cables. Define CW rotation of the motor with a positive drive command. The phasing test of the drive will give a CW rotation during the test.

To verify this, complete the following steps:

A Enable the drive

B Verify that the motor is free of any load

C Navigate the keypad to display parameter Pr **05.012**

D Set the parameter to 1. The phasing test will command the motor to move one rev CW. It will also reset the parameter to 0

E If the motor moves in the CCW direction, the motor power is wired incorrectly

F Swap the U and V phases and repeat A through D of Step 2.

NOTE

Disregard any encoder phasing trip at this time **[Autotune X trips]**.

Step 3: Verify wiring of commutation signals. The Unidrive M/Digitax HD *Control User Guide,* section 12 (Pr **03.025**) is helpful for this step.

If no trips were encountered during Step 2, this step can be skipped.

To verify correct commutation, follow the steps below:

A Enable the drive

B Navigate the keypad to display parameter Pr **05.012** and set the parameter to 1

C If an **Autotune X** trip results, rewiring is needed

D Swap the U and V commutation signals at the drive end.

Repeat steps A through C to verify.

4 PowerTools Studio Software

4.1 Introduction

PowerTools Studio is the software used to configure hardware type, setup parameters, I/O functionality, motion profiles, user programs, and networks for the PTi210 module. PowerTools Studio software operates on a PC running Windows® 7, 8, or 10 and can also be used as a diagnostic tool and troubleshooting assistance.

4.2 Installing PowerTools Studio

PowerTools Studio can be downloaded from the Control Techniques website at the following address: www.controltechniques.com.

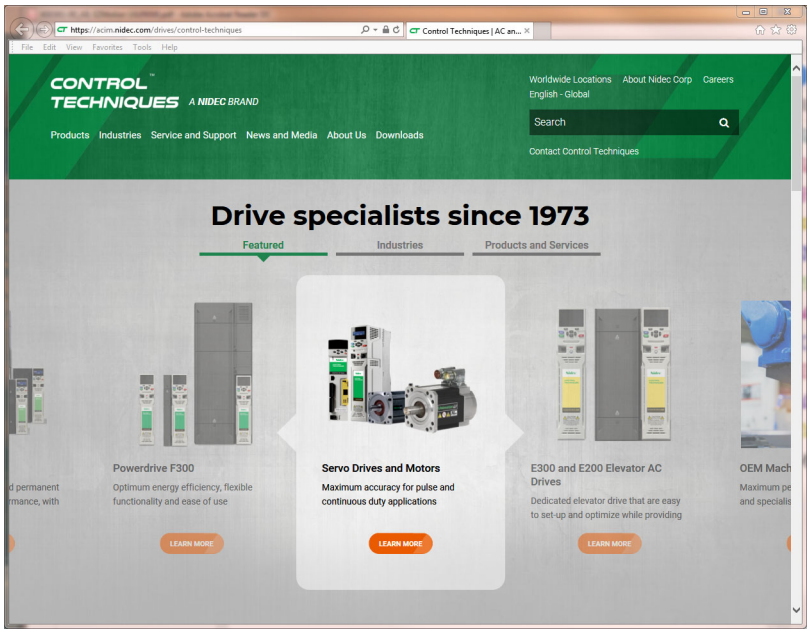


Figure 4-1: Control Techniques Website

4.3 How PowerTools Studio is Organized

The PowerTools Studio software is made up of six major components. These components are the Menu Bar, Tool Bar, Hierarchy Tree, View, View Tab, and Status Bar. Note that some of these components and sub-components are only available under certain conditions (i.e., while online, while on a certain view, etc.).

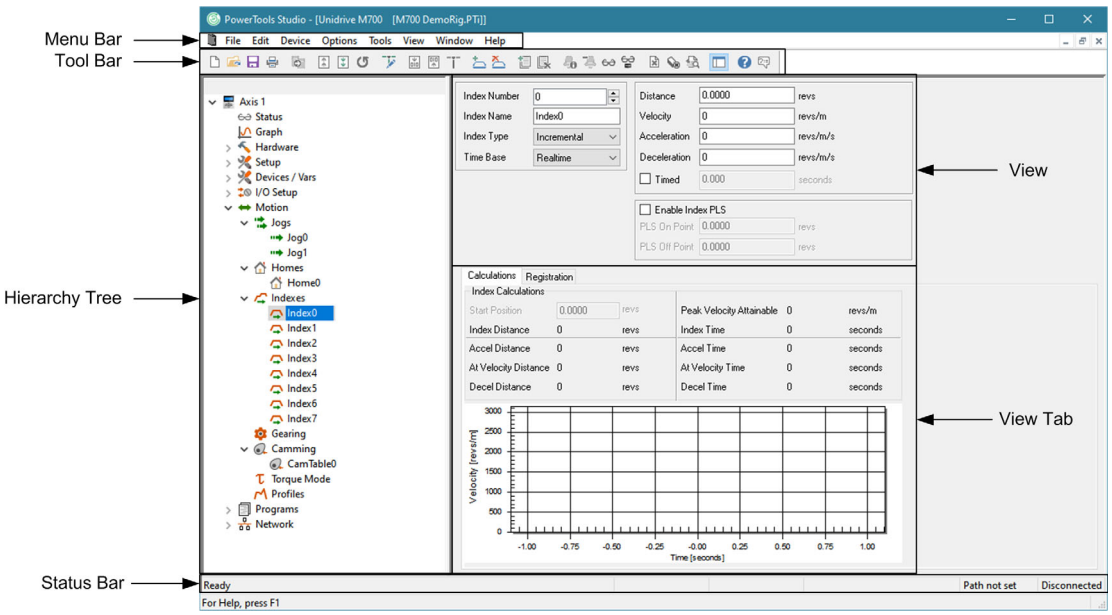


Figure 4-2: PowerTools Studio Organization

4.4 Menu Bar

Figure 4-3 shows the Menu Bar as found in PowerTools Studio. The items available on the Menu Bar may change under certain conditions (i.e., online, configuration open, etc.).

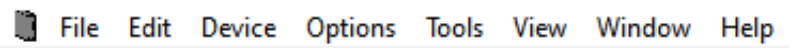


Figure 4-3: Menu Bar

To use a menu choose one of the following methods:

- On the menu bar, click a menu name to display a list of commands. On the menu, either click a command or use the DOWN ARROW to move down the list, and then press ENTER.
- Press ALT and press the underlined letter in the menu name. Then press the underlined letter in the option name. For example, to open a new configuration file, press ALT and press F to open the File menu. Then press N for New, see Figure 4-4.

4.4.1 File

The **File** menu on the menu bar contains many different options for file handling (i.e., opening, closing, saving, and printing files, etc.). Figure 4-4 below shows the **File** menu expanded.

Some of the **File** menu options are only shown when an application file is currently loaded, so if no application file is loaded then the relevant menu options will not be shown.

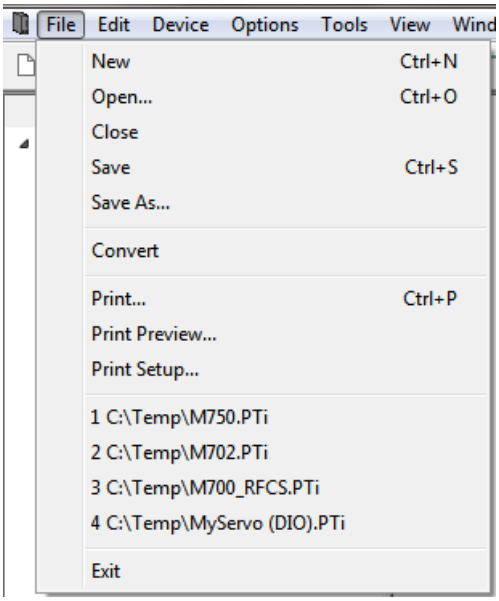


Figure 4-4: File Menu

New

New will open a new PowerTools Studio file. The user will be asked what type of configuration to create Unidrive M or Digitax HD.

Open

Open allows the user to open an existing application created with PowerTools Studio. Navigate to the directory that the desired file is located in, and double-click on the specific file. Doing so will open the file for editing.

Close

Close will close the active configuration. If multiple files are open, the active file's Title Bar will be highlighted.

Save

Selecting **Save** will save the active file on the users PC. The location to which the file is saved is based on where the file was previously saved. If the file has not yet been saved, the **Save As** dialog box will open instead.

Save As...

Save As allows the user to save the active file using a different name or to save an existing file to a different directory location. Navigate to the directory to which the file is to be saved, and click ***Save***.

Convert

Convert allows the user to convert an existing file created in an earlier version of PowerTools Studio. When using the **File** > **Open** feature, PowerTools Studio will open the existing file at the same interface revision at which it was created. This means that not all of the latest features currently supported by PowerTools Studio may be available within the application file. If, after opening an existing file, a user wishes to upgrade the file to the latest interface revision so that all the latest features are available, it can be done by selecting **Convert**.

Upgrading to the latest revision could possibly change some of the operation of the module, so it is highly recommended to save the file prior to upgrading it, in the event that one needs to revert to the original file. Once the file is upgraded, the user can then save the new file with a different name, and then download that file if they so choose.

When opening a new file, **Convert** will be unavailable because a new file is always created at the latest interface revision.

Print

Print will send the active file to the printer specified by the user. The **Print Options** dialog box will open allowing the user to specify which sections of the configuration are to be printed. By default, all sections will be printed. To remove a given section from the printout, clear the specific check box.

Print Preview

Print Preview will open a new window that displays what an actual hardcopy printout would look like. This can be helpful to determine if formatting is correct.

Print Setup

Selecting **Print Setup** allows the user to change the Target Printer, Paper Type, Paper Source, Print Orientation, and other printer related parameters.

Recently Used Files

Also displayed on the **File** menu are the last four files that were edited using PowerTools Studio. To quickly access one of these last four files, simply click on the file name in the **File** menu. Clicking on one of these files will open the configuration for editing.

4.4.2 Edit

Figure 4-5 shows the **Edit** menu as selected from the PowerTools Studio Menu Bar.

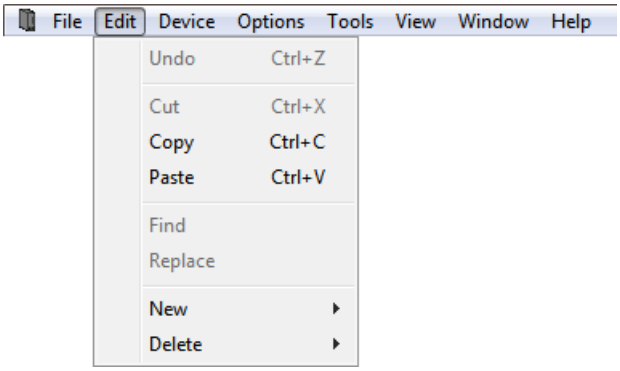


Figure 4-5: Edit Menu

Undo

Selecting **Undo** will undo the last change made to a user program. Up to the last ten changes made can be undone.

Cut

Selecting **Cut** will remove the selected text from a user program. To select text in a program, place the mouse pointer at the leftmost character to be selected, then press and hold the left mouse button dragging the cursor over the text until the mouse pointer is positioned over the final desired character, then release the mouse button. Once the text is selected, the text can be cut, copied, or pasted.

Copy

Selecting **Copy** will copy any selected text in a user program. To select text in a program, place the mouse pointer at the leftmost character to be selected, then press and hold the left mouse button dragging the cursor over the text until the mouse pointer is positioned over the final desired character, then release the mouse button. Once the text is selected, the text can be cut, copied, or pasted.

Paste

Selecting **Paste** will place the last cut or copied text into a user program. See Cut and Copy above for further information.

Find

Selecting **Find** will open the Find window. In the Find window, the user can type in a specific word, number, or any character that they wish to find in a user program. Once the user enters the text they wish to find, the ***Find Next*** or ***Mark All*** button is clicked. The ***Find Next*** button will highlight the next segment of code after the cursor that matches the search text. The ***Mark All*** button will put a mark next to each line of the program that has matching text. The user also has several other options on searching the program for matching text.

Replace

Selecting **Replace** will open the Find window (see **Find** above) with an additional parameter called Replace With. Using this method will search the user program for text that matches the text in the Find What text box, and replace it with the text in the Replace With text box. The user can select to replace just the next match, or all existing matches with the ***Replace All*** button.

New

Selecting **New** will open a sub menu allowing the user to add a new Index, Home, Program, or CAM Table. Indexes, Homes, Programs, and CAM Tables may not be added while online with the PTi210 module.

Index

Selecting **Edit** > **New** > **Index** will add a new index to the configuration. Indexes are added in sequential order. The new index will be the next highest available index number. Adding an index will take you directly to the new index view. A maximum number of 100 Indexes can be configured.

Home

Only one home is available in the initial release of the PTi210 module.

Program

Selecting **Edit** > **New** > **Program** will add a new user program to the configuration. Programs are added in sequential order. The new program will be the next highest available program number. Adding a program will take you directly to the new program view. A maximum number of 100 Programs can be configured.

CAM Tables

Selecting **New** > **Edit** > **CAM Tables** will add a new CAM Table to the configuration. CAM Tables are added in sequential order. The new CAM Table will be the next highest available number. A maximum number of 32 CAM Tables can be configured.

Delete

Selecting **Edit** > **Delete** will open a sub menu allowing the user to delete an existing Index, Home, Program, or CAM Table. Indexes, Homes, Programs, and CAM Tables may not be deleted while online with the PTi210 module.

Index

To delete an index, the user must select the specific index they wish to delete on the hierarchy tree. Once the index is selected, click **Edit** > **Delete** > **Index** on the Menu Bar. Doing so will delete the index instance. Once the index is deleted, the data stored on the index view cannot be recovered.

Home

Homes cannot be deleted in the initial release of the PTi210 module.

Program

To delete a program, the user must select the specific program they wish to delete on the hierarchy tree. Once the program is selected, click on **Edit** > **Delete** > **Program** on the Menu Bar. Doing so will delete the program instance. Once the program is deleted, the program code cannot be recovered.

CAM Tables

To delete a CAM Table, the user must select the specific CAM Table they wish to delete on the hierarchy tree. Once the CAM Table is selected, click **Edit** > **Delete** > **CAM Table** on the Menu Bar. Doing so will delete the CAM Table instance. Once the CAM Table is deleted, the data stored on the Camming view cannot be recovered.

4.4.3 Device

Figure 4-6 shows the **Device** menu selected from the PowerTools Studio Menu Bar.

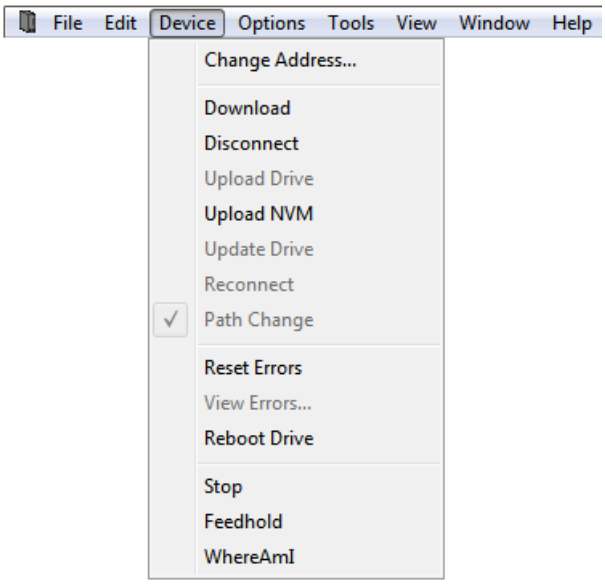


Figure 4-6: Device Menu

Change Address

On the **Device** menu, choose **Change Address**. This allows the user to change the Modbus node address of a drive/module system. The user must be online with the device for **Change Address** to operate correctly. The user will be prompted for the new node address. After entering the new node address, click ***OK***. The drive will immediately change to the new address.

Download

Download will send the active configuration from the PC to the target node address (specified on the Setup view). For more information on Downloading, see section 5 *Communications* in this manual.

Disconnect

Selecting **Disconnect** will terminate communications between the PC and any nodes the PC is online with.

Upload Drive

Upload Drive will upload only the node address specified in the active configuration. To use **Upload Drive**, open a new file and set the Node Address on the Setup view to the address you wish to upload. Then on the **Device** menu, choose **Upload Drive**. This will overwrite the active configuration with the uploaded data.

Upload NVM

Selecting **Upload NVM** will read the current value from each of the parameters in the PTi210's NVM memory and display it in the PowerTools Studio configuration. The file can then be saved to retain the current values stored in NVM.

Update Drive

Selecting **Update Drive** will send any parameters that have been changed since the last download into RAM. Doing so allows the user to send changes to the system without requiring a complete download. Certain parameters when changed require a complete download and cannot be sent to RAM. If one of these parameters is changed, **Update Drive** will be unavailable on the **Edit** menu.

Reconnect

Selecting **Reconnect** from the **Device** menu will reconnect communications from the target node to the PC. Reconnect allows PowerTools Studio to go online without needing to perform a full upload or download.

In order for reconnect to work, the active file in PowerTools Studio must be exactly the same as the file that resides in the PTi210's memory. If the files are not the same, the reconnect will fail.

Path Change

Selecting **Path Change** from the **Device** menu will open the Set Path Dialog box, the Set Path Dialog box allows the user to scan the network for devices (default) or manually set the protocol and connection details to download or upload.

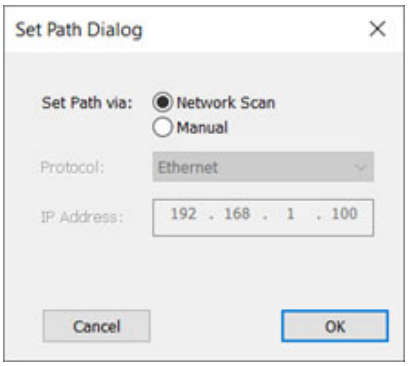


Figure 4-7: Set Path Dialog

When the Manual option is selected, the user can then select either the RTU (Serial) or Ethernet protocol, for the RTU (Serial) setting, the Modbus ID number (device node address) and Com Port can be changed.

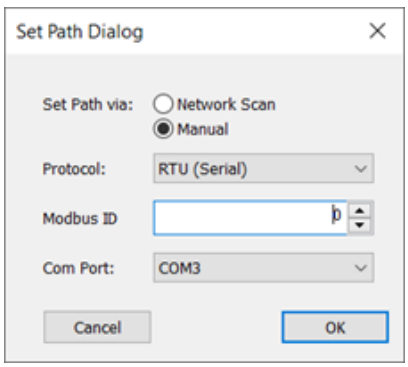


Figure 4-8: Manual RTU (Serial) Connection

For Ethernet, only the IP Address can be configured

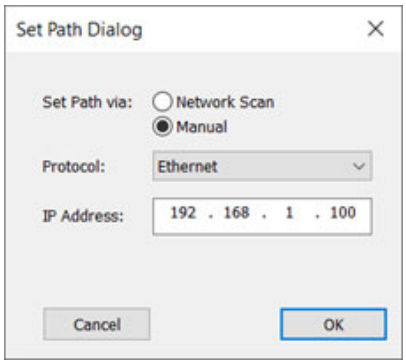


Figure 4-9: Manual Ethernet Connection

If the connection path is configured manually, then the user must ensure the correct details are configured, otherwise the connection may not be made to the drive.

NOTE

The network scan is automatically selected by default whenever the communication path is changed from the menu or from the toolbar.

Reset Errors

Selecting **Reset Errors** will clear any active Errors or Trips. If the trip condition still exists, the trip may reactivate immediately after clearing it.

View Errors

On the **Edit** menu, choose **View Errors** to open the Active Errors pop up window. The Active Errors window will show any error conditions that have not been reset.

Reboot Drive

Selecting **Reboot Drive** will cause the PTi210 module to reboot itself (similar to cycling power). Rebooting will cause PowerTools Studio to lose communications with the PTi210 module.

Stop

Selecting **Stop** from the **Device** menu will stop all motion and programs that are currently active in the PTi210 module. Until the Stop is cleared (by selecting **Stop** from the **Device** menu again), motion and programs will be prevented from being initiated.

Feedhold

Selecting **Feedhold** will put the PTi210 module into a feedhold condition. Feedholding is a means of pausing motion that is active. For more information on Feedhold, see Section 8 - Starting and Stopping Motion in this manual.

Where Am I? (Global)

Selecting **Where Am I?** will launch a utility that shows the user what line in a user program is currently being processed. If multiple user programs are running simultaneously, the user will be asked to specify which task they wish to follow. A blue arrow will appear next to the active line of the program. The global Where Am I will continuously update until it is deactivated. This is different from the Where Am I found on the Program Toolbar.

4.4.4 Options

Figure 4-10 shows the **Options** menu as selected from the PowerTools Studio Menu Bar.

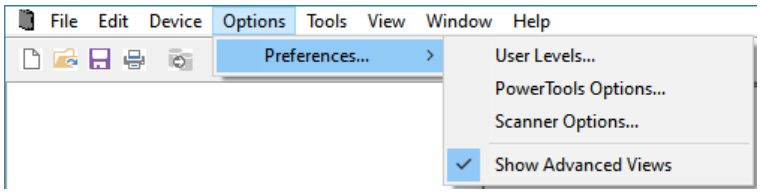


Figure 4-10: Options Menu

Preferences

Under the **Preferences** menu, there are three sub-menus: **User Levels**, **PowerTools Options**, and **Scanner Options**. Selecting one of these sub-menus allows the user to configure the preferences related to that specific topic. Selecting **Show Advanced Views** will show advanced views in the hierarchy tree. The Advanced Views include the Drive Menu Watch and Drive Menu Initialize views.

User Levels

Selecting **Options** > **Preferences** > **User Levels** allows the user to change the quantity and complexity of available parameters.

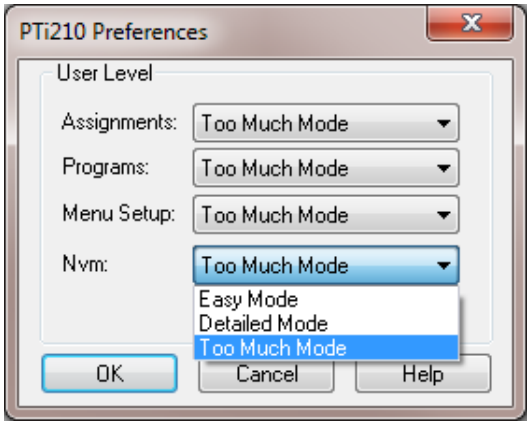


Figure 4-11: User Levels

If a given user level is set to "Easy Mode", then only parameters used in the most basic applications are available. If user level is set to "Detailed Mode", then the most commonly used parameters are available. If user level is set to "Too Much Mode", then all parameters will be visible. This feature is designed to make the most common parameters easier to find and use in programs, assignments, and throughout the software.

PowerTools Options

Selecting **Options** > **Preferences** > **PowerTools Options** allows the user to configure certain settings for the way PowerTools Studio software functions. Figure 4-12 shows the **PowerTools Options Preferences** window.

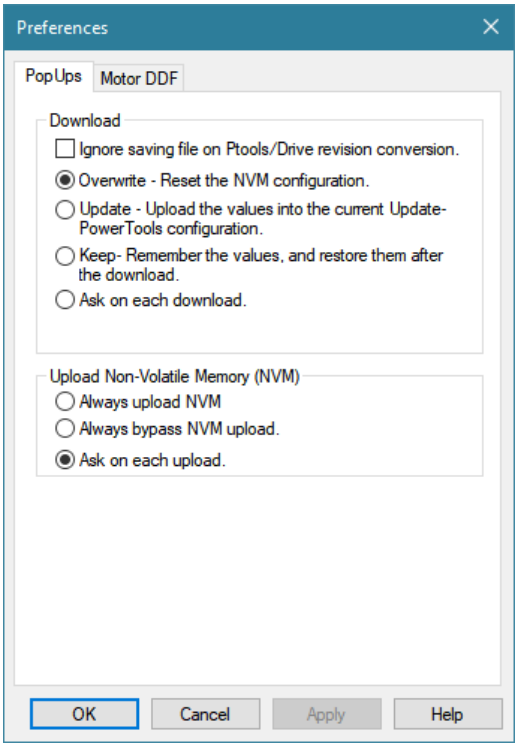


Figure 4-12: PowerTools Studio Preferences-PopUps Tab

The PopUps tab is used to configure several options related to downloading files, uploading files, and file saving. Once the parameters have been set in this window, the user will no longer be prompted with pop-ups when they upload or download files.

For more information on these options, refer to *PowerTools Studio Operation Preferences* on page 27 in this manual.

Motor DDF

The Motor DDF tab specifies the standard and user defined motor .ddf files that PowerTools Studio will use.

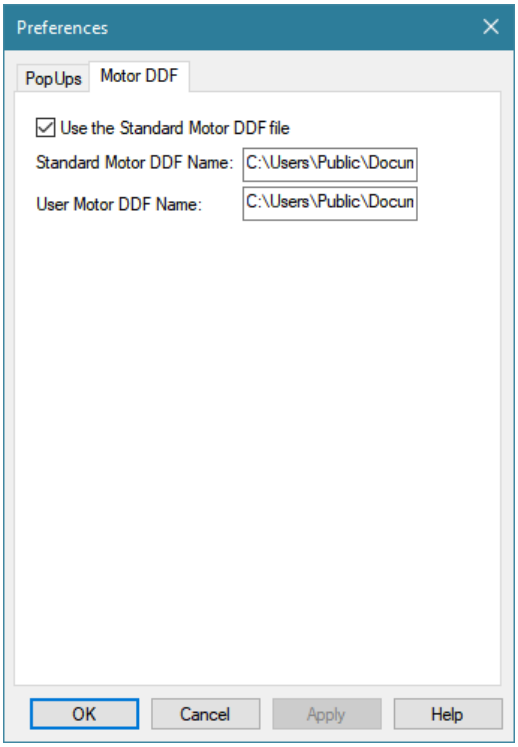


Figure 4-13: PowerTools Studio Preferences - Motor DDF Tab

Use the Standard Motor DDF file

Select this check box to include the standard motor ddf file in the Motor Type selection list when creating a new configuration file.

Standard Motor DDF Name

This box specifies the name and location of the standard motor ddf file. The default file name is stdmotor.ddf.

User Motor DDF Name

When a custom motor is created this box specifies the file name and location of the motor ddf file where the motor information will be stored. The default file name is usermotor.ddf.

4.4.5 Scanner Options

Selecting **Options > Preferences > Scanner Options** allows the user to configure the communication interface which PowerTools Studio will use to communicate with the drive.

At least one protocol must be selected (RTU and/or Ethernet).

When the communication path is not configured in PowerTools Studio, or the user changes the connection path, PowerTools Studio will scan for drives using the enabled protocols.

Figure 4-14 shows the **Discovery Settings** window for the serial RTU protocol and Figure 4-15 shows the **Discovery Settings** for Ethernet.

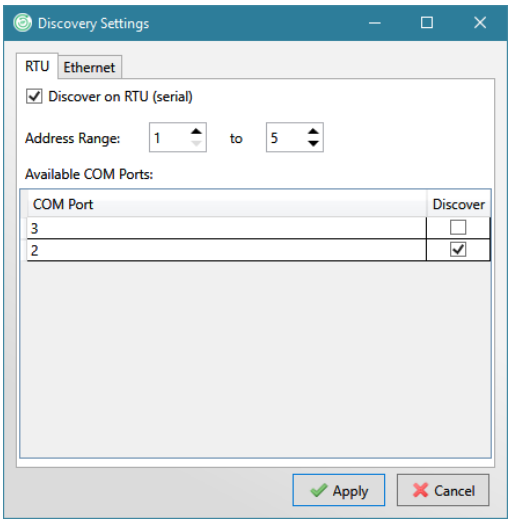


Figure 4-14: PowerTools Studio Communication Discovery Settings (serial RTU)

The available serial communication ports will be shown and must be selected to be included in the discovery scan. Communication ports not selected will not be scanned.

When using the RTU protocol, PowerTools Studio will iterate through each selected communication port and serial address within the specified Address Range using each baud rate in turn to discover connected drives.

NOTE

In previous versions of PowerTools Studio (and PowerTools Pro) the user was required to select the baud rate manually, this is not required in PowerTools Studio (V01.01.XX.XX) or later.

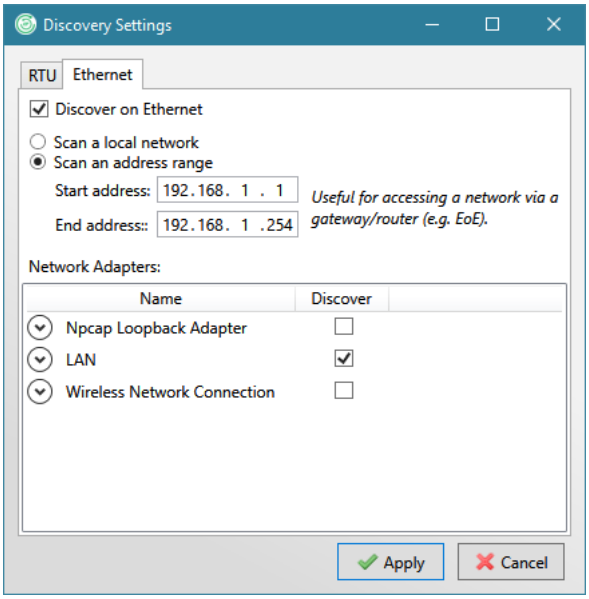


Figure 4-15: PowerTools Studio Communication Discovery Settings (Ethernet)

For both serial RTU and Ethernet, the user can configure which node addresses they wish to poll when uploading, downloading or flash upgrading devices. PowerTools Studio will not discover any devices with node addresses outside the specified range.

4.4.6 Tools

Figure 4-16 shows the **Tools** menu as selected from the PowerTools Studio Menu Bar.

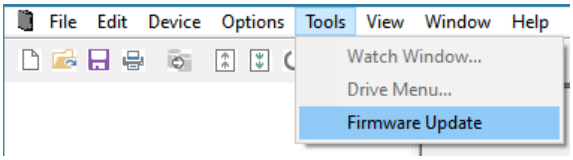


Figure 4-16: Tools Menu

Watch Window

Selecting **Watch Window** will launch a diagnostics tool that allows the user to view the current value of multiple PTi210 parameters while online with the drive. For more information on the Watch Window, refer to *Diagnostics* on page 239 in this user guide.

Drive Menu

The Drive Menu Watch Window allows the user to read or write a single Unidrive M/Digitax HD Menu Parameter from within PowerTools Studio.

Reading a Menu Parameter

To read a Menu Parameter, enter the Menu Parameter number to be read in the Menu.Parameter text box. The Menu Parameter can be entered using either the **mm.pp** or **mm.ppp** formats (where **mm** is the Menu number and **pp** or **ppp** is the parameter number). Once the Menu parameter is entered, then click on the ***Read*** button. The value read from the drive will be displayed in the Parameter Data box.

Writing to a Menu Parameter

To write to a Menu Parameter, enter the Menu Parameter number to be read in the Menu.Parameter text box. The Menu Parameter is entered using either the **mm.pp** or **mm.ppp** formats (where **mm** is the Menu number and **pp** or **ppp** is the parameter number). Once the Menu parameter is entered, then enter the value to be written to that parameter in the Parameter Data box. Then click on the ***Write*** button. The value in the Parameter Data box will be written to the specified Menu Parameter. If the user wishes to verify that the data was written properly, they could either click the ***Read*** button, or navigate to the parameter using the drive keypad manually.

Firmware Update

Selecting **Firmware Update** will launch the utility that lets the user upgrade the firmware in the PTi210 module. New firmware becomes available from Control Techniques to add new features or to upgrade prior releases.

When the user selects **Firmware Update**, PowerTools Studio will search for all devices available on the network. Once PowerTools Studio detects devices, a window similar to that shown in Figure 4-17 will appear.

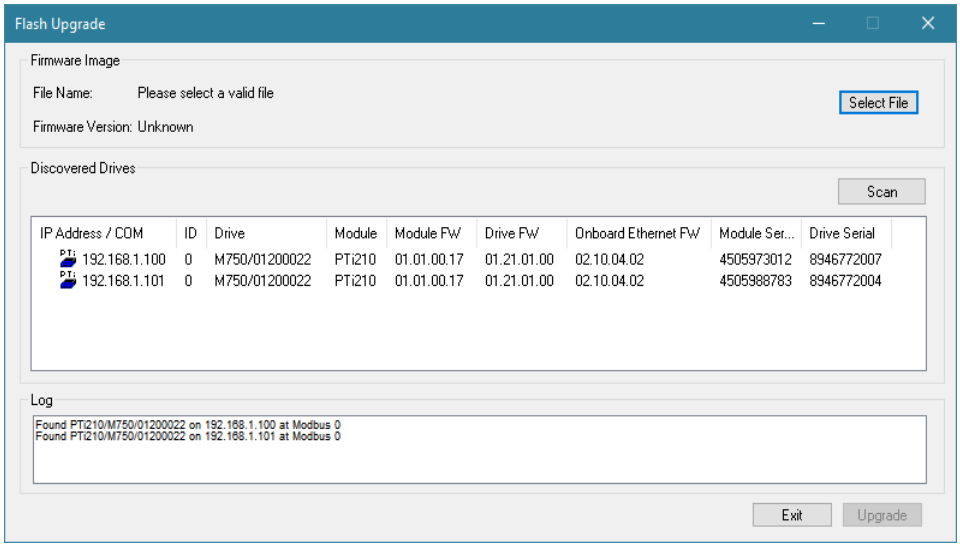


Figure 4-17: Flash Upgrade Window

To upgrade the firmware in a selected node, click on the ***Select File*** button in the top-right corner of the window. Navigate to the folder location where the new flash file is stored. Select the new flash file and click ***Open***. If the selected flash file is compatible with the selected device, a green LED will appear to the left of the device. If the selected flash file is not compatible with the selected device, a red LED will appear to the left of the device. If the LED to the left of the device is green then click ***Upgrade***.

The upgrade process can take up to fifteen minutes using serial Modbus RTU at 19200 baud (Ethernet will be a lot quicker).

4.4.7 View

Figure 4-18 shows the **View** menu as selected from the PowerTools Studio Menu Bar.

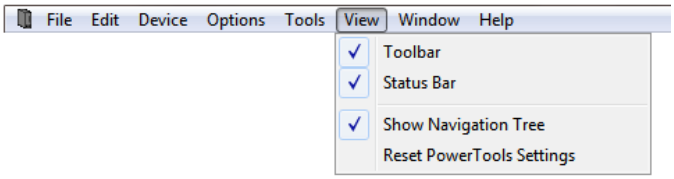


Figure 4-18: View Menu

Toolbar

By default, the Toolbar is visible on the PowerTools Studio screen. To hide the Toolbar, select **Toolbar** to remove it from the display. If the Toolbar is not visible, select Toolbar to make it appear again.

Status Bar

By default, the Status Bar is visible on the PowerTools Studio screen. To hide the Status Bar, select **Status Bar** to remove it from the display. If the Status Bar is not visible, select Status Bar to make it appear again.

Show Navigation Tree

By default, the Hierarchy tree is visible on the PowerTools Studio screen. Some users with low resolution monitors wish to hide the Hierarchy tree to allow for more room while programming. To hide the Hierarchy tree, select **Show Navigation Tree** to remove it from the display. If the Hierarchy tree is not visible, select **Show Navigation Tree** to make it appear again.

Reset PowerTools Settings

Clicking on **Reset PowerTools Settings** renames the emc_fm3.ini file to old_emc_fm3.ini and creates a new emc_fm3.ini with all the original default settings.

4.4.8 Window

Figure 4-19 shows the **Window** menu as selected from the PowerTools Studio Menu Bar.

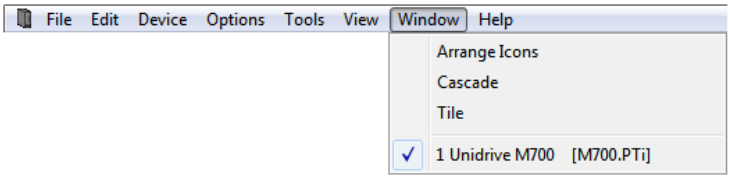


Figure 4-19: Window Menu

Arrange Icons

No function.

Cascade

If the user has multiple configurations open simultaneously, selecting **Cascade** will neatly layer the windows so the title bars are visible and the active configuration is in front.

Tile

If the user has multiple configurations open simultaneously, selecting **Tile** will resize each of the windows to have equal area on the screen. The active configuration will have a highlighted title bar on the window.

Current Files Open

If the user has multiple configurations open simultaneously, each of the open files will be listed on the **Window** menu. The active file will have a check mark next to it as seen in Figure 4-19. By selecting a different file from this list, the selected file will become the active file.

4.4.9 Help

Figure 4-20 shows the **Help** menu as selected from the PowerTools Studio Menu Bar.

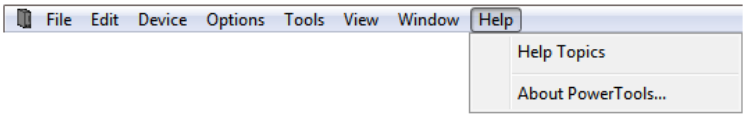


Figure 4-20: Help Menu

Help Topics

By selecting **Help Topics**, the help file will be launched allowing the user to lookup and read information related to the PTi210 module and PowerTools Studio software.

About PowerTools...

About PowerTools will open a window that shows what revision of PowerTools Studio software is currently running.

4.5 Toolbar

4.5.1 New

Same as **File** > **New** from the menu bar. Selecting ***New*** will open a new PowerTools Studio configuration file. The user will be asked what type of configuration to create. Select the correct drive type.

4.5.2 Open

Same as **File** > **Open** from the menu bar. Selecting ***Open*** will allow the user to open an existing application created with PowerTools Studio. Navigate to the directory that the desired file is located in, and double-click on the specific file. Doing so will open the file for editing.

4.5.3 Save

Same as **File** > **Save** from the menu bar. Selecting ***Save*** will save the active file on the users PC. The location to which the file is saved is based on where the file was previously saved. If the file has not yet been saved, the Save As control box will open instead.

4.5.4 Print

Same as **File** > **Print** from the menu bar. Selecting ***Print*** will send the active file to the printer specified by the user. A **Print Options** dialog box will open allowing the user to specify which sections of the configuration are to be printed. By default, all sections will be printed. To remove a given section from the printout, clear the specific check box by clicking on the check mark.

4.5.5 Upgrade Configuration

Same as **File** > **Convert** from the menu bar. PowerTools Studio will open the existing file at the same interface revision at which it was created. This means that not all of the latest features currently supported by PowerTools Studio may be available within the application file. If, after opening an existing file, a user wishes to upgrade the file to the latest interface revision so that all the latest features are available, it can be done by pressing the ***Upgrade Configuration*** button.

Upgrading to the latest revision could possibly change some of the operation of the module, so it is highly recommended to save the file prior to upgrading it, in the event that one needs to revert to the original file. Once the file is upgraded, the user can then save the new file with a different name, and then download that file if they so choose.

When opening a new file, the ***Upgrade Configuration*** button will be unavailable because a new file is always created at the latest interface revision.

4.5.6 Upload

Same as **Device** > **Upload** from the menu bar. Selecting **Upload** will scan the network for available nodes, and then upload the specified node address configurations. Upload is only available if a configuration is not already open.

4.5.7 Download

Same as **Device** > **Download** from the menu bar. Selecting ***Download*** will send the active configuration from the PC to the target node address (specified on the Setup view). For more information on Downloading, see *Communications* on page 22 in this manual.

4.5.8 Reconnect to Device

Same as **Device** > **Reconnect** from the menu bar.
Selecting ***Reconnect to Device*** will re-establish the connection to the drive. The user will have the option to upload the NVM values if required at this time.

This button will only be available if no connection is currently established to the drive. If changes have been made to the application configuration then the user must download the configuration before the connection is re-established.

4.5.9 Change Connection Path

Same as **Device** > **Path Change** from the menu bar. Selecting ***Change connection path*** will open the Set Path Dialog box. The Set Path Dialog box allows the user to scan the network for devices (default) or manually set the protocol and connection details to download or upload.

For the RTU (Serial) protocol, the user can set the Modbus ID number (device node address) and Com Port.

For Ethernet, only the IP Address can be configured.

If the connection path is configured manually, then the user must ensure the correct details are configured, otherwise the connection may not be made to the drive.

NOTE

The network scan is automatically selected by default whenever the communication path is changed from the menu or from the toolbar.

4.5.10 Update to RAM

Same as **Device** > **Update Drive** from the menu bar. Selecting ***Update to RAM*** will send any parameters that have been changed since the last download into RAM. Doing so allows the user to send changes to the system without requiring a complete download. Certain parameters when changed require a complete download and cannot be sent to RAM. If one of these parameters is changed, the Update Drive option will be unavailable on the **Edit** menu. Parameter values only sent to RAM will be lost when power is cycled.

4.5.11 Upload NVM

Selecting ***Upload NVM*** (Non-Volatile Memory) will read the current value from each of the parameters in the PTi210's NVM memory and display it in the PowerTools Studio configuration. The file can then be saved to retain the current values stored in NVM.

4.5.12 Disconnect

Same as **Device** > **Disconnect** from the menu bar. Selecting ***Disconnect*** will terminate communications between the PC and any nodes the PC is online with.

4.5.13 New Index

Same as **Edit** > **New** > **Index** from the menu bar. Selecting ***New Index*** will add a new index to the configuration. Indexes are added in sequential order, up to a maximum of 100. The new index will be the next highest available index number. Adding an index will take you directly to the new index view. Can not be used while online.

4.5.14 Delete Index

Same as **Edit** > **Delete** > **Index** from the menu bar. To delete an index, the user must select they specific index they wish to delete on the hierarchy tree. Once the index is selected, click ***Delete Index*** on the toolbar. Doing so will delete the index instance. Once the index is deleted, the data stored on the index view cannot be recovered. Can not be used while online.

4.5.15 New Program

Same as **Edit** > **New** > **Program** from the menu bar. Selecting ***New Program*** will add a new user program to the configuration, up to a maximum of 100. Programs are added in sequential order. The new program will be the next highest available program number. Adding a program will take you directly to the new program view. Can not be used while online.

4.5.16 Delete Program

Same as **Edit** > **Delete** > **Program** from the menu bar. To delete a user program, the user must select the specific program they wish to delete on the hierarchy tree. Once the program is selected, click ***Delete Program*** on the toolbar. Doing so will delete the program instance. Once the program is deleted, the program code cannot be recovered. Can not be used while online.

4.5.17 View Current Errors

Same as **Device** > **View Errors...** from the menu bar. Selecting ***View Current Errors*** will open the Active Errors pop up window. The Active Errors window will show any error or trip conditions that have not been reset.

4.5.18 Clear Errors

Same as **Device** > **Reset Errors** from the menu bar. Selecting ***Clear Errors*** will clear any active errors. If the fault condition still exists, the error may reactivate immediately after clearing it.

4.5.19 Watch Window

Same as **Tools** > **Watch Window** from the menu bar. Selecting ***Watch Window*** will launch a diagnostics tool that allows the user to view the current value of multiple PTi210 parameters while online. For more information on the Watch Window, refer to *Diagnostics* on page 239 of this manual.

4.5.20 Drive Watch Window

Selecting ***Drive Watch Window*** will launch a diagnostics tool that allows monitoring and editing of individual drive menu parameters while online. For more information see *Drive Menu* on page 18.

4.5.21 Stop All

Using the ***Stop All*** button on the toolbar will stop all motion and programs that are currently active in the PTi210 module. The ***Stop All*** button will toggle on and off meaning that once it is clicked to stop motion and programs, it will remain active until it is clicked again. Until the ***Stop All*** is toggled off, motion and program will be prevented from being initiated.

4.5.22 Feedhold

Same as **Device** > **Feedhold** from the menu bar. Selecting ***Feedhold*** will put the PTi210 module into a feedhold condition. Feedholding is a means of pausing motion that is active. For more information on Feedhold, see *Starting and Stopping Motion* on page 179 in this manual.

4.5.23 Global Where Am I?

Same as **Device** > **Where Am I?** from the menu bar. Selecting ***Where Am I?*** will launch a utility that shows the user what line in a user program is currently being processed. If multiple user programs are running simultaneously, the user will be asked to specify which Task they wish to follow. A blue arrow will appear next to the active line of the program. The global Where Am I will continuously update until it is deactivated. This is different from the Where Am I found on the Program Toolbar.

4.5.24 Hide/Show Hierarchy Tree

Same as **View** > **Show Navigation Tree** on the menu bar. By default, the Hierarchy tree is visible on the PowerTools Studio screen. Some users with low resolution monitors wish to hide the hierarchy tree to allow for more room while programming. To hide the hierarchy tree, select ***Hide/Show Hierarchy Tree*** from the toolbar to remove it from the display. If the hierarchy tree is not visible, select Navigation Tree to make it appear again.

4.5.25 Help Contents

Same as **Help** > **Help Topics** on the menu bar. By selecting ***Help Contents***, the help file will be launched allowing the user to lookup and read information related to the PTi210 module and PowerTools Studio software.

4.5.26 Context Sensitive Help (CSH)

Using ***Context Sensitive Help*** (or CSH) will show detailed information from the help file related to the object that is clicked with the mouse. To use Context Sensitive Help, click the ***CSH*** button on the toolbar, the mouse graphic will turn from a pointer to a pointer with a question mark next to it. Once the mouse pointer graphic changes, click on a parameter on any of the PowerTools Studio views that the user wants help information for.

4.6 Hierarchy Tree

Figure 4-2 on page 9 shows the Hierarchy Tree as found in PowerTools Studio. The Hierarchy Tree is a navigational aid that helps the user step through a configuration. The different parameters related to configuring the system are grouped into logical groups and placed on different views within the software. To see a certain view, the user simply selects that view from the Hierarchy Tree. Once the user clicks on a given branch on the Hierarchy tree, that view is then displayed on the right-hand side of the screen.

Groups on the hierarchy tree can be expanded and collapsed much like with Windows Explorer. To expand a group of views, click on the " > " symbol next to the grouping. Doing so will show each of the branches available within that group. To collapse a group on the hierarchy tree, click on the " ⌄ " symbol next to the grouping. Doing so will hide all of the branches within that group.

Natural progression through the hierarchy tree, starting at the top, and working towards the bottom will step the user through the entire configuration. The Hierarchy tree starts at the top with Hardware, then moves on to Setup parameters, I/O Setup, Motion, and then finishes with Programs. When the user gets to the bottom of the hierarchy tree, the configuration should be ready to be downloaded to the PTi210 module.

4.7 View

Figure 4-2 on page 9 shows an example of a view in PowerTools Studio. A View will typically contain text boxes, list boxes, check boxes, or other windows editing mechanisms. The views are designed to separate parameters into logical groups so that they are easier to find and use. The view that is visible at any time is dependant upon what branch is selected on the hierarchy tree. To see a specific view, it must be selected on the PowerTools Studio hierarchy tree.

4.8 View Tabs

Some views in PowerTools Studio (see explanation of view above) have tabs, in the bottom half of the view to organize more specific parameters. Some examples of View Tabs are Calculations and Online. An Online Tab is only visible when online (connected) with a module.

On the Calculations Tab the user will find simple calculations to help realize how much time a motion will take, or how much time or distance is covered during certain segments of a motion profile. On the Index – Calculation Tabs, a graph is created based on data entered by the user to give some visualization of what the profile looks like.

The Online Tab is used to show feedback and other diagnostic information to the user while online with a module. The parameters shown on an Online view will change depending on which View is being displayed. PowerTools Studio must be online in order to see the Online Tab.

4.9 Status Bar

The Status Bar is used to give quick diagnostic information to the user about the status of the drive/module, what motion is active, Position and Velocity Feedback, Communication Path, and Online Status. For more details, see the *Diagnostics* on page 239.

5 Communications

5.1 Communications Protocol

PowerTools Studio communicates with the PTi210 module using 32-bit Modbus RTU or Ethernet protocol.

The following table shows the supported drives and their factory fitted communication interface.

Drive	Derivative	Communication Interface
Unidrive	M700	Ethernet
	M701	Modbus RTU
	M702	Ethernet
Digitax HD Ethernet	M750	Ethernet
Digitax HD Base	M751	Modbus RTU

5.2 Connecting the PC to the PTi210 Module

5.2.1 Modbus

The PTi210 module communicates with the PC through the drive's EIA-485 communication port. An EIA-232 to EIA-485 converter is necessary to communicate with the drive. The Unidrive M701 and Digitax HD Base M751 each have two RJ45 serial interface ports on the front of the drive.

Control Techniques offers a pre-made cable for this purpose called “CT Comms Cable" (previously named SE71). Figure 5-2 shows the CT Comms Cable with a standard 9-pin 'D' type connector for connection to the EIA-232 port on the PC, an alternative cable for connection to the PC USB port is also available from Control Techniques (Part number 4500-0096), see Figure 5-1.

Figure 5-1: CT Comms Cable (USB)

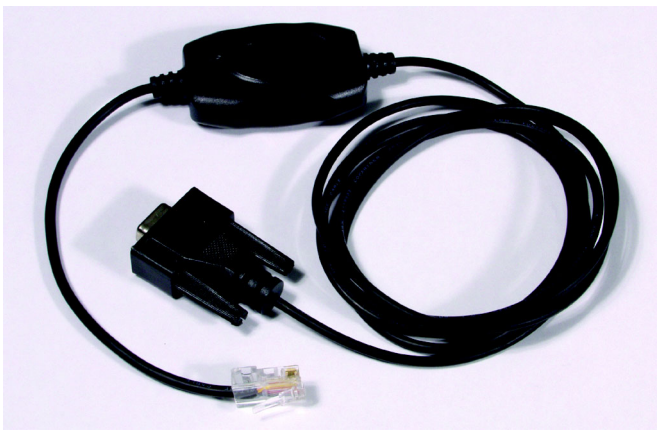


Figure 5-2: CT Comms Cable (9-pin 'D')

On the Unidrive M701 drive, the two RJ45 serial interface ports are located on the front of the drive, just below the keypad and in line with the status LED, and are identified by the text "485".

On the Digitax HD Base M751 drive, the two RJ45 serial interface ports are located on the front of the drive, just above the external 24 Vdc supply connectors.

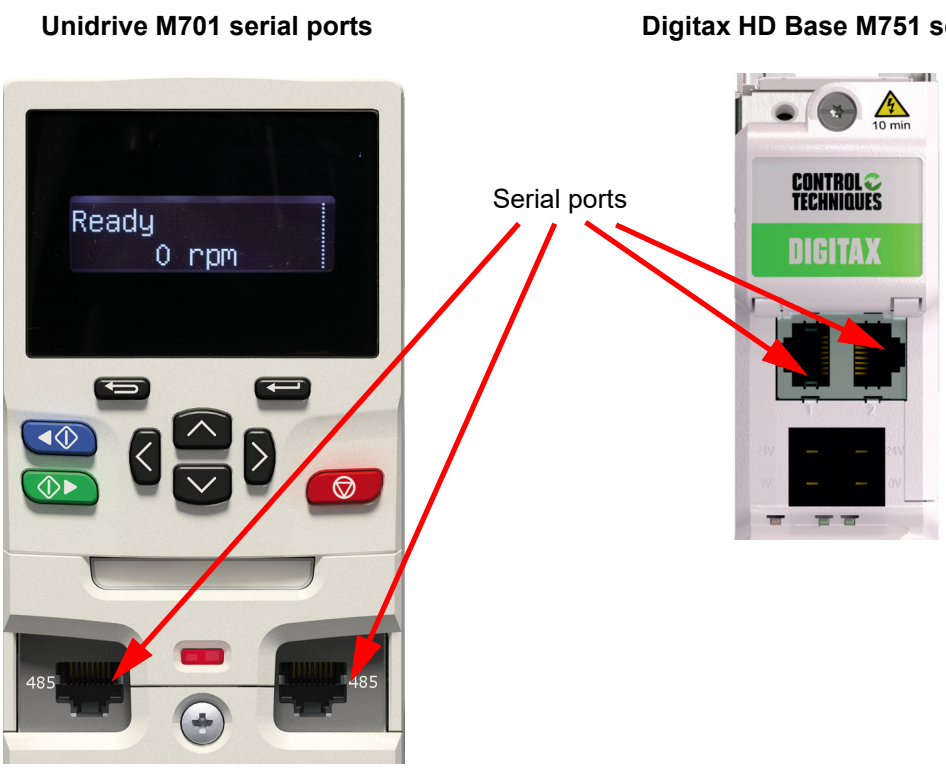


Figure 5-3: Unidrive M701/Digitax HD M751 Serial Port Connections

The pin out connections are the same for both the Unidrive M701 and the Digitax HD M751 drives and are shown in the following table.

Pin	Function
1	Termination Resistor
2	RX TX
3	0 V
4	+24 V
5	Not Used
6	TX Enable
7	RX\ TX\
8	RX\ TX\ (Link to pin 1 for termination resistor)

5.2.2 Ethernet

Ethernet communication is provided as standard on the Unidrive M700, M702, and Digitax HD Ethernet M750, PowerTools Studio is designed to communicate using the onboard Ethernet interface with firmware V02.07.00.00 or later, it will not communicate with earlier firmware versions or via an Ethernet capable option module. The drive communicates to the PC using a standard RJ45 connection to a 10 Mbps/100 Mbps Ethernet system. For more information on the Ethernet interface see the relevant drive User Guide.

5.3 Configuring Communications in PowerTools Studio

When attempting to upload or download a configuration using PowerTools Studio, the software may need to be configured to the correct communication settings for the intended connection.

To configure the communication scanner preferences, select **Options** > **Preferences** > **Scanner Options** from the menu bar.

For more information on configuring the communication path please refer to the section **Scanner Options** on page 16.

5.4 Uploading and Downloading using PowerTools Studio

Figure 5-4 will be used throughout the Uploading and Downloading section of the manual to describe certain processes.

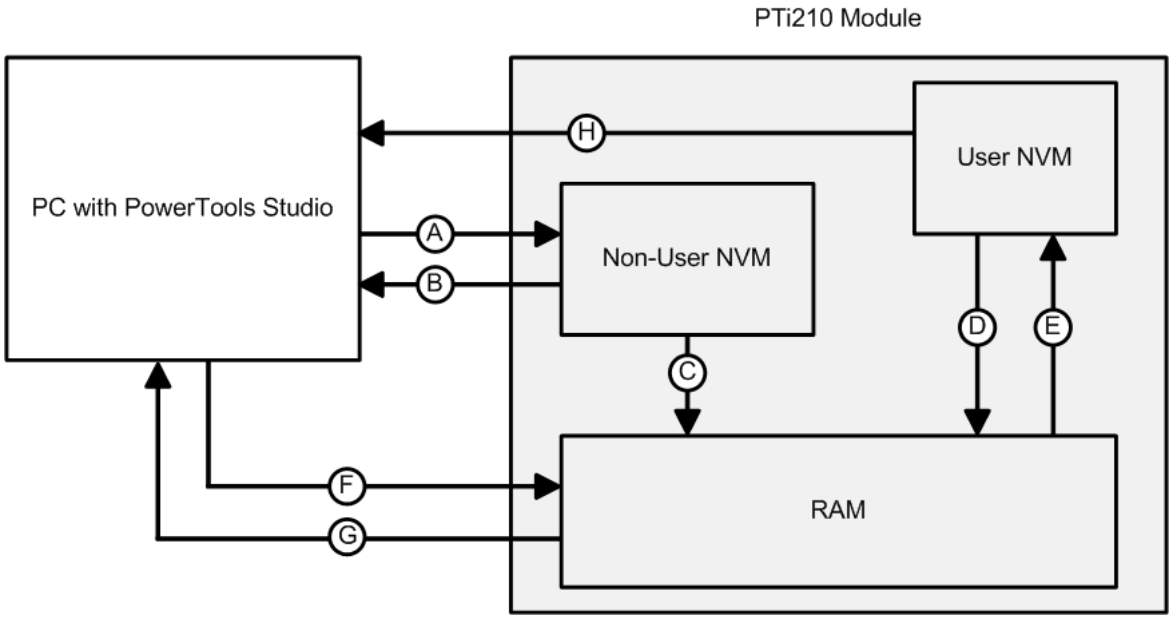


Figure 5-4: Memory Storage in the PTi210

5.4.1 Uploading

Uploading is the process of reading the configuration stored in the PTi210 module and loading that data into a configuration file on the PC. Arrow B in Figure 5-4 represents a standard Upload.

There are two ways to upload a configuration, if no configuration is currently loaded then the user must scan the network(s) and select which configuration to upload, in this case all detected drives regardless of drive type will be listed. If a configuration is currently loaded then the network scan will only list the detected configurations for similar drive types to the currently loaded configuration drive type.

Alternatively, from firmware V01.09.xx.xx, the user can manually select the node address (Modbus ID/IP Address) to connect to and bypass the network scan.

No configuration file loaded

With no configuration file loaded in PowerTools Studio, click on the ***Upload All…*** button on the toolbar or select ***Device*** > ***Upload All…***, PowerTools Studio scans the selected network(s) for drives, when the required drive is detected the user can stop the scan and select the required drive configuration to upload then click the ***Upload*** button to upload the configuration into PowerTools Studio. Alternatively, if the node address has been manually configured, then PowerTools Studio will attempt to use the configured node address settings to upload the configuration.

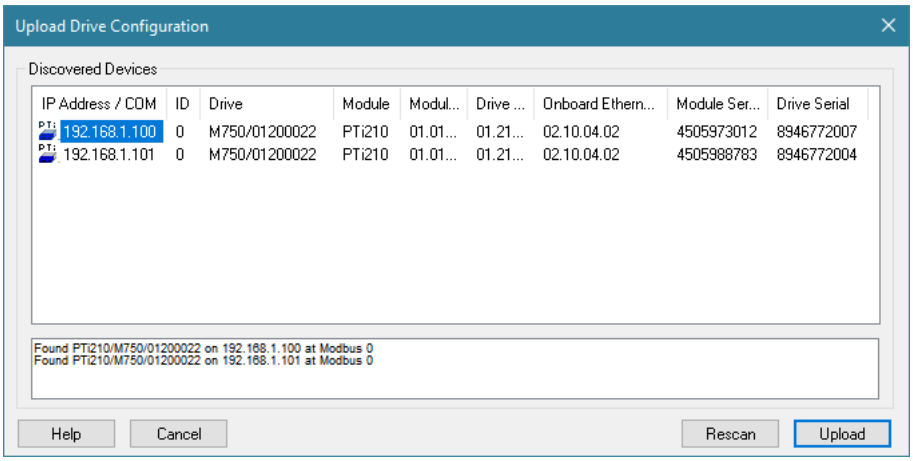


Figure 5-5: Upload Drive Configuration Dialog Box

Configuration file loaded

If a configuration file is currently loaded in PowerTools Studio, from the menu bar select **Device** > **Upload Drive**, PowerTools Studio scans the selected network(s) for similar drives of the same type, when the required drive is detected the user can stop the scan and select the required drive configuration to upload by clicking the ***Upload*** button. Alternatively, if the node address has been manually configured, then PowerTools Studio will attempt to use the configured node address settings to upload the configuration.

Whichever method is used, the Upload Drive Configuration dialog box will contain the following information for each drive found:

- IP Address/COM
- Modbus Address ID
- Drive Type
- Module Type
- Module Firmware Version
- Drive Firmware Version
- Module Serial Number
- Drive Serial Number

5.4.2 Downloading

Downloading is the process of sending the PowerTools Studio configuration from the PC to the PTi210 module. Changes made in PowerTools Studio will not take effect until the information has been downloaded or the ***Update to RAM*** button on the toolbar has been clicked.

Arrow A in Figure 5-4 represents a standard Download.

To download information to the PTi210 module, click on the ***Download*** button on the PowerTools Studio toolbar or select **Device** > **Download** from the menu bar.

If the node address has been manually configured (see section 4.5.9 - Change Connection Path), then the download process will begin immediately using the manually configured connection path.

If the node address has not been manually configured, then the PowerTools Studio will scan the network(s) for available devices and the Download to Device ID x window will be shown.

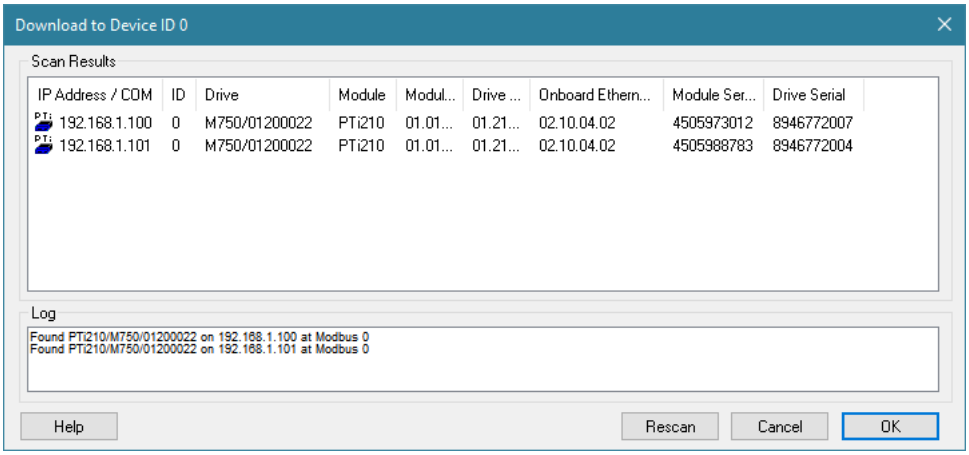


Figure 5-6: Download Drive Configuration Dialog Box

The Download to Device ID x (x = the node address of the drive) dialog box will open, all communication connections are scanned and the results appear. The Download Drive Configuration dialog box contains the following information for every device found:

- IP Address/COM
- Modbus Address ID
- Drive Type
- Module Type
- Module Firmware Version
- Drive Firmware Version
- Module Serial Number
- Drive Serial Number

The required drive can then be selected by clicking on it, and the loaded configuration can be downloaded to it by clicking the ***OK*** button.

5.4.3 Non-Volatile Memory (NVM) Options for Uploading and Downloading

When Uploading or Downloading, the user may be presented with options on how to handle the values stored in NVM memory. NVM is a type of memory that does not lose its contents when power is removed. Values in RAM are lost on power down, while values in NVM are retained. Following is a description of the options the user may encounter while uploading or downloading.

Uploading

When uploading from the PTi210 module, the values that were last downloaded are uploaded and put into a PowerTools Studio configuration file (Arrow B in Figure 5-4). At the completion of the upload, the user will be asked if they wish to upload the NVM values. This dialog box is shown below.

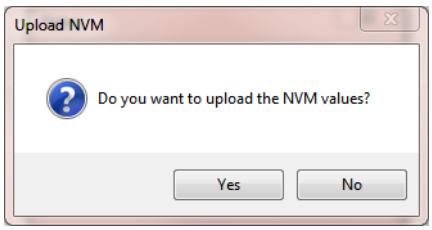


Figure 5-7: Upload NVM Option Window

By selecting ***Yes***, the values of all parameters stored in NVM will be uploaded and entered into the PowerTools Studio configuration file values (Arrow H in Figure 5-4). If ***No*** is selected, the values entered into the PowerTools Studio configuration file will remain the same as those that were last downloaded to the PTi210 module.

Downloading

When downloading to the PTi210 module the user will be required to select how to handle the NVM parameters upon downloading. Figure 5-8 shows the dialog box asking the user to select one of three options for the download.

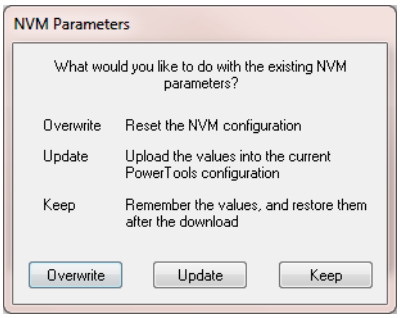


Figure 5-8: Download NVM Option Window

A description of each of the options is as follows:

Overwrite

This option will overwrite all the parameters stored in NVM with the current values in the user configuration (Just Arrow A in Figure 5-4). The values that are in NVM prior to the download will be lost.

Update

This option will upload the current NVM parameter values from the PTi210 module and enter them into the user configuration. Once the NVM values have been stored in the file, the file is fully downloaded (First Arrow H followed by Arrow A in Figure 5-4. Data from H is saved in PowerTools Studio configuration).

Keep

This option will download the entire user configuration, but then NVM parameters will be restored to the value prior to download. This is similar to the Update option, but the Keep option does not upload the NVM values into the user configuration (First Arrow H followed by Arrow A, but data from H is not stored in PowerTools Studio file).

The table below shows an example of how these three options work:

	Value Before Download	Value After Download		
		Overwrite Option	Update Option	Keep Option
PowerTools Studio Value for Index.0.Vel	150	150	500	150
NVM Value for Index.0.Vel	500	150	500	500

5.4.4 Update to RAM

The ***Update to RAM*** button can be used to send changes to the PTi210 module without performing a complete download (symbolized by Arrow F in Figure 5-4). The ***Update to RAM*** button is found in the PowerTools Studio toolbar. This operation will send only those changes that have been made since the last Update to RAM or **Device** > **Download** to the PTi210 module. The changes will take effect immediately upon clicking on the button.

The changed parameters will be sent to the PTi210 module without stopping motion or disabling the drive. Because of this, it is important to use caution when changing motion parameters while the motor is in motion.

The ***Update to RAM*** button saves the parameters only to RAM and not to Non-Volatile Memory (NVM). Therefore, if the system power is removed, any changes made using the ***Update to RAM*** button will be lost. In order to save changes to NVM, a full-download must be performed.

The flowchart in Figure 5-9 describes a typical process using the Update to RAM to make changes, and then downloading when complete to save changes to NVM.

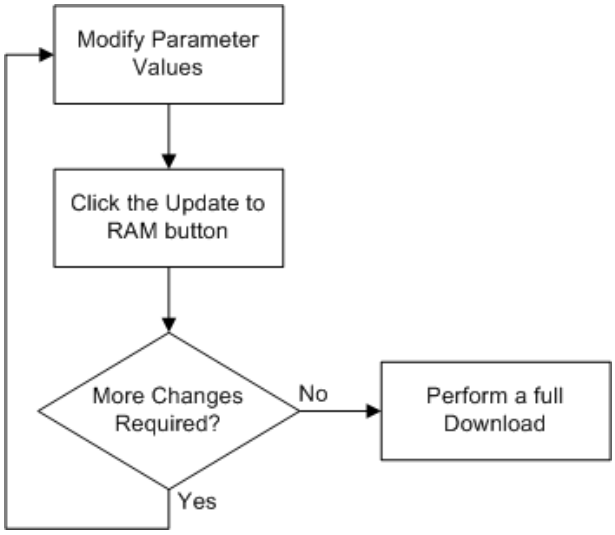


Figure 5-9: Update to RAM flowchart

The ***Update to RAM*** button operates according to the following rules:

- If no parameters have been modified, the Update to RAM button will be disabled.
- If the user modifies a parameter that does not require a full download, the Update to RAM button will become enabled.
- If while the button is enabled, the user modifies a parameter that requires a full download, the Update to RAM button will become disabled.
- When the user clicks on the ***Update to RAM*** button, all the modified parameters are transmitted to the PTi210 module. Once transmitted, the button will become disabled again until another parameter is changed.
- If the user performs a full download while the button is enabled, the ***Update to RAM*** button will be disabled when the download is complete.
- If the user modifies parameters, and then disconnects (stops communications), the ***Update to RAM*** will be disabled, and the changes will not be sent.

5.4.5 PowerTools Studio Operation Preferences

To avoid getting all the option windows described above every time an Upload or Download is performed, the user can set certain preferences. To configure the preferences, select **Options** > **Preferences** > **PowerTools Options** from the menu bar. Figure 5-10 shows the **Preferences** window.

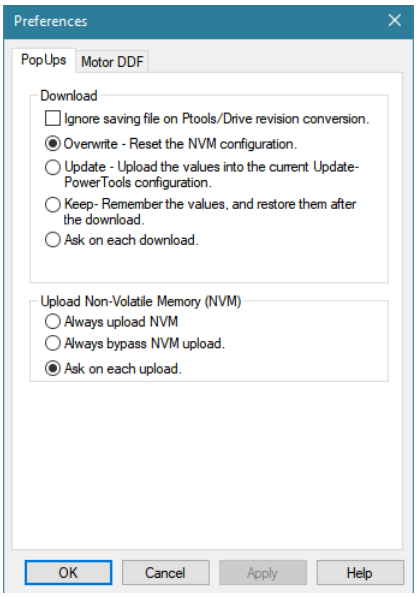


Figure 5-10: PowerTools Studio Preferences Window - PopUps Tab

Following is a description of each of the preference settings:

Download Group

Ignore saving file on Ptools/Drive revision conversion

On a download PowerTools Studio first checks the revision of the PTi210 module firmware before downloading the configuration. If the firmware in the module is older than the matching software revision, PowerTools Studio then converts the user configuration to match the firmware revision in the module. Because certain parameters may need to be changed to match the firmware revision, PowerTools Studio asks the user if they wish to save the file before it is converted. If the user wishes to download without saving every time, and therefore avoid being asked on every download, this check box should be selected.

Overwrite - Reset the NVM configuration

When this radio button is selected the "Overwrite" option will be used on every download to the module. For more details on how the Overwrite option works, see the Download NVM Options above.

NOTE

It is required to Overwrite the Non-Volatile Memory on the first download to the module since no Non-Volatile Memory parameters have been loaded into the drive on initial startup.

Update - Upload the values into the current Update PowerTools Studio configuration

When this radio button is selected the "Update" option will be used on every download to the module. For more details on how the Update option works, see the Download NVM Options above.

Keep - Remember the values and restore them after the download

When this radio button is selected the "Keep" option will be used on every download to the module. For more details on how the Keep option works, see the Download NVM Options above.

Ask on each download

When this radio button is selected, the user will be prompted on every download to select either the Overwrite, Update, or Keep option. This is the default preference setting.

Upload Non-Volatile Memory (NVM) Group

Always upload NVM

When uploading a configuration, PowerTools Studio uploads from a Non-User NVM location, so the data uploaded only matches exactly what was last downloaded. If any parameters in the Save to NVM list have changed since the last download, those new values would not be uploaded. By selecting this radio button, the parameter values in the Save to NVM list will be uploaded into the PowerTools Studio configuration after the normal upload.

Always bypass NVM upload

When uploading a configuration, PowerTools Studio uploads from a Non-User NVM location, so the data uploaded only matches exactly what was last downloaded. By selecting this radio button, the parameter values in the Save to NVM list will NOT be uploaded.

Ask on each upload

When this radio button is selected, PowerTools Studio will ask the user via a pop-up window whether to upload the NVM or to bypass the NVM upload on every upload.

5.4.6 Secure Downloading

The Secure Download feature allows the user to download a configuration that prevents anyone from uploading the file, or going online with the system. This is used to protect a file from being accessed by unauthorized personnel. If a secure file is downloaded to the PTi210 module, all diagnostics capabilities in the software are lost. The only way to go online with the system again is to download the original (non-secure) file over the secure version, or to download a completely new file.

Before performing a secure download, the file must first be saved in the secure file format. To do this, open the file you wish to save in the secure format using PowerTools Studio. Then click **File** > **Save As** on the menu bar. The following **Save As** window should appear on your screen.

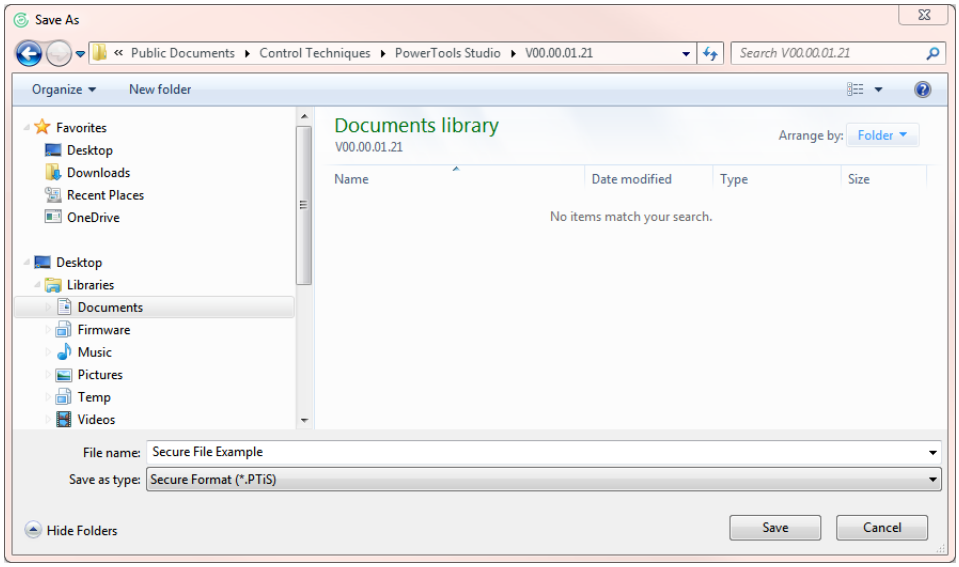


Figure 5-11: Secure File - Save As Window

On this window, enter the name of the secure file to create and select the "Secure Format (*.PTiS)" selection from the "Save as type" selection box, then click ***Save***. Doing so will save the file in the secure file format (.PTiS).

The secure file will be saved to the same directory as the standard file. To perform the Secure Download, close all open files in PowerTools Studio, click on **Device** menu, **Secure Download** command as shown in Figure 5-12.

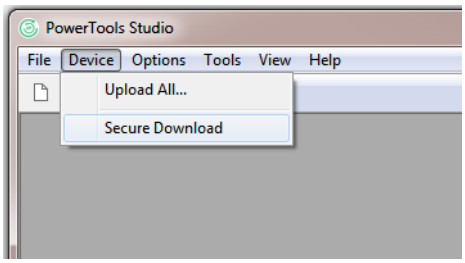


Figure 5-12: Secure File Download

A window will then open asking the user to select the secure file they wish to download. Select the secure file that was just saved, and click ***Open***. This will download the secure file to the target device.

A secure file (.PTiS) cannot be opened or modified. The file extension cannot be changed to allow the user to open it. The secure file is only valid for use by the secure download function. If a user attempts to upload a secure file, a message will appear indicating that the file that resides in the PTi210 module has been protected by the user. An example of this message is shown in Figure 5-13.

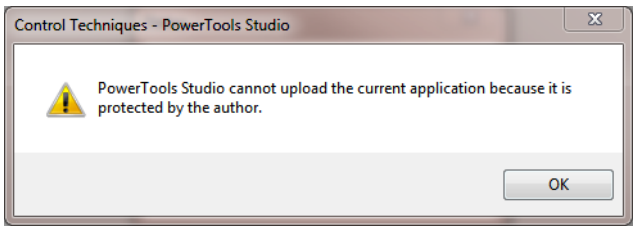


Figure 5-13: Upload Failure - File Protected By User message

5.4.7 Change Path

PowerTools Studio allows the user to change between a drives communication IP address or Com port.

To change communication path, click the ***Change connection path*** button on the PowerTools Studio toolbar or on the **Device** menu, click **Path Change,** the user can then select to manually enter the node address or allow the network scanner to run (default). After the network scanner has scanned (if enabled) and displayed any detected drives (Figure 5-14), the user then clicks the **OK** button on the Network Scanner results to show the Change Path dialog box, from which the required drive is selected (Figure 5-15).

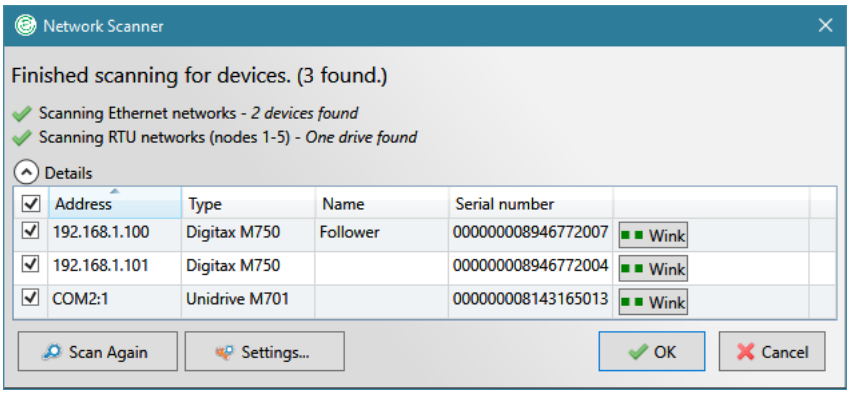


Figure 5-14: Network Scanner results dialog box

To select the desired connection path, the user should click the **OK** button, this will open the Change Path dialog box.

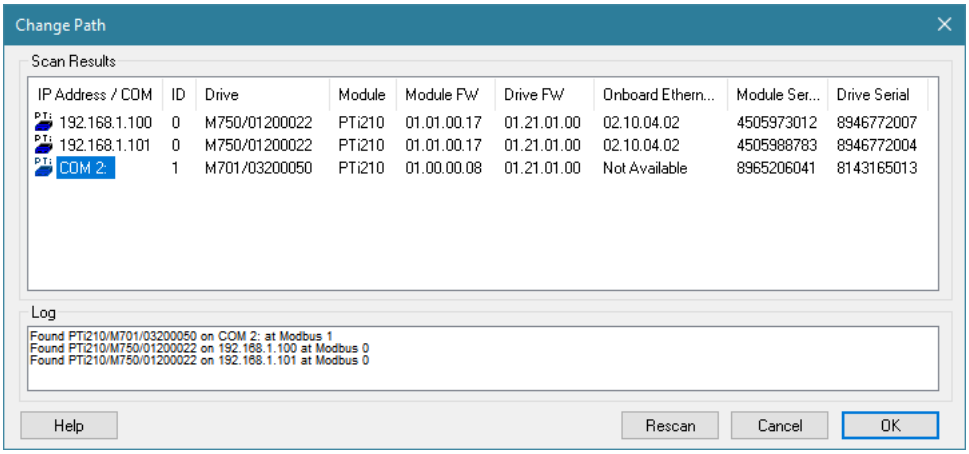


Figure 5-15: Change Path Dialog Box

The Change Path dialog box contains the following information for every device found:

- IP Address/COM
- Modbus Address ID
- Drive Type
- Module Type
- Module Firmware Version
- Drive Firmware Version
- Module Serial Number
- Drive Serial Number

Select the devices communication connection you wish to change to and click ***OK***.

If the node address is manually configured, then the configured connection path will be displayed in the Status Bar.

5.5 Accessing Drive Menu Parameters Using Other Communication Programs

It is possible, for diagnostics or monitoring purposes, to access the drive menu parameters using other communications programs (for example, Control Techniques' Drive Commissioning Tool (Connect)), however, as the PTi210 Module accesses many drive menu parameters (some only during initialization), care should be taken to ensure that no drive menu parameter used by the PTi210 Module is written to using other communications programs, as any changes may affect the operation of the PTi210 Module, resulting in damage to machinery or injury to personnel.

6 How Motion Works

The PTi210 module offers six different motion object types. These six types are Jogs, Home, Indexes, Gearing, Camming, and Torque Mode. All motion objects run on what is called a Profile. Only one motion object can run on a Profile at a time. For applications that need to run multiple motion objects simultaneously, the PTi210 module has two different Profiles. This will be discussed further in the Summing Multiple Profiles on page 44 section 6-7.

This section will concentrate on how the different motion objects work, and not on how they are configured using PowerTools Studio. For more information on how to configure the motion objects in PowerTools Studio, see Configuring an Application on page 47.

6.1 Jog

Jog is a motion object that has Acceleration, Velocity, and Deceleration, but no dedicated distance. The user pressing a push button or foot pedal often controls jog motion. When the user presses the Jog button, the jog accelerates up to the jog velocity and continues to run at that velocity until the user releases the Jog button. When the user releases the Jog button, the jog decelerates from the Jog Velocity back down to zero velocity (stopped).

Figure 6-1 shows an example of a Jog profile.

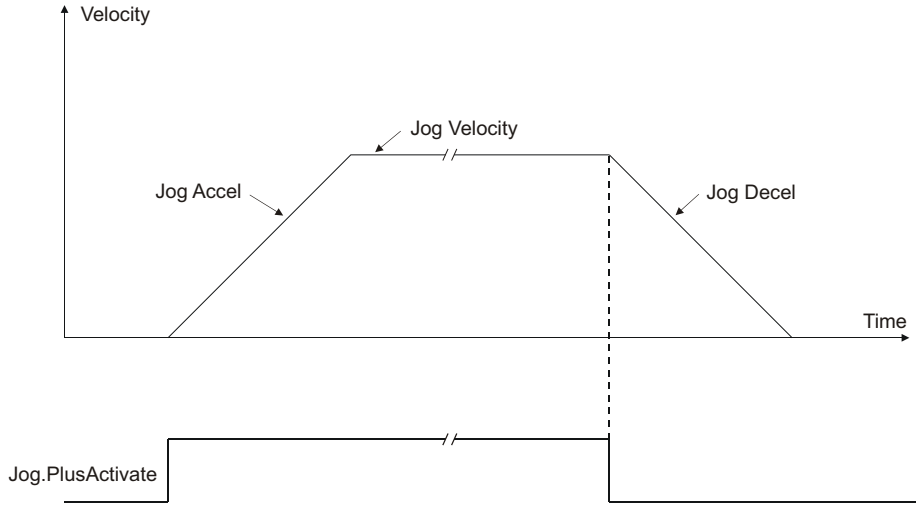


Figure 6-1: Jog Profile Diagram

The distance traveled by a Jog is entirely dependant on the duration that the Jog signal is held active. To accurately control the distance the motor moves, an Index is the preferred motion object type.

6.2 Home

Home is a motion object that has Acceleration, Velocity, and Deceleration. A Home works by accelerating up to the specified velocity until a reference signal activates. Once the reference signal activates, the motor either begins to decelerate to a stop immediately (called Calculated Offset), or continues at the Home velocity and comes to a stop a specified distance from where the reference signal activated (called Specified Offset).

The Home is typically used to define a reference position (or Home position) on a machine. This can be done in several different ways depending on what type of Home Reference is used. The Home Reference determines what action or signal defines the Home position. The three types of Home Reference supported by the PTi210 module are Marker, Sensor, and Sensor then Marker. Following is a description of the three different Home types.

6.2.1 Home to Marker

Most motors used with the drive have an encoder mounted in the back end of the motor. This encoder is used to feed positional data back to the drive or the PTi210 module for position control. Some encoders have a special signal called the Encoder Marker Channel that activates once every revolution of the motor. The PTi210 module can use this Marker as a reference signal when executing a Home.

The Marker provides an extremely accurate means of homing the system. The PTi210 module will detect a rising edge of the Encoder Marker signal every revolution of the motor shaft. The Home routine begins by accelerating up to the Home Velocity. The motor continues at the Home Velocity until the Encoder Marker activates. The motor then decelerates to a stop immediately, or continues for a specified offset distance. Figures 6-2 and 6-3 show examples of the Home to Marker profile, with calculated offset and with specified offset respectively.

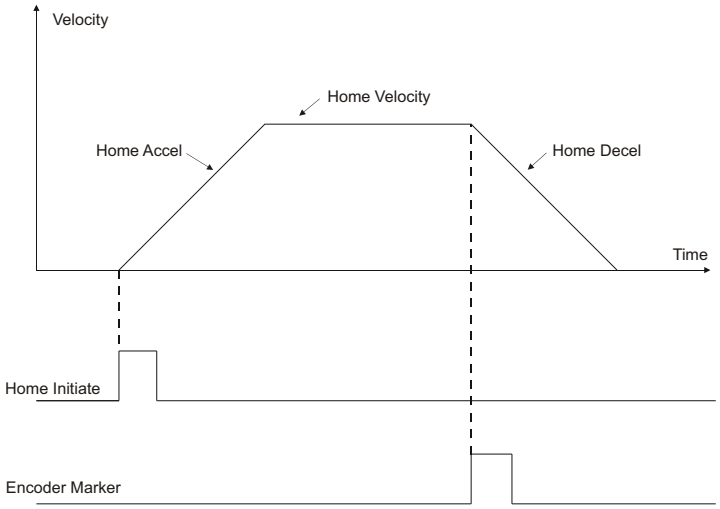


Figure 6-2: Home to Marker Profile (Calculated Offset)

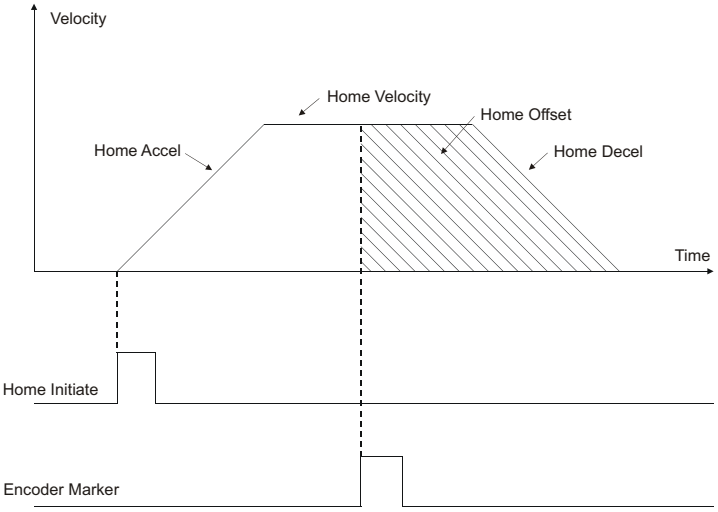


Figure 6-3: Home to Marker Profile (Specified Offset)

6.2.2 Home to Sensor

The Home to Sensor profile acts much like the Home to Marker profile, but instead of using the Encoder Marker as a reference, an external sensor mounted to the machine is used as the reference. In a Home to Sensor routine, the motor accelerates to the Home Velocity. The motor continues at the Home Velocity until the external sensor activates. Once this sensor activates, the motor immediately decelerates to a stop, or continues for a specified offset distance before stopping. Figure 6-4 and Figure 6-5 show examples of the Home to Sensor profile, with calculated offset and specified offset respectively.

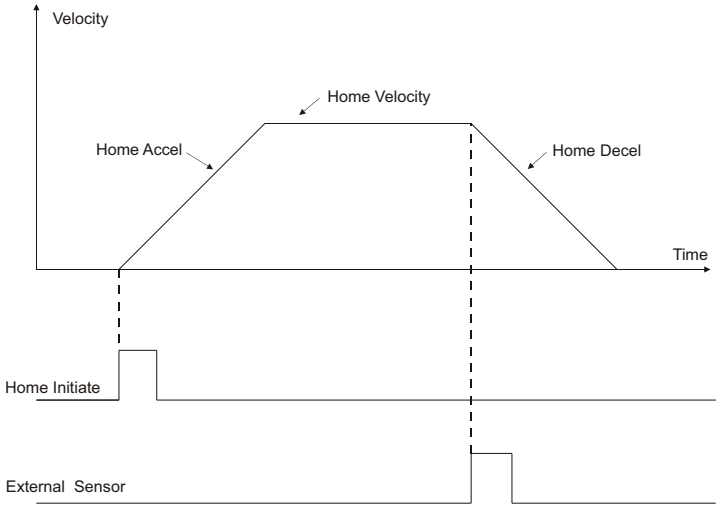


Figure 6-4: Home to Sensor Profile (Calculated Offset)

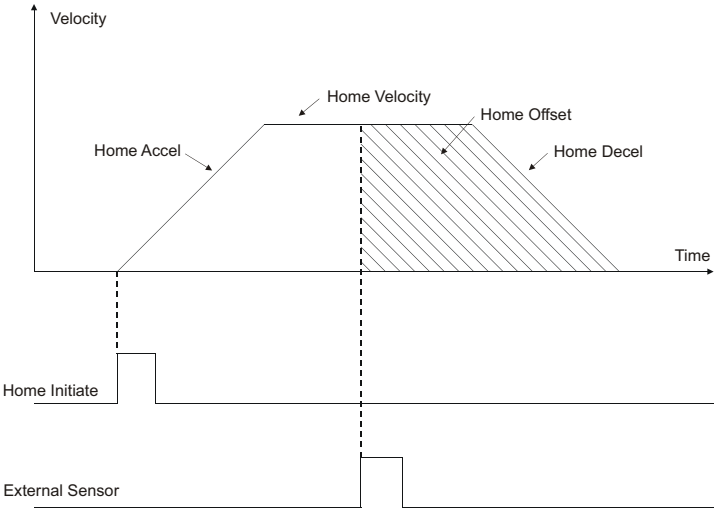


Figure 6-5: Home to Sensor Profile (Specified Offset)

6.2.3 Home to Sensor then Marker

The Home to Sensor then Marker profile is a combination of the two home types described above. Because in many applications the load will be more than one revolution away from the desired home position, a Home to Marker cannot be used because the marker activates once every revolution of the motor. Therefore, an external sensor is mounted on the machine to determine the home position. The PTi210 module allows the user to home first to the external sensor, followed by a home to the next marker pulse. The Home to Sensor then Marker combines the accuracy of homing to the encoder marker with the flexibility of homing to an external sensor.

The Home to Sensor then Marker routine begins by accelerating up to the Home Velocity. The motor then continues at the Home Velocity until the external sensor activates. After the sensor activates, the motor continues at the Home Velocity until the next rising edge of the encoder marker is detected. Once the encoder marker activates, the motor either begins to decelerate immediately, or continues for a specified offset distance before stopping. Figure 6-6 and Figure 6-7 show examples of the Home to Sensor then Marker profile, with calculated offset and specified offset respectively.

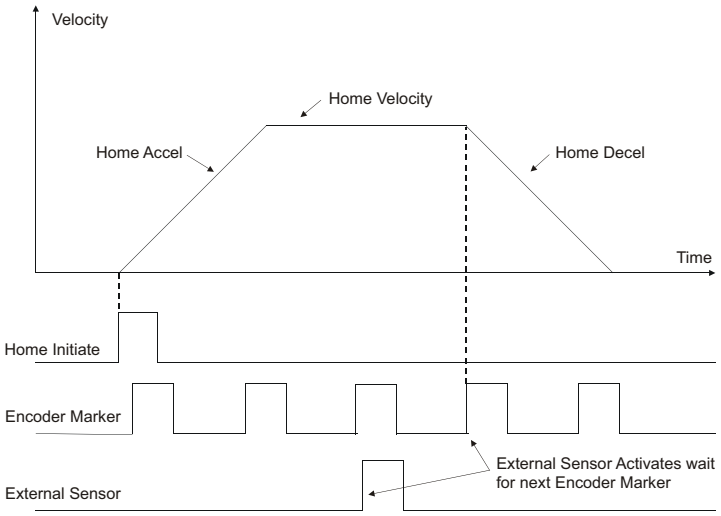


Figure 6-6: Home to Sensor then Marker Profile (Calculated Offset)

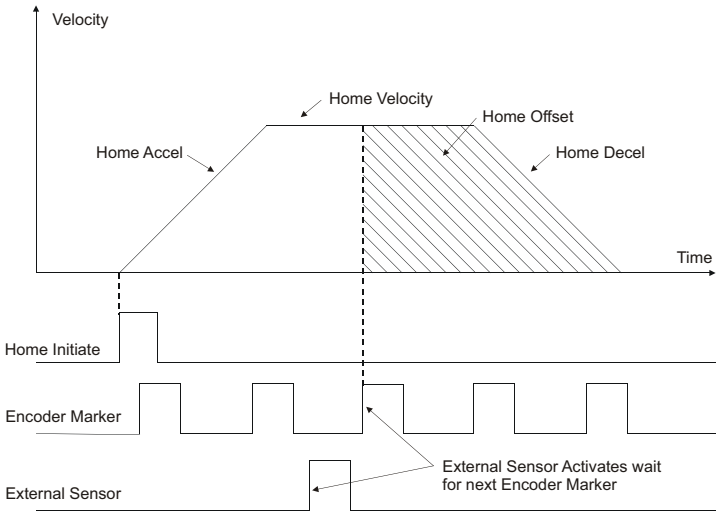


Figure 6-7: Home to Sensor then Marker Profile Specified Offset)

6.2.4 If On Sensor Options

In a Home to Sensor, or Home to Sensor then Marker profile special conditions must be created to handle the situation when the External Sensor is already active when Home is initiated. Different users want the system to act differently in this condition, so the PTi210 module has pre-programmed solutions for this condition. The two options are explained below.

Back Off Before Homing

If the Home Sensor is active when home is initiated, one option is to move the motor in the direction opposite from the programmed direction until the home sensor deactivates, and then again in the positive direction until the home sensor activates. Figure 6-8 shows the Home to Sensor profile when the Home Sensor is active when the home is initiated and Back Off Before Homing is selected.

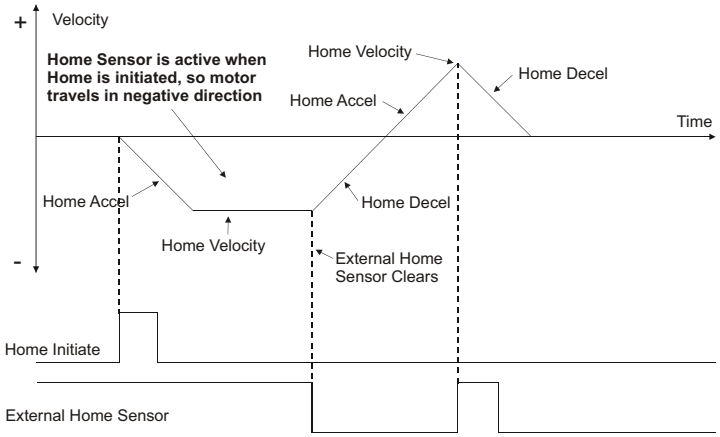


Figure 6-8: Home to Sensor (Back Off Before Homing)

Go Forward To Next Sensor

On some machines, motion in a certain direction may be prohibited due to mechanical design. In this case, the Back Off Before Homing option may not be practical. In this case, if the Home Sensor is active when the Home is initiated, the motor continues in the programmed direction until the next "Rising Edge" of the external home sensor is detected. Figure 6-9 shows the Home to Sensor profile when the Home Sensor is active when home is initiated and Go Forward To Next Sensor is selected.

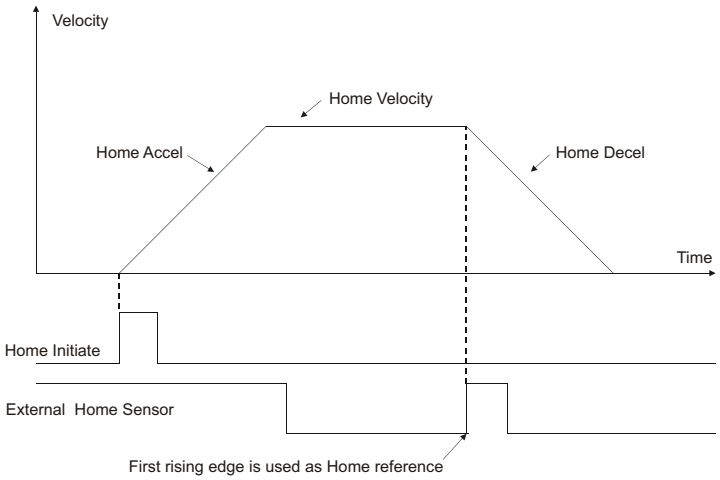


Figure 6-9: Home to Sensor (Go Forward To Next Sensor)

6.3 Index

An Index profile is used to move the motor a precise distance or to a specific position. There are many different applications that can be solved using different combinations of Index types. The five major types of Indexes are Absolute, Incremental, Registration, Rotary Plus, and Rotary Minus. Each of these Index types are described in detail below.

6.3.1 Absolute Index

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 dependant 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.

Figure 6-10 and Figure 6-11 show examples of an Absolute Index profile.

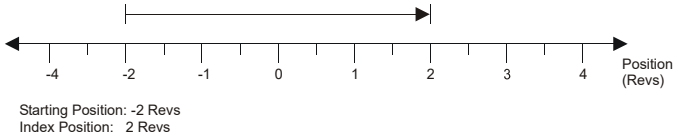


Figure 6-10: Absolute Index Profile (Example 1)

Figure 6-11: Absolute Index Profile (Example 2)

NOTE

If Rotary Rollover is active, an Absolute Index will take the shortest path to the specified index position.

6.3.2 Incremental Index

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 dependant on the starting position before the Index was initiated.

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

Figure 6-12 and Figure 6-13 show examples of an Incremental Index profile.

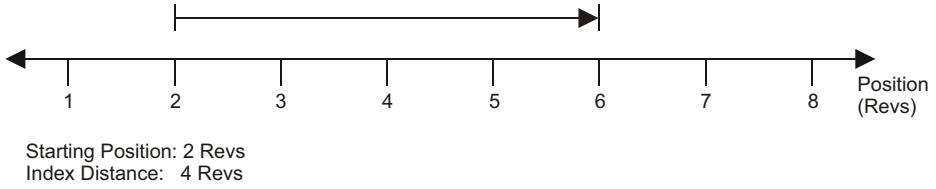


Figure 6-12: Incremental Index Profile

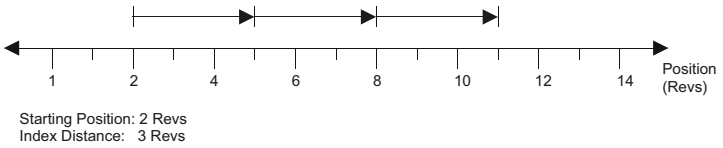


Figure 6-13: Incremental Index - Repeated 3 Times

6.3.3 Correction Index

A Correction index is intended to continuously run on the second profile correcting any position drift. It will adjust the motor position based on changes to it's index.#.dist parameter. The Correction indexes use incremental distance values. Updates to the index distance while the correction index is executing will take effect immediately by recalculating the index on the fly. In other words, if this index is in progress and the distance value is changed, the move profile instantaneously recalculates based on the index's current position, speed and acceleration. Once the Correction index is initiated it will remain active until stopped by the user with the Profile.#.MotionStop function.

Example:

Correction index distance sources are user program calculations, fieldbus inputs or analog input values. The index 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 the Index.#.Dist parameter, the index's distance value will be updated by the Analog Input. This can be set to a automatic refresh using the Analog Input view

6.3.4 Posn Tracker Cont Index and Posn Tracker Once Index

Posn Tracker Cont and Posn Tracker Once are indexes which expect their position values to be dynamically changed while executing. Position Tracker indexes use absolute position values. Posn Tracker Cont index once initiated, will remain active until stopped by the user with the Profile.#.MotionStop function. The Posn Tracker Once index will accept position changes until the target position is reached, at which point the index is complete.

The index's position value can be updated via fieldbus, by simply writing to the index position parameter. Posn Tracker Indexes are used to follow dynamic changes to the end point of the index prior to and during the index motion. If the analog input's Destination is set to an Index.#.distance, the index's position value will be updated by the Analog to Position scaling found in the Analog Input view.

Posn Tracker also accepts on the fly changes to index velocity, acceleration and deceleration. The index is recalculated on the next trajectory update.

6.3.5 Registration Index

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 begins to decelerate immediately, or it continues at velocity for a specified offset distance.

Registration to Sensor

In a Registration Index with Sensor defined as the registration signal, the index travels at velocity until an external sensor or switch activates. The sensor or switch must be wired to a digital input on the PTi210 module, Unidrive M/Digitax HD drive, or SI-I/O module. To get the highest accuracy for the Registration to Sensor, a PTi210 module digital input must be used to take advantage of the high-speed capture capability. Three Figures below show examples of a Registration Index to Sensor using different Offset values.

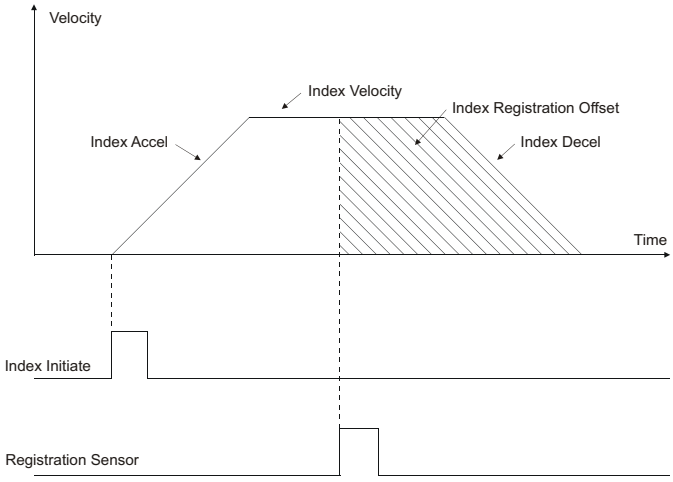


Figure 6-14: Registration to Sensor Profile (Offset > 0)

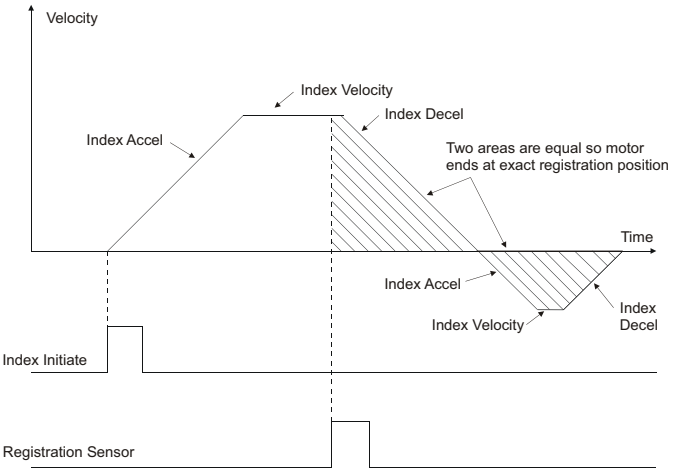


Figure 6-15: Registration to Sensor Profile (Offset = 0)

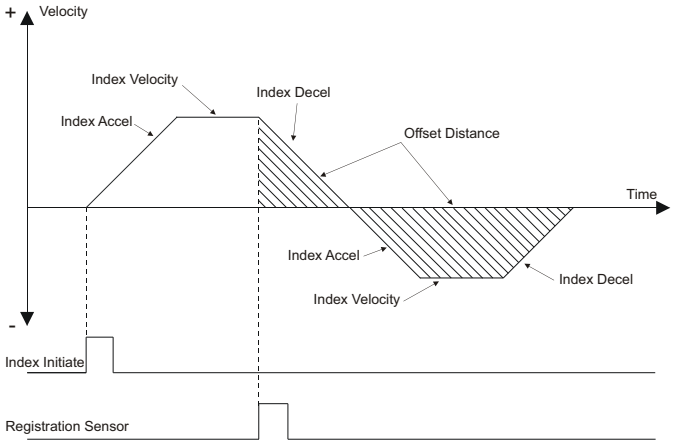


Figure 6-16: Registration to Sensor Profile (Offset < 0)

6.3.6 Rotary Plus Index

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). Figure 6-17 compares a Rotary Plus Index to an Absolute Index (Rotary Rollover is enabled).

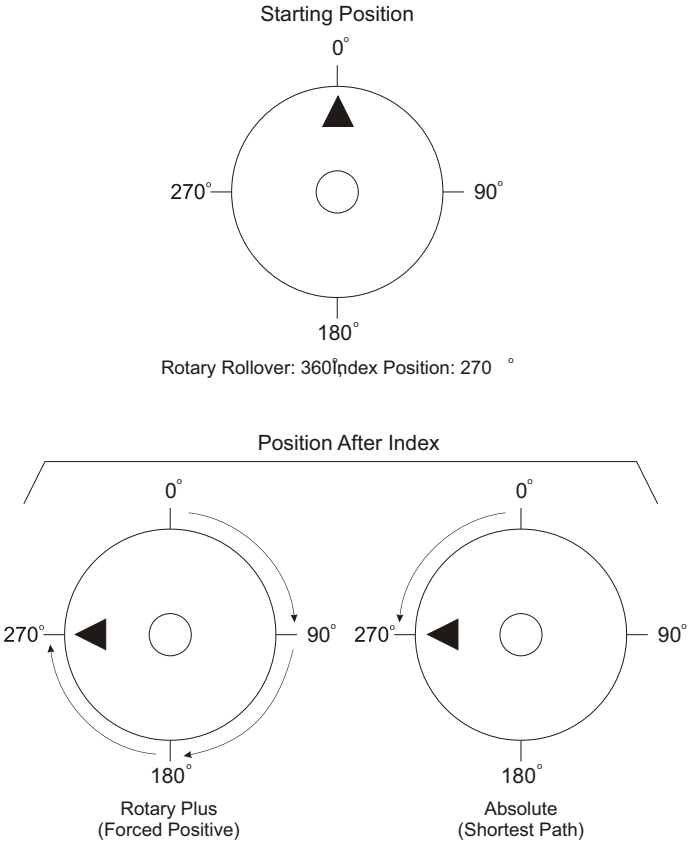


Figure 6-17: Rotary Plus Index Profile

6.3.7 Rotary Minus Index

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). Figure 6-18 compares a Rotary Minus Index to an Absolute Index (Rotary Rollover is enabled).

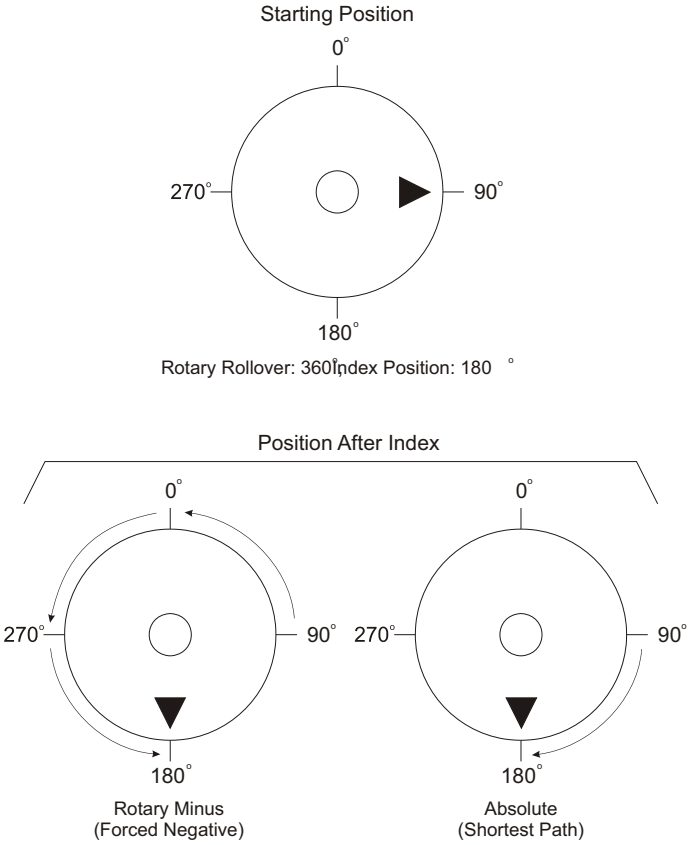


Figure 6-18: Rotary Minus Index Profile

6.3.8 Timed Index

A Timed Index is not a specific type of index like the other types listed above. Timed Index is simply an option for the other types of indexes. Each index type (other than registration indexes) can be configured as a Timed Index.

In many applications, the user knows how far the load must move in a certain period of time. Rather than making the user calculate an acceleration, velocity, and deceleration so that an index takes the right amount of time, the PTi210 module allows the user to enter the distance and the time instead.

In a Timed Index, the user provides the distance and time, and the firmware automatically calculates the accel, velocity, and decel to finish in the right period of time. Figure 6-19 shows an example of a Timed Index profile.

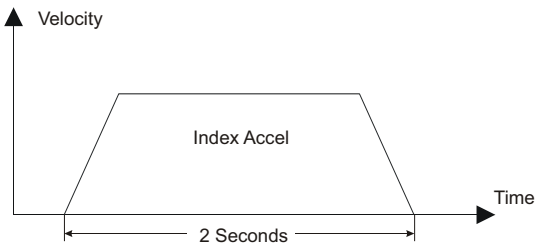


Figure 6-19: Timed Index Profile

In some cases, an index time is calculated based on other parameters in a user program. To avoid possible machine damage, the user can specify maximum values for accel, velocity, and decel. Therefore, when the PTi210 module firmware calculates accel, velocity, decel, they will never exceed the maximum values specified by the user, In this case, where the calculation is limited by a maximum value, the index will not finish in the specified time. If this happens, a parameter called Index.ProfileLimited will activate. It will remain active until cleared by the Index.ResetProfileLimited destination.

6.4 Gearing

The Gear motion profile is used to slave the motion of the motor to the motion of a master axis at a specified ratio. Gearing is often referred to as "electronic line shafting" or "electronic gearing". To gear a follower axis to a master axis, a ratio (called the gear ratio) must be specified. The Gear Ratio defines the relationship between the master and follower motion.

The ratio is calculated as follows:

Gear Ratio =	# of Follower Distance Units
	1 Master Distance Unit

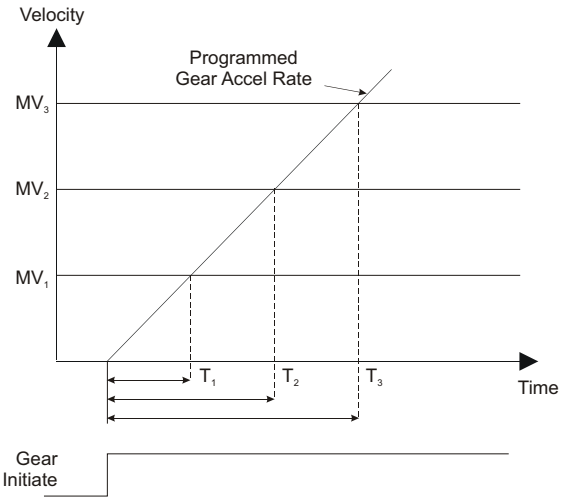
The ratio is the number of follower distance units to move the motor per master distance unit of travel. Follower Distance Units are configured on the User Units view. Master Distance Units are configured on the Master Setup screen.

The gear ratio can be positive or negative and is a signed 32-bit parameter. The resolution of the parameter is determined by the number of decimal places configured for the Master Velocity Units on the Master Setup screen.

By default, gearing does not use acceleration or deceleration ramps with respect to the master encoder. This means that once gearing is activated, peak torque is available to try to achieve the specified gear ratio. Therefore, if the master axis is in motion when gearing is activated, the control loop will attempt to achieve the programmed ratio within one update without programmed acceleration. Analogously, when gearing is deactivated, the motor will use peak torque to bring the motor to a stop without a deceleration ramp.

Acceleration and Deceleration ramps can be enabled by the user. If enabled, the Accel and Decel ramps are specified in units of Follower Units / Velocity Time Base / Acceleration Time Base. Note that this is a Realtime ramp. Therefore, the time that it takes to reach the programmed ratio depends on how fast the master is traveling when gearing is activated.

Figure 6-20 demonstrates that the faster the Master Velocity, the longer it will take to reach the programmed ratio. If the Master Axis is not moving when gearing is initiated, then the follower locks into its programmed gear ratio instantly (no acceleration time required).



$T_1 < T_2 < T_3$ The greater the master velocity (MV) the longer it takes to accelerate to the Gear Ratio.

Figure 6-20: Gear Acceleration Diagram

The GearRatio can be changed on the fly (while gearing is active and in motion), but acceleration or deceleration must be enabled to use ramps to achieve the new ratio. If gearing accel and/or decel ramps are not enabled, the motor will attempt to achieve the new ratio in one trajectory update.

6.5 Camming

Electronic cams provide a non-linear motion function for a single axis. The basic motion can best be illustrated in Figure 6-21 below of a mechanical cam and cam follower (or slave).

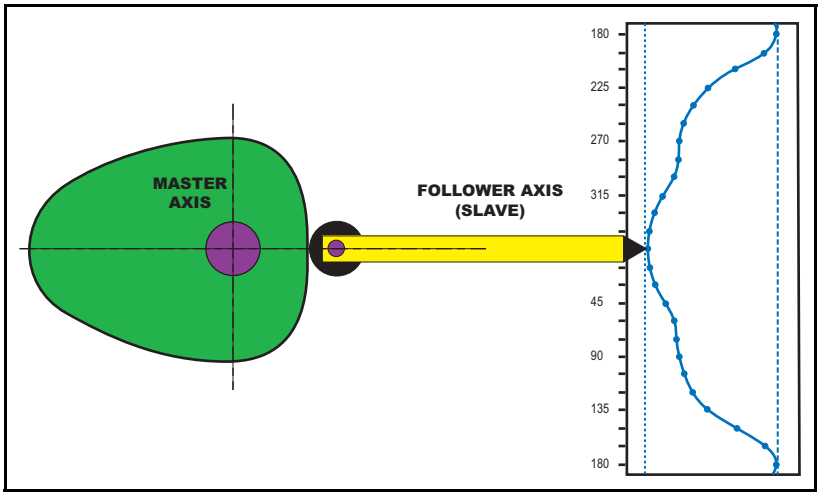


Figure 6-21: Mechanical Master and Slave Follower

As the master axis (the cam lobe) rotates, the follower axis produces a non-linear motion profile. This same profile can then be produced with a single motor driving a linear axis programmed with an electronic cam.

The cam motion object uses a master/follower principle in a synchronized mode and also has a follower with Realtime mode that allows the follower to travel through its cam table without a physical master axis moving.

Control Techniques provides a Cam as a collection of cam table(s) that can be used individually of chained together to form a full sequence of motion. Each cam table is a user specific sequence of movements whereby the user can specify the master and follower movement along with the interpolation type. Coupled with a user program to monitor the flow, the motion can dynamically be altered by changing the cam table chains selecting a different sequence of tables. You can further adjust the flow by dynamically changing the cam tables themselves or using a cam table time base index to adjust time or distance.

As an alternative, the cam is initiated in the same manner as jogs, home and indexes.

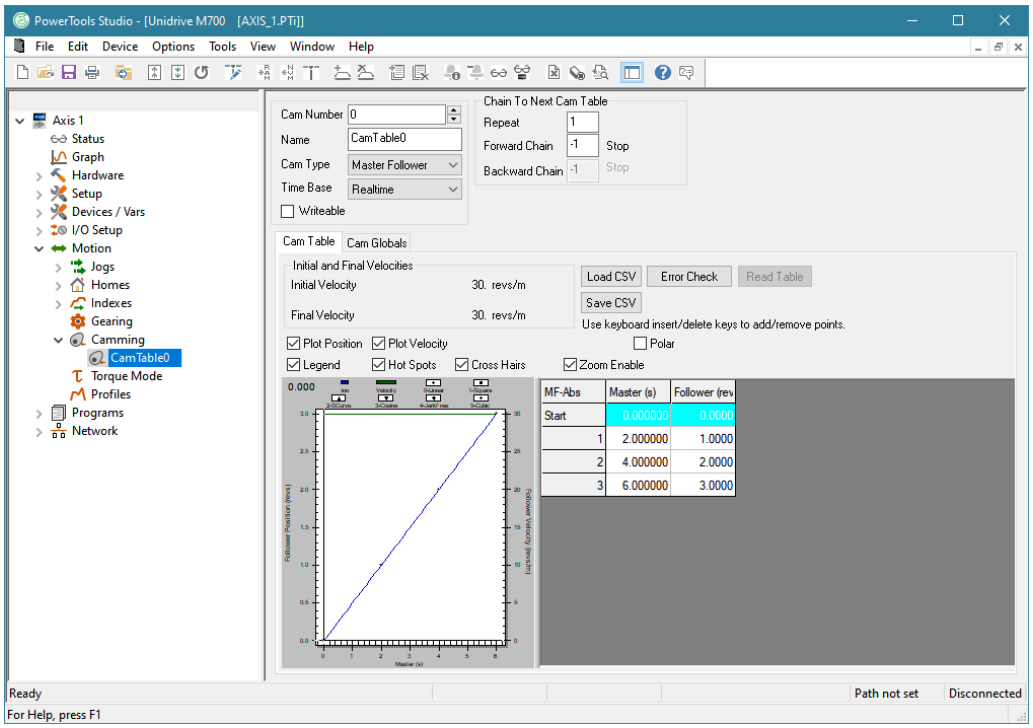


Figure 6-22: Camming View

6.6 Motion Timebase (Realtime vs. Synchronized)

The Timebase for a profile determines what parameter is used as the denominator for velocity and acceleration units. The default Timebase for all motion types (other than gear) is Realtime.

A Timebase of Realtime specifies that the denominator for velocity and acceleration units are units of actual time: Minutes, Seconds, or Milliseconds (defined on the User Units View). Therefore, units of velocity for a realtime profile are as follows:

Follower Distance Units / Velocity Timescale.

And units of acceleration for a realtime move are

Foll Distance Units / Velocity Timescale / Accel Timescale.

A few examples of these units are as listed below.

Realtime

Follower Distance Units	Master Distance Units	Vel. Timescale	Accel Timescale	Velocity Units	Accel/Decel Units
Inches	N/A	Sec	Sec	Inches/Sec	Inches/Sec/Sec
Revs	N/A	Min	Sec	Rev/Min	Rev/Min/Sec
Degrees	N/A	Sec	msec	Degrees/Sec	Degrees/sec/msec

Based on these units, we see that velocity and acceleration of the motor are dependent upon actual time (minutes or seconds).

Selecting a Timebase of Synchronized means that all units of velocity and acceleration are a function of Master Distance rather than time. Therefore, the motor velocity acceleration and position are all functions of the position and velocity of the master axis. The units for velocity of a Synchronized move are as follows:

Follower Distance Units / Master Distance Unit

Therefore, the user specifies the number of follower distance units that the motor will travel per Master Distance Unit. If the distance units for master and follower are the same, then the user in effect specifies a ratio for the velocity.

The acceleration units for a synchronized move are again a function of Master Distance. Acceleration and Deceleration units are as follows:

Follower Distance Units / Master Distance Unit / Master Distance Unit

A few examples of Synchronized motion units are listed in the following table.

Synchronized

Follower Distance Units	Master Distance Units	Vel. Timescale	Accel Timescale	Velocity Units	Accel/Decel Units
Inches	MstrInch	N/A	N/A	Inches/ MstrInch	Inches/MstrInch/ MstrInch
Revs	MstrRev	N/A	N/A	Revs/ MstrRev	Revs/MstrRev/MstrRev
Degrees	MstrInch	N/A	N/A	Degrees/ MstrInch	Degrees/MstrInch/ MstrInch

6.7 Summing Multiple Profiles

Motor motion or "Axis" motion may be generated from either of two Profiles: Profile.0 and Profile.1. Each of these Profiles can run any type of motion (i.e. Home, Index, Jog, Gear) at any time. Both of the Profiles can generate motion simultaneously. For example while Gearing, an incremental index can be initiated "on top" of the Gear velocity using two separate profiles. The distance and velocity of the two profiles are summed to generate the overall position command and velocity command for the motor.

In order to run motion on both Profiles simultaneously, a program must be used. To specify which profile a motion object runs on, the On Profile instruction is used. The default Profile is Profile.0 and therefore it is unnecessary to specify On Profile.0 in user programs. If no Profile is specified, the default profile is used.

All motion run from the Assignments view is automatically run on Profile 0. It is not possible to change the Profile on which motion run from the Assignments screen operates. Therefore, in order to run motion from both the Assignments screen and from a program simultaneously, motion initiated by the program must be run on Profile 1.

Figure 6-23 shows an example to two separate profiles (Index 0 and Index 1). Each profile is shown individually, and then a summed profile diagram is shown to demonstrate what the overall profile looks like when the profiles are summed.

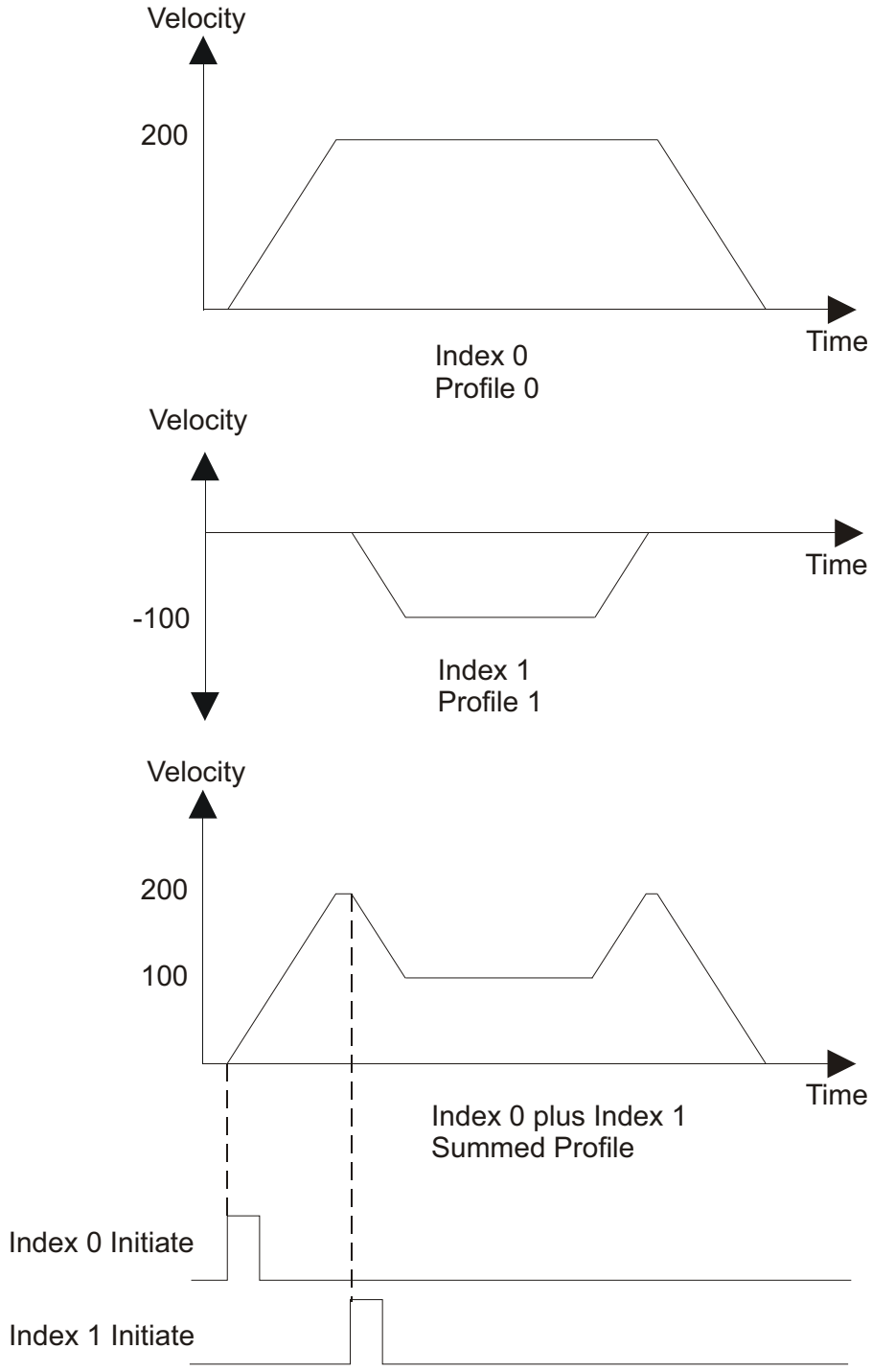


Figure 6-23: Two Indexes Summed Profile

7 How I/O Works

7.1 I/O Scan

When used to activate PTi210 functions, the Unidrive M/Digitax HD and the PTi210 I/O are scanned every trajectory update. If the Unidrive M/Digitax HD I/O is used to trigger functions in the base drive itself (i.e. Threshold Detectors, Programmable Logic, Binary Sum, etc.), then the Unidrive M/Digitax HD I/O is only scanned every 4 milliseconds. The scan rate is different when the drive I/O is used with PTi210 functions because the PTi210 module processor scans the I/O faster than the base drive firmware does.

7.2 PTi210 I/O

There are three digital inputs and two digital outputs on the PTi210 connector. These I/O are scanned normally at the trajectory update rate (user configured). If an input is assigned to a function that does not use captured data, then it will be updated every trajectory update.

The PTi210 I/O can also be captured with 1 microsecond accuracy by using the High Speed Capture object in the PTi210 module. If an input is assigned to a function that uses captured data (i.e. Index.#.Initiate), then the input will automatically be captured with 1 microsecond accuracy, and the data is passed to the destination.

To use the PTi210 I/O in the PowerTools Studio Assignments view, the three Inputs are called ModuleInput.# and the two Outputs are called ModuleOutput.# (where # represents the specific I/O number).

7.3 Unidrive M700/M701 I/O

The Unidrive M700/M701 have three I/O points that are configured by the user to be Inputs or Outputs, along with three dedicated Inputs. The scan rate of the I/O on these drives depends on what they are being used for. If the I/O are being used by the PTi210 functions, then they are updated every trajectory update (user configured), otherwise the update rate depends on the destination parameter, for more information please refer to the relevant drive user guide.

To use the Unidrive M700/M701 I/O in the PowerTools Studio Assignments view, the three Input/Output lines are called DriveIO.#.In or DriveIO.#.Out depending on how they are configured (where # represents the specific I/O number). The configured Input/Output lines will be listed both on the Inputs group of Sources, and the Outputs group of Destinations. Therefore, it is important for the user to know how the Inputs/Outputs have been configured. The dedicated Inputs are referred to as DriveInput.# (where # represents the specific I/O number).

7.4 Unidrive M702/Digitax HD I/O

The Unidrive M702 and Digitax HD drives do not have any user programmable I/O but have two dedicated digital inputs and two dedicated digital outputs. The scan rate of the I/O depends on what they are being used for. If the I/O are being used by the PTi210 module functions, then they are updated every trajectory update (user configured), otherwise the update rate depends on the destination parameter, for limit switch functions (Pr **06.035**/Pr **06.036**) this will be 250 µs, for the drive STO input (Pr **06.029**), this will be 600 µs, and all other destinations will be 2 ms.

To use the Unidrive M702/Digitax HD I/O in the PowerTools Studio Assignments view, the two input lines are called DriveInput.# and the two output lines are called DriveOutput.#, where the # represents the specific I/O number. The configured Input lines will be listed on the Inputs group of Sources, and the Outputs listed on the Outputs group of Destinations.

7.5 SI-I/O Module I/O

The SI-I/O module has four I/O points that are configured by the user to be either Digital Inputs or Digital Outputs, along with three I/O points that are configured for either Analog or Digital Inputs, and one I/O point that can be configured either as a Digital Input or an Analog Output. The scan rate of the I/O on the SI-I/O module depends on how many SI-I/O modules are being used. If using one SI-I/O module, the I/O on the SI-I/O module is updated every 1 millisecond. If a second SI-I/O module is used, the I/O on both SI-I/O modules are updated every 2 milliseconds.

To use the SI-I/O I/O in the PowerTools Studio Assignments view, the Input/Output lines are called SlotX.DIO.#.In or SlotX.DIO.#.Out depending on how they are configured (where X represents the slot number the module is located in, and # represents the specific I/O number). The configured Input/Output lines will be listed both under the Inputs group of Sources, and the Outputs group of Destinations. Therefore, it is important for the user to know how the Inputs/Outputs have been configured. The dedicated Inputs are referred to as SlotX.Input.# (where X represents the slot number the module is located in, and # represents the specific I/O number).

Because of the slower scan rate of the SI-I/O module inputs and outputs, it is recommended that these I/O not be used for critical motion functions (e.g. Home Switches, Registration Sensors, Travel Limits, PLSs, etc.).

7.5.1 SI-I/O Module digital I/O

The SI-I/O digital I/O setup uses the Collective Digital IO (CDIO) Mappings Mode (Pr **S.01.019**) with Pr **20.040** used as the digital input destination register and Pr **20.039** as the digital output source register.

The digital input destination register setup is shown in the following table.

Table 4-1 Digital input destination register setup

I/O Name	Terminal Number	Pr 20.40	
		Bit	Value
Digital I/O 1	2	0	1
Digital I/O 2	3	1	2
Digital I/O 3	4	2	4
Digital I/O 4	5	3	8
Digital Input 5	7	4	16
Digital Input 6	8	5	32
Digital Input 7	9	6	64
Digital Input 8	11	7	128

The digital output source register setup is shown in the following table.

Table 4-2 Digital output source register setup

I/O Name	Terminal Number	Pr 20.39	
		Bit	Value
Digital I/O 1	2	0	1
Digital I/O 2	3	1	2
Digital I/O 3	4	2	4
Digital I/O 4	5	3	8
N/A	N/A	4	N/A
N/A	N/A	5	N/A
N/A	N/A	6	N/A
N/A	N/A	7	N/A
Relay 1	21, 22	8	256
Relay 2	22, 23	9	512

7.6 SI-I/O 24 Plus Module digital I/O

The SI-I/O 24 Plus option module provides additional digital I/O and encoder interface, it is primarily designed to facilitate migrating an existing application using the Epsilon drive to the Digitax HD drive.

The digital I/O consists of sixteen digital inputs and eight digital outputs available on a 44 way high density 'D' type connector.

The SI-I/O 24 Plus digital I/O setup configuration is available in the Hardware view when the option module is selected for a particular slot.

More information on the SI-I/O 24 Plus option module can be found in the relevant User Guide.

8 Configuring an Application

8.1 Introduction

All applications in the PTi210 are configured using PowerTools Studio configuration software. For specific questions on PowerTools Studio operation, please refer to *PowerTools Studio Software* on page 9 of this manual.

The hierarchy tree in PowerTools Studio gives the user a basic template on how to configure an application. The user should start at the top of the hierarchy tree, filling out necessary parameters on the different views, working all the way to the bottom of the hierarchy tree. Once the user is more familiar with the software, they may choose to skip various views.

By following the hierarchy tree from top to bottom, the user will start by configuring the hardware that the PTi210 will be used with (Drive Type, Motor Type, etc.). The user will then configure the different Setup views for all the different configuration parameters (i.e. User Units, Tuning, Torque Limits, etc.). Next, the user configures any devices or variables that are needed in the application. Then the Digital and Analog I/O for the drive, PTi210, and any SI-I/O module. The user then defines all of the different motion profiles (Jogs, Homes, Indexes, Gearing, Camming) to be used in the application. After the hardware, setup parameters, devices/vars, I/O, and motion have been configured, the user writes programs to tie the entire application together. Once the programs are complete, the user can create the interface for Modbus communications to the different drive/module parameters.

The maximum application configuration size is limited to 64 kB.

8.2 Define Hardware

8.2.1 Drive/Encoder/Motor

The Drive/Encoder/Motor view allows the user to configure the Drive Model, Encoder Type and Motor Type being used in the application. Figure 8-1 shows the Drive/Encoder/Motor view.

Drive Model

The Drive Model list box show the available drive models.

Motor Type

The Motor Type list box shows the available motors in the .ddf file.

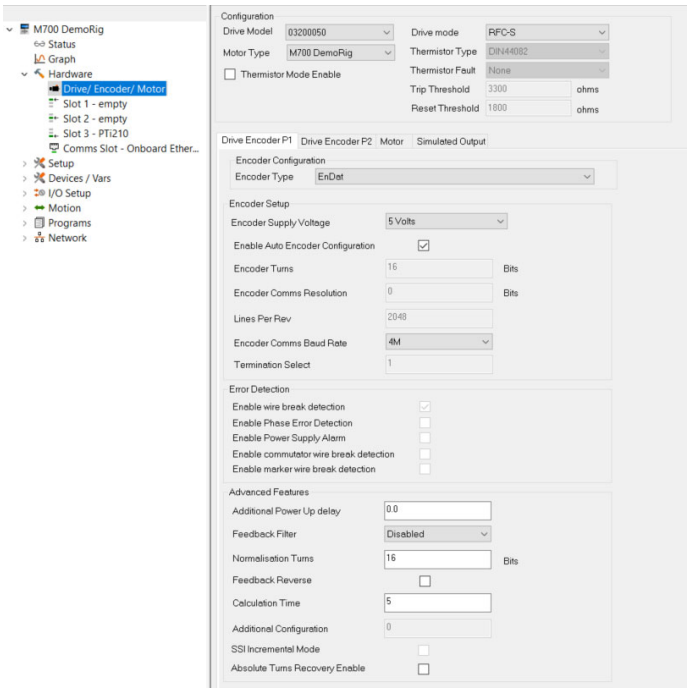


Figure 8-1: Drive/Encoder/Motor View -PTi210/Unidrive M/Digitax HD Setup

Motor Configuration (RFC-A/RFC-S) - PTi210/Unidrive M/Digitax HD

Drive mode Selection List Box

The Unidrive M/Digitax HD is capable of running in different operating modes. These modes are as follows:

1. Open Loop Mode (Volts/Hz)
2. Closed Loop RFC-A (induction) Mode
3. Closed Loop RFC-S (servo) Mode
4. Regen Mode

However, the PTi210 module is only able to control the drive while in modes 2 and 3 listed above (Closed-loop RFC-A Mode and Closed-loop RFC-S). The user must select which operating mode their application will use in the Drive mode list box. Based on the setting of this parameter, many different views and tabs within PowerTools Studio will change to use the appropriate parameters. When this selection is changed, the user will be taken to the Motor Tab on the Drive/Encoder view so that the necessary motor parameters can be entered.

The PTi210 module will also automatically change the drive mode to match the mode selected in this list box on every power-up or warm start.

Thermistor Mode Enable

If the motor attached to the drive has a thermistor or thermal switch for thermal protection, then this check box should be selected. If the motor does not have a thermistor or thermal sensor, then the check box should be clear.

If the check box is selected, the drive monitors the thermal device to determine if the motor is over-heated, in which case a Thermistor trip will activate on the drive.

Internally, this check box is used to configure the thermistor feature inside the drive. The functionality is as follows:

If check box is selected:

The PTi210 sets the Unidrive M700/M701/M702 “Analog Input 3 Mode” to Thermistor without Short-Circuit Detection Mode on power up by setting Pr **07.015** = 8.

If check box is cleared:

For Unidrive M the PTi210 module checks “Analog Input 3 Mode” on power-up by reading Pr **07.015**. If Pr **07.015** = 8, then the PTi210 changes the mode to Voltage Mode by setting Pr **07.015** = 6 on the Unidrive M700/M701, and on the Unidrive M702 the thermistor mode is disabled by setting Pr **07.015** = 10. If Pr **07.015** <> 8 then the PTi210 does not modify the value of Pr **07.015**.

For more details on functionality of thermistor mode, see the relevant drive user guide.

NOTE

This parameter only applies to thermal devices connected to the Unidrive M/Digitax HD Encoder Port or Pin 8 on the Unidrive M Control Connectors and NOT to thermal devices connected to the encoder port on a SI-Universal Encoder module. Users must employ parameter **1M.123** on the encoder module if the thermal device is attached to the module.

Braking Resistor Temperature Monitor Disable

When this check box is clear the internal braking resistor temperature will be monitored. Select the check box if no internal braking resistor is installed in the drive or to disable the Brake R Too Hot trip (see Figure 8-2).

Motor Tab

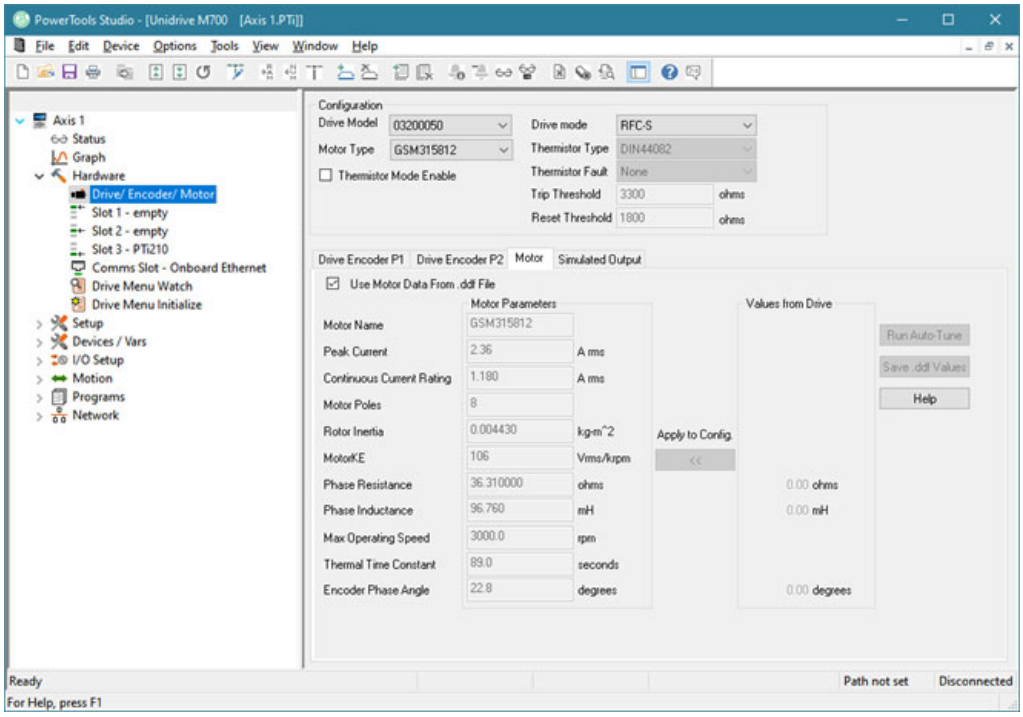


Figure 8-2: Drive/Encoder/Motor View - Motor Tab

The Motor Tab on the Drive/Encoder view is used for many different functions:

1. To see/verify the motor data for a standard motor that has been selected
2. To create a new motor entry in the .ddf file
3. To Run an Auto-Tune
4. To store Auto-Tune results into an existing configuration

The primary function of this tab is to define the parameters for the given motor that is to be connected to the drive. Depending on the selected drive operating mode (RFC-A or RFC-S), the motor parameters on this tab will be very different. This is because servo motors and induction motors are very different in design and operation.

Following is a description of all the different functions on the Motor Tab.

Use Motor Data From .ddf File Check Box

When selecting a motor for use with the drive and PTi210 module, the user has two basic options:

1. Use a motor that already exists in the specified standard motor file **(Options > Preferences > PowerTools Options > MotorDDF)** or the specified custom motor file **(Options > Preferences > PowerTools Options > MotorDDF)**.
2. Create a custom motor that has not been used before.

The default standard motor filename is 'StdMotor.ddf' and the default custom motor filename is 'UserMotor.ddf'.

When selecting option 1 from above (use an existing motor), the user simply selects one of the motors from the Motor Type list at the top of the Drive/Encoder view. Once the user selects a motor from the Motor Type list, the data for that motor is read from the pertinent .ddf file and then is displayed in the Motor Parameters column on the Motor tab (see Figure 8-2). The parameters in this column will be dimmed and unavailable because the motor information comes directly from the .ddf file.

If the user wishes to edit one or more of the parameters read from the .ddf file, it is necessary to clear the “User Motor Data From .ddf File” check box. Clearing the check box will break the “link” between the motor data displayed on this view, and the motor data in the .ddf file. This is necessary because as soon as the user changes any of the values, it no longer matches the .ddf file, and is now in effect a “custom motor”. When the “Use Motor Data From .ddf File” check box is cleared, all of the values in the Motor Parameters column will become available, and the Motor Name will be changed to “New Motor” so that there is no association with the existing motor that was previously selected. The user can now change any of the values as desired and give the motor a new name. Once the values have been changed, the motor data only exists within the active configuration. To save the new values into the .ddf file, the user must click the ***Save .ddf Values*** button on the right side of the tab.

8.2.2 Motor Parameters Column

Motor Parameters column is a column of data displayed on the Motor tab within the Drive/Encoder view (See Figure 8-2). This column of data contains the values for each of the motor data parameters. The values in this column are unavailable if the

"Use Motor Data From .ddf File" check box is selected. This means that since the data is associated with the .ddf file, it cannot be changed. The values in this column become available when the "Use Motor Data From .ddf File" check box is cleared. The user can then change one or more of the parameter values because they're no longer linked to data in the .ddf file.

If the user does edit motor parameter values on this tab, then those values are only stored within that particular configuration file. In order to save the values to the .ddf file, the user must click the "***Save .ddf Values***" button on the right side of the tab.

The parameters displayed in the Motor Parameters column will change depending on the drive operating mode selected (RFC-A or RFC-S) in the Drive mode list box on the Drive/Encoder view.

8.2.3 Servo Motors (RFC-S Mode)

When RFC-S Mode is selected from the Drive mode list box or the application file is for a drive configured in RFC-S mode, the user must create a new servo motor data file by editing the following details in the Motor Parameters column.

Motor Name

The motor name is limited to 12 characters and must begin with an alpha character (non-numeric character). This is the motor name that will appear in the "Motor Type" list box on the Drive/Encoder view in PowerTools Studio.

Peak Current

Specifies the peak current allowed by the motor. The range is 0.000 to 99999.999 Amps (rms). The motor manufacturer typically provides the peak current data.

If a system is "drive limited" (meaning that the motor can handle more current than the drive can deliver), the peak current actually used by the system may be lower than the value specified here.

Continuous Current

Specifies the continuous current allowed by the motor. It is used to determine the drive's continuous current and peak current limits. The drive can also limit the continuous current to the motor based on the drive capacity. The range is 0.000 to 99999.999 Amps (rms). The motor manufacturer typically provides the continuous current data.

If a system is "drive limited" (meaning that the motor can handle more current than the drive can deliver), the continuous current actually used by the system may be lower than the value specified here.

Motor Poles

Specifies the number of magnetic poles on the motor. The supported values are 2 to 480. The motor manufacturer typically provides the motor pole information.

Rotor Inertia

This parameter specifies the inertia of the motor rotor. The range is 0.00000 to 1000.00000 kg*m^2. The PTi210 module uses this parameter to interpret the "Inertia Ratio" parameter found on the Tuning view. "Inertia Ratio" is specified as a ratio of reflected load inertia to motor inertia.

Motor Ke

Specifies the Ke of the motor. The units are Vrms/ kRPM. The line-to-line voltage will have this RMS value when the motor is rotated at 1000 rpm. The range is 0 to 100000. The motor manufacturer will typically provide the Ke data.

Phase Resistance

Specifies the phase resistance of the motor. You can determine this value by measuring the resistance between any motor phase and ground with an ohm-meter. The range is 0.000000 to 500.000000 Ohms. The motor manufacturer will typically provide the phase resistance data.

NOTE

This parameter is not the same as the phase resistance parameter found in the .ddf file. Do not copy this value from stdmotor.ddf.

Phase Inductance

Specifies the phase inductance of the motor. This is the inductance measured from phase to ground and NOT phase-to-phase. The range is 0.00 to 5000.00 mH.

Max Operating Speed

Specifies the maximum operating speed of the motor. It is used by the drive to limit the Velocity Command. The valid range for this parameter is 0.00 to 33000.00 rpm.

Thermal Time Constant

Specifies the Thermal Time Constant of the motor. This parameter is used by the drive for thermal protection of the motor. The drive models the temperature of the motor using a formula that generates an overload accumulator value. The formula is a function of the Thermal Time Constant. When the accumulator reaches 100 %, the drive can trip or foldback depending on other drive settings. For more information on the Thermal Time Constant, please refer to the relevant drive *User Guide* (Pr **04.015**, **04.016**, and **04.019**).

Encoder Phase Angle

This is the angle between rising edge of the V commutation signal and the peak of VVW back EMF signal when rotating the motor in the clockwise direction.

Reference for the clockwise direction is looking at the front end of the motor shaft. See relevant drive *User Guide* for more information (Pr **03.025**).

8.2.4 Induction Motors (RFC-A Mode)

When the user has selected RFC-A Mode from the Drive mode list box, the user must create a new induction motor data file by editing the following details in the Motor Parameters column.

Motor Name

The motor name is limited to 12 characters and must begin with an alpha character (non-numeric character). This is the motor name that will appear in the Motor Type list box on the Drive/Encoder view in PowerTools Studio.

Peak Current

Specifies the peak current allowed by the motor. The valid range is 0.000 to 99999.999 Amps (rms). The motor manufacturer may or may not provide the peak current data. If no value is provided by the manufacturer, a typical value to use can be 2 times the Full Load Rated Current of the motor. For exact value for this parameter, it may be necessary to contact the motor manufacturer.

If a system is “drive limited” (meaning that the motor can handle more current than the drive can deliver), the peak current actually used by the system may be lower than the value specified here.

Full Load Rated Current

This parameter specifies the rated continuous current for the motor. Motor data sheets refer to this as the Full Load Rated Current. The valid range is 0.000 to 99999.999 Amps (rms). The motor manufacturer provides this information.

Some manufacturers refer to this value as Full Load Amps or F/L Amps.

NOTE

Since many induction motors can be wired to operate using either 200 V or 400 V supply ranges, it is important to use the Full Load Rated Current at the desired Voltage Rating.

Rated Voltage

This parameter specifies the motor’s rated operating voltage. This value is usually printed on the motor nameplate as well as on a motor data sheet provided by the motor manufacturer. The range for this parameter is dependant upon what drive is being used. The following table shows the maximum value for each drive voltage rating.

Drive	AC Supply Voltage			
	200	400	575	690
Unidrive M	265	530	635	765
Digitax HD	265	530	N/A	N/A

NOTE

Since many induction motors can be wired to operate using either 200 V or 400 V supply ranges, it is important to use the proper nameplate ratings based on the selected supply voltage.

Motor Poles

Specifies the number of magnetic poles on the motor. The supported values are 2 to 480. The motor manufacturer typically provides the motor pole information.

If the manufacturer does not provide the motor poles information, there is a simple calculation that allows you to determine the # of poles based on several known parameters. The calculation is as follows:

# of Poles = (2 * Rated Frequency * 60) / Motor Synchronous Speed

The synchronous speed of the motor is the rated speed without slip taken into account.

Rotor Inertia

This parameter specifies the inertia of the motor rotor. The range is 0.00000 to 1000.00000 kg*m^2. The PTi210 module uses this parameter to interpret the “Inertia Ratio” parameter found on the Tuning view. “Inertia Ratio” is specified as a ratio of reflected load inertia to motor inertia.

Rated Frequency

This parameter specifies the Rated Frequency of the motor. The applied frequency will directly affect the speed of the motor. This value is usually printed on the motor nameplate as well as on a motor data sheet provided by the motor manufacturer. The range for this parameter is 0.0 to 550.0 Hz.

Phase Resistance

Specifies the phase resistance of the motor. You can determine this value by measuring the resistance between any motor phase and ground with an ohmmeter. The range is 0.000000 to 500.000000 Ohms. The motor manufacturer will typically provide this data.

Transient Inductance

Specifies the phase inductance of the motor. This is the inductance measured from phase to ground, NOT phase-to-phase. The range is 0.000 to 500.000 mH.

Max Operating Speed
This parameter specifies the maximum speed of the motor when used with a variable speed drive to achieve velocities over the rated base speed of the motor.

This parameter is not to be confused with the Full Load Rated Speed of the induction motor. The motor manufacturer typically provides this value. The range for the Max Operating Speed is 33000.00 rpm (depending on encoder resolution).

Full Load Rated Speed
This parameter specifies the rated speed of the motor that would be achieved if the motor were attached directly to the rated line voltage. The motor manufacturer typically provides this information. The range for the Full Load Rated Speed is 0.0 to 33000.00 rpm (depending on encoder resolution).

Some manufacturers refer to this parameter as Base Speed, Base RPM.

Thermal Time Constant
Specifies the Thermal Time Constant of the motor. This parameter is used by the drive for thermal protection of the motor. The drive models the temperature of the motor using a formula that generates an overload accumulator value. The formula is a function of the Thermal Time Constant. When the accumulator reaches 100 %, the drive can trip or foldback depending on other drive settings. For more information on the Thermal Time Constant, please refer to the relevant drive *User Guide* (Pr **04.015**, **04.016**, and **04.019**).

Rated Power Factor
This parameter specifies the motor rated full load power factor. Power Factor is the angle between the motor voltage and motor current vectors. This parameter can be provided by the user, or can be measured using the Auto-Tune feature of the drive (described later). The range for the Power Factor parameter is 0.000 to 1.000.

Stator Inductance
This parameter specifies the motor stator inductance and is defined as the inductance of the motor stator when rated flux is applied. This parameter is used to create velocity loop gains for the PTi210 module and drive. This value can be provided by the motor manufacturer, or can be measured using the Auto-Tune feature of the drive (See *Run Auto-Tune Button* on page 54). The range for the Stator Inductance is 0.00 to 5000.00 mH.

Motor Kt
This parameter specifies the torque constant of the motor, which is defined as the amount of torque produced per Amp. The units for this parameter are (N*m) / Amp. This parameter can be provided by the motor manufacturer, or can be measured by using the Auto-Tune feature (explained later). The range for this parameter is 0.00 to 500.00 Nm/A.

Saturation Breakpoint 1
The rated level of flux in most induction motors causes saturation. Therefore, the flux against flux producing current characteristic is non-linear. The effects of saturation are to cause a step increase in torque when operating in torque mode as the speed increases into the field- weakening region. The user can simulate the non-linear behavior by defining two breakpoints on the flux vs. magnetizing current curve. This information is often not provided by the motor manufacturer, but can be measured by using the Auto-Tune feature (See *Run Auto-Tune Button* on page 54). The range for the Saturation Breakpoint is 0 to 100 % of rated flux.

For more information on the Saturation Breakpoint parameter, please refer to the relevant drive *User Guide*.

Saturation Breakpoint 2
See Saturation Breakpoint 1.

Saturation Breakpoint 3
See Saturation Breakpoint 1.

Saturation Breakpoint 4
See Saturation Breakpoint 1.

8.2.5 Values from Drive Column
The Values from Drive column is a group of parameters that are constantly being read from the drive. The theory of operation is that the user will often perform an Auto-Tune function that reads/measures/calculates data. The results of those measurements are read from the drive and displayed in the Values from Drive column. Once they are displayed in PowerTools Studio (in the Values From Drive column) the user can apply those values to the Motor Parameters column by clicking on "***Apply to Config.***", in the middle of the Motor tab (this button looks like a series of arrows pointing from the Values from Drive column towards the Motor Parameters column).

The values in the Values from Drive column are not saved as part of the configuration file. To save these values, the user must use the ***Apply to Config*** button to save them.

This column is only functional when online with the PTi210 module. When offline, the values in the Values from Drive column will all read zero.

8.2.6 Apply to Config. Button
When the user runs the Auto-Tune feature available in the drive, PowerTools Studio reads the results of the Auto-Tune and displays them in the Values from Drive column of the Motor tab. After the Auto-Tune, the measured values are only saved in the Drive NVM, and not in the PTi210 module. Therefore, in order to store the values in the PTi210 module, the Auto-Tune values must be applied to the configuration file. When the user presses ***Apply to Config.***, the values in the "Values From Drive" column are transferred into the

Motor Parameters column. Then the values must be downloaded by downloading the entire configuration file using **Device** > **Download**.

8.2.7 Run Auto-Tune Button

The Unidrive M and Digitax HD drive have the ability to run an Auto-Tune operation thereby measuring several different motor parameters. Doing so allows the drive to obtain certain parameters that are not typically provided by the motor manufacturer, and also optimizes other drive parameters to work properly with the connected motor/load.

PowerTools Studio allows the user to initiate this Auto-Tune feature from the Motor tab on the Drive/Encoder view.

The Unidrive M/Digitax HD provide various Auto-Tune tests depending on the drive operating mode (RFC-A/RFC-S), from a basic stationary test to more advanced rotational and inertia tests, however, PowerTools Studio only supports Mode 2 in RFC-S mode and in RFC-A mode, Modes 1 and 2.

Table 2-1 Auto-Tune test (RFC-S Mode)

Auto-Tune Value (Pr 05.012)	Test Type	Comment
2	Rotating	Performs a basic rotating Auto-Tune test to measure the necessary motor control parameters.

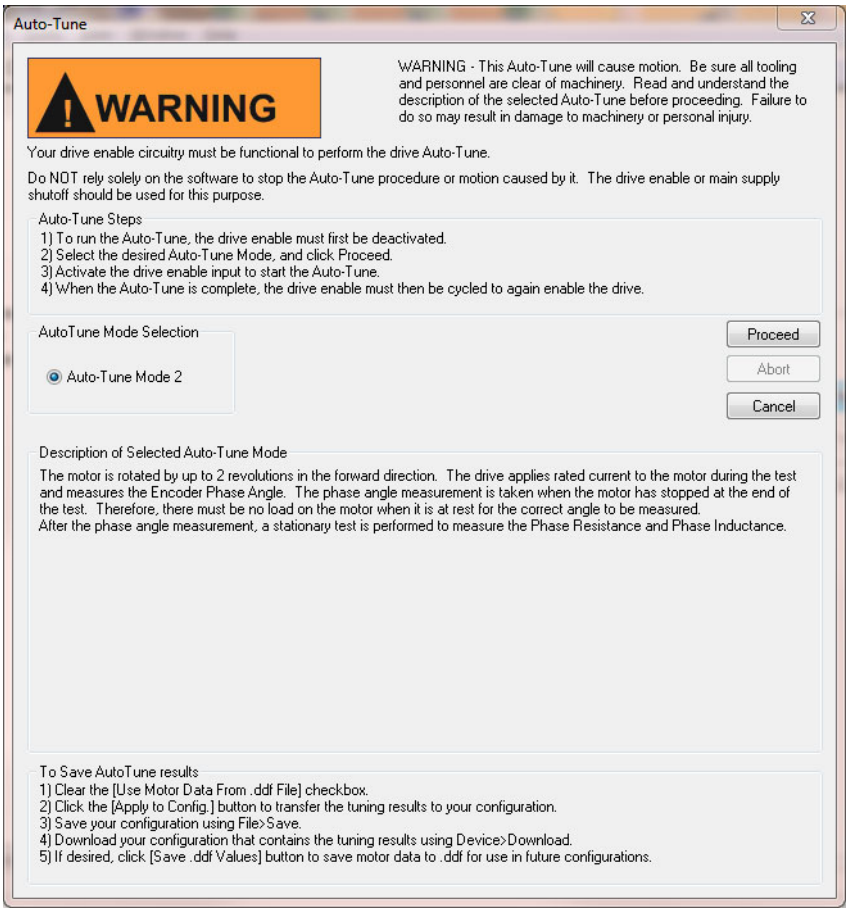


Figure 8-3: Auto-Tune Window - RFC-S Mode

Table 2-2 Auto-Tune tests (RFC-A Mode)

Auto-Tune Value (Pr 05.012)	Test Type	Comment
1	Basic	Stationary test to measure/set the following parameters for basic control: Pr 04.013 (*Current Controller Kp Gain)* Pr 04.014 (*Current Controller Ki Gain*) Pr 05.017 (*Stator Resistance*) Pr 05.024 (*Transient Inductance*) Pr 05.059 (*Maximum Deadtime Compensation*) Pr 05.060 (*Current At Maximum Deadtime Compensation*)
2	Improved	As Basic stationary test but rotates the motor shaft up to 67 % of rated frequency to provide more accurate values. The following parameters are written when the test has completed: Pr 05.010 (*Rated Power Factor*) Pr 05.025 (*Stator Inductance*) Pr 05.029 (*Saturation Breakpoint 1*) Pr 05.030 (*Saturation Breakpoint 3*) Pr 05.062 (*Saturation Breakpoint 2*) Pr 05.063 (*Saturation Breakpoint 4*) Pr 04.045 (*No-load Core Loss*) Pr 04.046 (*Rated Core Loss*) = 0

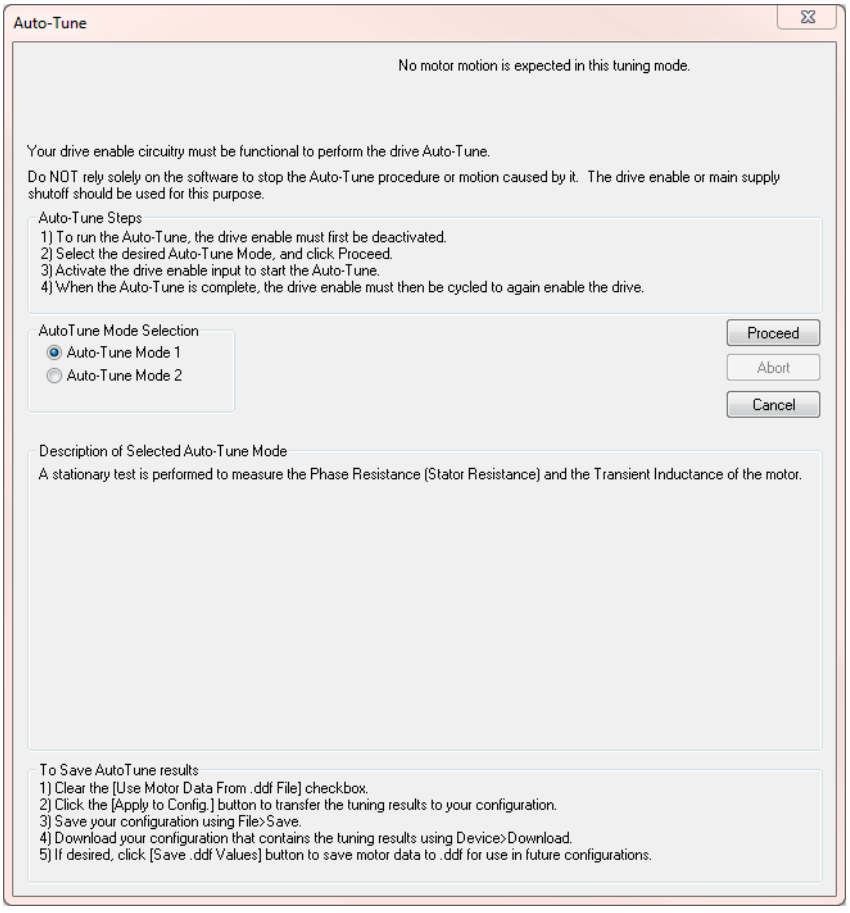


Figure 8-4: Auto-Tune Window for RFC-A Mode

To initiate an Auto-Tune from within PowerTools Studio, click on the ***Run Auto-Tune*** button on the right side of the Motor tab on the Drive/Encoder view. The Auto-Tune window opens and contains warnings and instructions related to the Auto-Tune procedure, as well as selection of the Auto-Tune mode.

Some Auto-Tunes cause motion while others do not. Some Auto-Tunes should be run with the motor unloaded, and others with the load attached. It is important to read and understand the warnings and instructions on the Auto-Tune windows. Examples of the Auto-Tune window is shown in Figure 8-3 and Figure 8-4. For more information on Auto-Tune refer to the relevant drive *User Guide*.

Entering Motor Data

When entering motor data parameters, some parameters are absolutely crucial to fundamental motor operation, while others are necessary only for optimum performance. The following chart defines the level of necessity for each motor data parameter.

RFC-S Mode

Motor Parameter		Required For Operation	Configured By
Name	**Number**		
Motor Name	N/A	No	User
Peak Current	N/A	Yes	User
Continuous Current	Pr 05.007	Yes	User
Motor Poles	Pr 05.011	Yes	User
Motor And Load Inertia	Pr 03.018	Yes	User/Auto-Tune
Motor Ke	Pr 05.033	Yes	User
Stator Resistance	Pr 05.017	Yes	User/Auto-Tune
Ld	Pr 05.024	Yes	User/Auto-Tune
Max. Operating Speed	N/A	Yes	User
Thermal Time Constant	Pr 04.015	No	User
Encoder Phase Angle	Pr 03.025	Yes	User/Auto-Tune
Rated Speed	Pr 05.008	Yes	User
Rated Voltage	Pr 05.009	Yes	User
Maximum Deadtime Compensation	Pr 05.059	Yes	User/Auto-Tune
Current At Maximum Deadtime Compensation	Pr 05.060	Yes	User/Auto-Tune
Current Controller Kp Gain	Pr 04.013	Yes	User/Auto-Tune
Current Controller Ki Gain	Pr 04.014	Yes	User/Auto-Tune
No-load Lq	Pr 05.072	Yes	User/Auto-Tune
Inertia Times 1000	Pr 04.033	Yes	User/Auto-Tune
Torque Per Amp	Pr 05.032	Yes	User

RFC-A Mode

Motor Parameter		Required For Operation	Configured By
Name	Number		
Motor Name	N/A	No	User
Peak Current	N/A	Yes	User
Full Load Rated Current	Pr 05.007	Yes	User
Motor Poles	Pr 05.011	Yes	User
Rated Voltage	Pr 05.009	Yes	User
Motor And Load Inertia	Pr 03.018	Yes	User/Auto-Tune
Rated Frequency	Pr 05.006	Yes	User
Stator Resistance	Pr 05.017	Yes	User/Auto-Tune
Transient Inductance	Pr 05.024	Yes	User/Auto-Tune
Stator Inductance	Pr 05.025	No	User/Auto-Tune
Max. Operating Speed	N/A	Yes	User
Full Load Rated Speed	Pr 05.008	Yes	User
Thermal Time Constant	Pr 04.015	No	User
Rated Power Factor [1]	Pr 05.010	Yes	User/Auto-Tune
Torque Per Amp Kt	Pr 05.032	Yes	Drive
Saturation Breakpoint 1	Pr 05.029	No	User/Auto-Tune
Saturation Breakpoint 2	Pr 05.062	No	User/Auto-Tune
Saturation Breakpoint 3	Pr 05.030	No	User/Auto-Tune
Saturation Breakpoint 4	Pr 05.063	No	User/Auto-Tune
Maximum Deadtime Compensation	Pr 05.059	Yes	User/Auto-Tune
Current At Maximum Deadtime Compensation	Pr 05.060	Yes	User/Auto-Tune
Inertia Times 1000	Pr 04.033	Yes	User/Auto-Tune
Current Controller Kp Gain	Pr 04.013	Yes	User/Auto-Tune
Current Controller Ki Gain	Pr 04.014	Yes	User/Auto-Tune
No-load Core Loss	Pr 04.045	No	User/Auto-Tune
Rated Core Loss	Pr 04.046	No	User

[1] – If the Stator Inductance (Pr **05.025**) is set to a value of zero, then the user MUST provide a value for the motor Rated Power Factor (Pr **05.010**) in order to run the motor. That value specified by the user is written to parameter Pr **05.010** and is used in the control algorithm. However, if the user enters a non-zero value for Stator Inductance, then the drive runs a background task that continuously calculates the motor Rated Power Factor Value. In this case, the user-entered value for Power Factor is ignored.

In some cases, the certain motor parameters are handled differently based on the values entered by the user.

8.2.8 Save .ddf Values Button

Once the user has entered the data for the motor they are using, they may or may not wish to save the motor data to the specified motor .ddf file so it can be easily recalled at a later time. If the user does not save the motor data to the specified motor .ddf file, then the motor data will only reside in the specific application configuration file that it has been entered into.

In order to save the motor data to the specified motor .ddf file, the user simply clicks on ***Save .ddf Values*** on the Motor Tab. Doing so will take all the parameter values and write them to the specified motor .ddf file automatically.

When saving to the .ddf file, if PowerTools Studio finds that a motor already exists with the same name, the User Defined Motor Name Conflict dialog box shown in Figure 8-5 will appear. The user must then decide how to proceed with saving the motor data .ddf file.

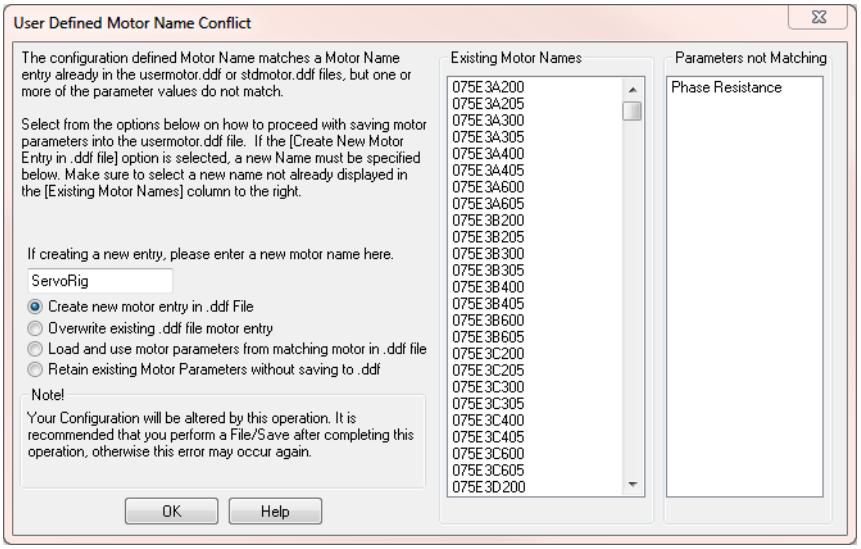


Figure 8-5: User Defined Motor Name Conflict

The **User Defined Motor Name Conflict** dialog box presents the user with four options on how to proceed with saving the motor data. Those four options are.

1. Create new motor entry in .ddf File

The user can select to keep the existing data and create a new entry into the specified user defined motor .ddf file with a different name. After selecting this option, the user simply enters a new name in the **Name** text box, shown in Figure 8-5. Then click ***OK***, the data will be written to the .ddf file using the new motor name.

2. Overwrite existing .ddf file motor entry

The user can select to overwrite the existing data in the .ddf file with the current data in the Motor Parameters column. If this option is selected, the data in the .ddf file will be overwritten and lost forever. The overwritten data cannot be recovered.

If the user attempts to overwrite data for a Standard Motor (in the specified standard motor .ddf file), the operation will be canceled and the user will be notified that they cannot proceed. Figure Figure 8-6 shows the error message that will be produced when the user attempts to overwrite a standard motor. In this case, the user would need to change the motor name before saving to the .ddf file.

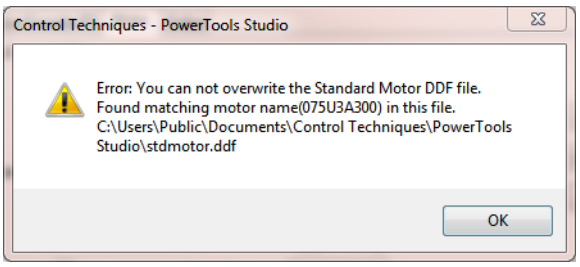


Figure 8-6: Error Message - Overwriting a Standard Motor .ddf file

3. Load and use motor parameters from matching motor in .ddf file

If this option is selected, the motor data in the specified motor .ddf or specified standard motor .ddf file for the matching Motor Name will overwrite the data in the Motor Parameters column. After this option is selected, the Use Motor Data From .ddf File check box will be selected, and all the parameter values will be unavailable.

4. Retain existing Motor Parameters without saving to .ddf

If the user selects this option, the values in the Motor Parameters column will not be written to the specified user defined motor .ddf file, and the values will only reside within the configuration file. The specific motor data values will not be available for selection in the Motor Type list box because they are not saved to the .ddf file. The “Save .ddf Values” operation is in effect canceled.

Existing Motor Names List

This list box is part of the User Defined Motor Name Conflict dialog box and contains all the names of the motors that exist in the specified user defined motor .ddf and specified standard motor .ddf files. When selecting a new name, it is important to select a name that is not already displayed in the list box.

NOTE

Please note that when attempting to overwrite existing motor details, the motor name is case-sensitive.

Parameters Not Matching List

This list box is part of the User Defined Motor Name Conflict dialog box and displays the parameter value(s) from the Motor Parameters column that do not match the equivalent parameter value in either the specified standard motor .ddf or specified user defined motor .ddf files, for the motor with the matching name.

This helps the user to determine whether they wish to overwrite, cancel, or create a new motor with this Save .ddf Values operation.

8.2.9 Help Button

The *Help* button, located on the right side of the Motor tab, will display help information for the different functions available on the tab.

8.3 General Encoder Settings

This section details the P1 and P2 general encoder settings for both the drive and SI-Universal Encoder option module.

For this document, only the Drive Encoder P1 settings are shown, the Drive Encoder P2 settings are similar but the parameter references will differ, please refer to the relevant drive User Guides for more information.

Drive Encoder P1

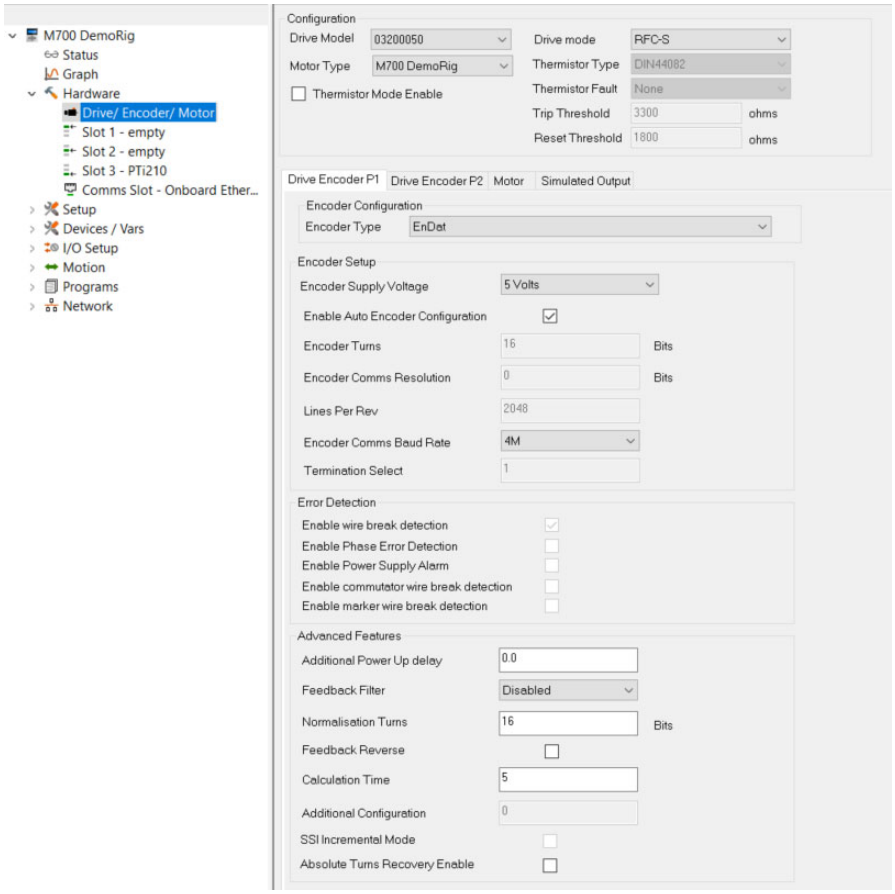


Figure 8-7: Drive Encoder P1

Encoder Configuration

Encoder Type

This drop-down box selects the feedback encoder type connected to the drive P1 interface.

This setting is written to drive parameter Pr **03.038** (*P1 Device Type*).

Encoder Supply Voltage

Because of the wide variety of encoders supported by the Unidrive M/Digitax HD, the user must have the ability to define the voltage supplied to the encoder hardware. The available voltage levels are 5 V, 8 V, and 15 V and are selectable from the list box.

NOTE

Be sure not to configure a supply voltage greater than that supported by the encoder. The Unidrive M/Digitax HD cannot protect against possible damage to the encoder if the supply voltage is set too high.

This setting is written to drive parameter Pr **03.036** (*P1 Supply Voltage*).

Enable Auto Encoder Configuration

When a suitable comms protocol encoder (e.g. SC.Hiperface, EnDat, BiSS, or SSI) is being used, the Unidrive M/Digitax HD can interrogate the encoder on power-up, and acquire many of the encoder setup parameters automatically. To enable this feature, the Enable Auto Encoder Configuration check box must be selected. When the check box is selected (active) the Encoder

Turns, Encoder Comms Resolution, and Equivalent Lines Per Revolution parameters will appear dimmed, implying that the user no longer needs to configure those parameters. If the check box is clear, then the user must specify the correct values for these three parameters (Encoder Turns, Encoder Comms Resolution, and Equivalent Lines Per Revolution).

The aforementioned encoder parameter values are stored in non-volatile memory embedded in the encoder. The values stored in NVM cannot be changed by the user.

This setting is written to drive parameter Pr **03.041** (*P1 Auto-config Select*).

Encoder Turns

This parameter determines the maximum number of revolutions before the Rev Counter register (Pr **03.028**) rolls over. If using an absolute encoder, this should be set at the maximum number of turns of the absolute encoder or lower.

The maximum number of turns is defined as follows:

Maximum Turns = 2N, where N is the Encoder Turns parameter

If the Enable Auto Encoder Configuration check box is selected, then the user does not need to enter this parameter, and the Encoder Turns box will appear dim.

This setting is written to drive parameter Pr **03.033** (*P1 Rotary Turns Bits*).

Encoder Comms Resolution

Where encoder communications is used to initially read the absolute encoder position (e.g SC.Hiperface or SC.EnDat), the comms resolution must be set to the maximum resolution of the absolute position data.

If the Enable Auto Encoder Configuration check box is selected, then the user does not need to enter this parameter, and it will appear dim.

This setting is written to drive parameter Pr **03.035** (*P1 Comms Bits*).

Lines Per Rev

When sin/cos signals are used, the number of encoder lines per revolution must be set up correctly to give the correct speed and position feedback. The Lines Per Rev (LPR) is defined as follows:

Position Feedback Device	LPR
Ab	Number of lines per revolution
Fd, Fr[1]	Number of lines per revolution / 2
SC.Hiperface, SC.EnDat, SC.SSI, SC BiSS	Number of sine waves per revolution

[1] - Not currently supported in PowerTools Studio.

If the Enable Auto Encoder Configuration check box is selected, then the user does not need to enter this parameter, and it will appear dim.

This setting is written to drive parameter Pr **03.034** (*P1 Rotary Lines Per Revolution*).

Encoder Comms Baud Rate

This parameter defines the baud rate for the encoder communications. The list box allows the user to select from various baud rates between 100 k baud and 4 M baud.

This setting is written to drive parameter Pr **03.037** (*P1 Comms Baud Rate*).

Termination Select

This parameter is used to enable or disable the terminations on the position feedback interface inputs.

This setting is written to drive parameter Pr **03.039** (*P1 Termination Select)*.

Error Detection

These options enable or disable certain encoder error detection features in the drive relevant to the selected feedback encoder, for more information please refer to the relevant drive User Guide.

These settings are written to drive parameter Pr **03.040** (*P1 Error Detection Level*).

Additional Features

Additional Power Up delay

When the position feedback is initialised, at power-up or at any other time, a delay is included before the information from the feedback device is used or any attempt is made to communicate with the device, this amount of delay depends on the encoder type, this parameter specifies an additional delay (up to 25.0 seconds) to be applied before communication with the encoder commences.

This setting is written to drive parameter Pr **03.049** (*P1 Additional Power-up Delay*).

Feedback Filter

This parameter defines the time period for a sliding window filter that may be applied to the feedback signal from the drive P1 position feedback interface.

This setting is written to drive parameter Pr **03.042** (*P1 Feedback Filter*).

Normalisation Turns

This parameter specifies the number of turns bits, (maximum of 16) included in the 32-bit normalisation parameters (e.g. Pr **03.058**, Pr **03.059** etc.).

This setting is written to drive parameter Pr **03.057** (*P1 Normalisation Turns*).

Feedback Reverse

This parameter when selected (checked) reverses the direction of the position feedback signal.

This setting is written to drive parameter Pr **03.056** (*P1 Feedback Reverse*).

Calculation Time

This parameter is only valid for EnDat and BiSS encoders and is the time from the first edge of the clock signal from the position feedback interface until the encoder has calculated the position and is ready to return this information.

This value is calculated automatically during encoder initialisation if Enable Auto Encoder Configuration is enabled and is written to drive parameter Pr **03.060** (*P1 Calculation Time*).

Additional Configuration

This parameter is only valid for BiSS encoders and specifies additional configuration for encoders which is not specified in the other encoder setup parameters.

This value is retrieved from the encoder during encoder initialisation (if supported) if Enable Auto Encoder Configuration is enabled and is written to drive parameter Pr **03.074** (*P1 Additional Configuration*).

SSI Incremental Mode

This parameter is only valid for SSI encoders and enables the drive SSI Incremental Mode (Pr **03.047**).

Absolute Turns Recovery Enable

This parameter allows turns information beyond the number of turns bits provided by the position feedback device for the relevant drive encoder interface. This parameter writes to Pr **03.073** for P1 and Pr **03.173** for P2.

For single turn encoders, (i.e. Encoder Turns = 0), the PTi210 would normally maintain the number of revolutions internally, however this information is lost on power down. With later drive firmware versions, the drive can be configured to monitor the number of revolutions for both the P1 and P2 encoder interfaces, these are automatically saved and restored on a power-cycle.

When Absolute Turns Recovery is enabled, the PTi210 will read the number of revolutions from the drive parameter for the relevant encoder port (Pr **03.028** for P1 and Pr **03.128** for P2) on startup.

NOTE

For SSI encoders, the SSI Incremental Mode must also be enabled to use this feature.

8.4 Simulated Encoder Output

Simulated Output Tab

This tab allows the user to configure the simulated encoder output of the drive.

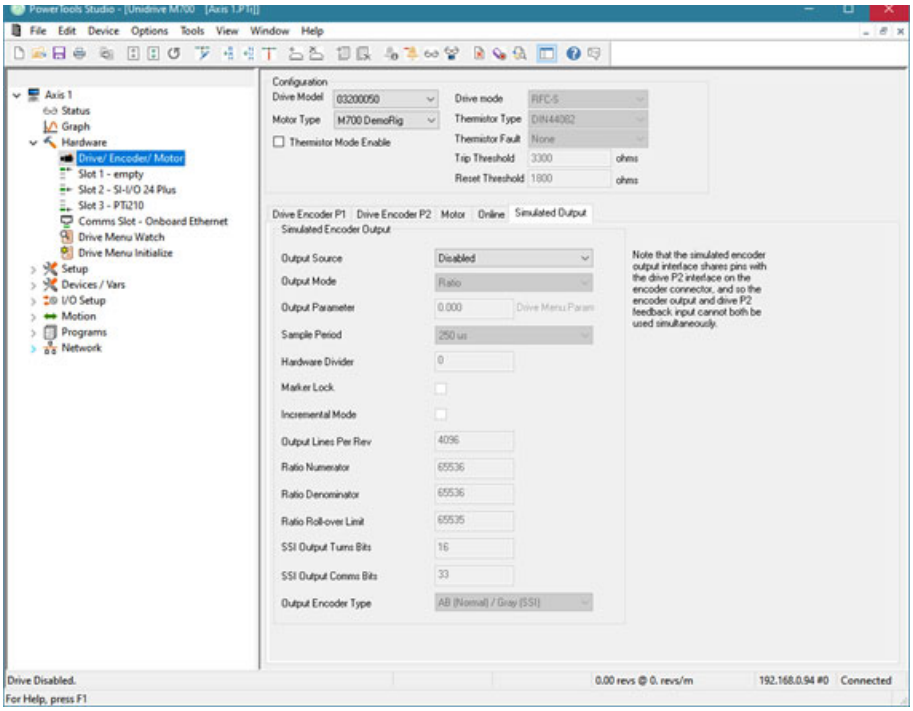


Figure 8-8: Drive/Encoder/Motor View - Simulated Output Tab

NOTE

The simulated encoder output interface and the drive P2 feedback interface share pins on the encoder connector, this means that both interfaces cannot be used at the same time.

Output Source

Selects the source of data for the encoder simulation output to follow.

The available settings are:

- **Disable**d - This disables the encoder simulation output functionality of the drive, which allows the pins on the encoder connector to be used for the P2 encoder input instead.
- **Hardware Passthrough** - This sets the encoder simulation Output Mode to Hardware mode.
- **Virtual Master** - This sets the encoder simulation source to the parameter for the virtual master of the PTi210, so that the virtual master is output.
- **Drive Parameter** - This allows the encoder simulation source to be set to any drive parameter that has the correct 16-bit format.
- **Slot [X] P[Y]** - These options set the encoder simulation source to be the parameter for the 1st or 2nd encoder feedback interface [Y] of the specified slot [X] option module, such as the SI-Universal Encoder or SI-I/O 24 Plus.

Output Mode

Selects the method the drive will use to convert the input signal into the simulated encoder output.

This setting is written to drive parameter Pr **03.088** (*Encoder Simulation Mode*).

The available settings are:

- **Hardware** - In this mode, the encoder simulation output is derived directly in hardware from the P1 position feedback interface in the drive (Pr **03.029**), and the output is derived from the input with negligible delay.

The following parameters are only available in this mode:

- **Hardware Divider**
  This parameter can be used to scale the encoder simulation output relative to the P1 input, so a divider of 2 would mean that for every two counts on the P1 input, the encoder simulation would output one count**.**
  This setting is written to drive parameter Pr **03.089** (*Encoder Simulation Hardware Divider*).
- **Marker Lock**
  When the marker lock is off, the marker output from the encoder simulation will match the marker of the encoder input source exactly.

  When the marker lock is on, the incremental output signals are adjusted on each marker event so that the A and B are high with an AB type output, or F is high with an FD or FR type output. Marker locking is not recommended if the number of lines per revolution of the encoder simulation source combined with the ratio does not give an encoder simulation output with a multiple of 4 counts per revolution (i.e. between each output marker event) for

AB signals, or a multiple of 2 counts for FD or FR signals, because this causes a count error in the system receiving these signals. The input marker pulse width is not adjusted to take account of the divider ratio, but is simply routed from the input to the output. Therefore the output marker pulse becomes shorter with respect to the output incremental signals as the divider ratio is increased.

This setting is written to drive parameter Pr **03.090** (*Encoder Simulation Hardware Marker Lock*).

- **Lines Per Rev** - When in this mode, the Output Lines Per Rev parameter is used to scale the encoder simulation output relative to its input.

  The input is multiplied by the value of this parameter, then divided by 65536, to get the output. 65536 is used as it is the maximum value of the 16-bit source parameter.

  The following parameters are available in this mode:
  - **Output Lines Per Rev**
    This setting is written to drive parameter Pr **03.092** (*Encoder Simulation Output Lines Per Revolution*).
  - **Incremental Mode**
    When incremental mode is disabled, the encoder simulation output position will be kept synchronised with the input position. If there are large step changes in the input position for any reason, the output will simulate additional fast pulses in the right direction up to 500 kHz until the output position matches the input (after scaling). The output position will always match the scaled input position, although it may take a short delay to get there due to this frequency limit. At initialisation, the encoder simulation output will not use this method of simulating pulses, the output will have a step change to match the input.

    When incremental mode is enabled, the encoder simulation output will only follow the changes in position as they happen, and it will not attempt to "catch up" with large step changes in input position by simulating fast counts to interpolate the position change. If a large step position change occurs on the input, the simulated output will have the exact same change.

    Incremental mode only has an effect when the encoder simulation source is set to the drive's P1 feedback interface (Pr **03.029**), otherwise it always acts as if incremental mode is disabled.

    This setting is written to drive parameter Pr **03.091** (Encoder Simulation Incremental Mode Select).

- **Ratio** - In ratio mode, the source position is multiplied by the ratio between the numerator and the denominator before being output.

  For example, to scale an input encoder with 65536 counter per revolution to an encoder simulation output with 1024 counts per revolution, the numerator could be set to 1024, and the denominator could be set to 65536. This way, the simulated output encoder position would increment one time for every 64 times the input encoder's position was incremented.

  The following parameters are available in this mode:
  - **Ratio Numerator**
    This setting is written to drive parameter Pr **03.093** (*Encoder Simulation Numerator*).
  - **Ratio Denominator**
    This setting is written to drive parameter Pr **03.094** (*Encoder Simulation Denominator*).
  - **Ratio Roll-over Limit**
    The roll-over limit specifies the number of simulated output encoder counts before a marker pulse is output.

    This setting is written to drive parameter Pr **03.095** (*Encoder Simulation Output Roll-over Limit*).
- **SSI** - In this mode the B output becomes the clock input and the A output is the data output.
  The following parameters are available in this mode:
  - **SSI Output Turns Bits**
    This parameter specifies how many of the most significant bits of the outputted SSI data will be used to represent the encoder turns.

    This setting is written to drive parameter Pr **03.096** (*Encoder Simulation SSI Turns Bits*).
  - **SSI Output Comms Bits**
    This parameter specifies the total number of bits to be transmitted by the encoder simulation output.

    This setting is written to drive parameter Pr **03.097** (*Encoder Simulation SSI Comms Bits*).

Output Parameter

When the encoder simulation output mode is set to **Drive Parameter**, this parameter specifies which drive parameter the encoder simulation output follows. For example, setting it to **03.029** (*P1 Position*) would cause the encoder simulation output to mimic the P1 encoder input.

In other encoder simulation modes, this parameter has a fixed value, e.g. when **Virtual Master** is selected, this parameter is fixed to the parameter of the virtual master position in the PTi210's setup menu in the drive.

This setting is written to drive parameter Pr **03.085** (*Encoder Simulation Source*).

Sample Period

When the encoder simulation output source is enabled and not set to **Hardware Passthrough**, this parameter can be used to adjust the rate at which the encoder simulation output is updated to follow the encoder simulation source.

The available options are:

- 250 µs
- 1 ms
- 4 ms
- 16 ms

Output Encoder Type

When the encoder simulation mode is set to either **Hardware, Lines Per Rev**, or **Ratio**, this drop-down selects what incremental encoder type will be simulated on the output.

The available options are:

- AB quadrature encoder
- FD (Frequency/Direction) encoder
- FR (Forward/Reverse) encoder

When the encoder simulation mode is set to **SSI**, this drop-down select whether the simulated output will be a Gray code type SSI encoder, or a Binary code type SSI encoder.

8.5 Slot # View

The Slot # View(s) allow the user to configure which option modules are populated in which slots. Some of the views, depending on the option module selected, have configuration or setup parameters associated with the module. Each of the views for the different option modules are shown below. The Unidrive M has three slots available and the Digitax HD has two slots.

8.5.1 Empty Slot View

If Empty Slot is selected in the Slot # Module list box, the remainder of the view should be blank. The hierarchy tree automatically updates to show that no module is populated in the specific slot, see Figure 8-9.

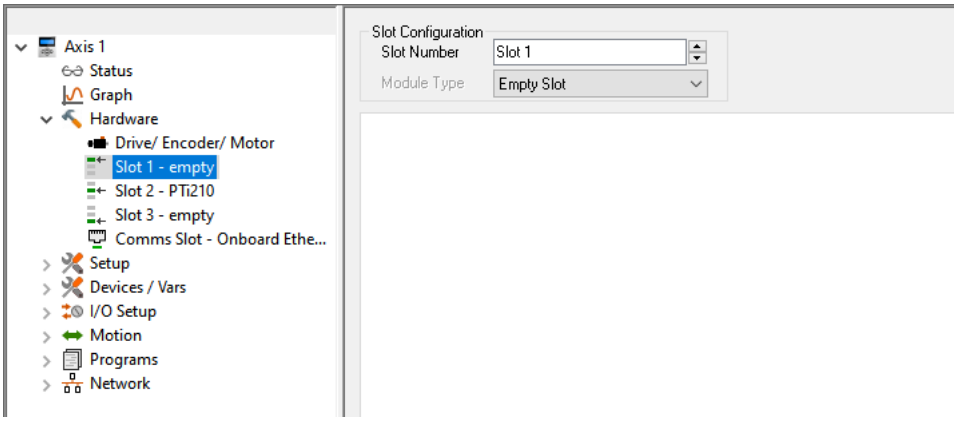


Figure 8-9: Slot # View (Empty Slot)

8.5.2 PTi210 Module View

If PTi210 is selected in the Slot # Module list box, the remainder of the view should be blank. The hierarchy tree automatically updates to show that a PTi210 module is populated in that specific slot, see Figure 8-10.

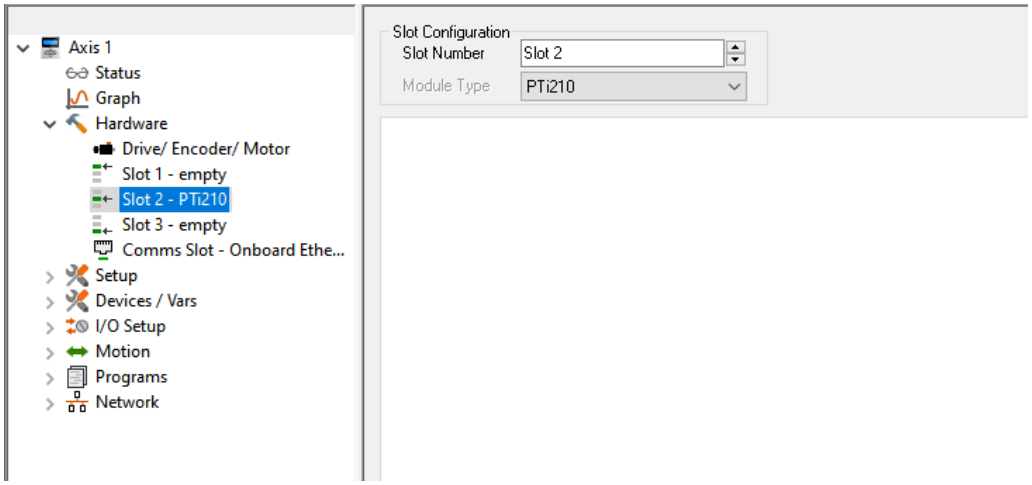


Figure 8-10: Slot # View (PTi210 Module)

8.5.3 SI-I/O Module View

If SI-I/O is selected in the Slot # Module list box. The hierarchy tree automatically updates to show that an SI-I/O module is populated in that specific slot, see Figure 8-11.

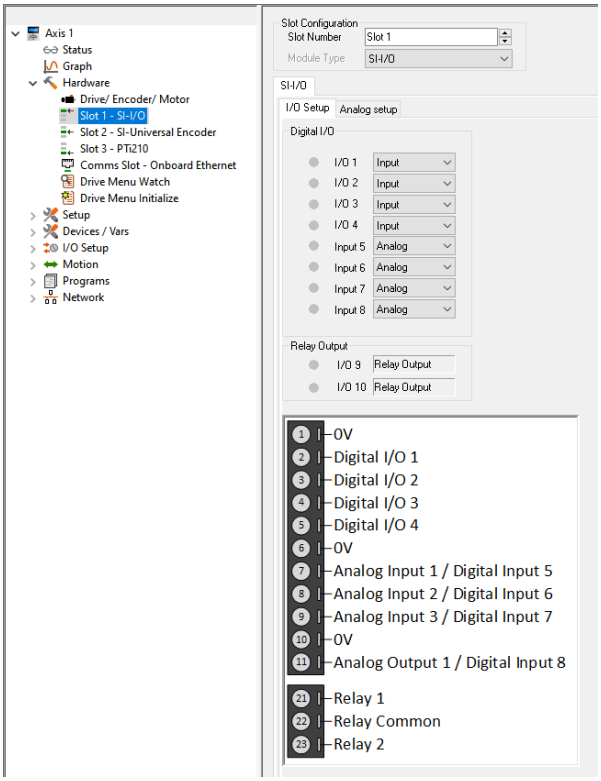


Figure 8-11: Slot # View (SI-I/O Module)

SI-I/O tab

The SI-I/O tab contains the parameters to configure the individual modes for the I/O terminals, the first four I/O terminals (I/O 1 to I/O 4) can be configured for either digital inputs or digital outputs. Input 5, Input 6 and Input 7 can be either digital input or analog input and Input 8 being either digital input or analog output.

To the left of each terminal is an indication of the state of that terminal, for a digital input/output this indication will be green when the terminal is active.

Analog setup tab

The Analog setup tab contains the parameters to configure the analog inputs and output, if no analog input or output is configured then the parameters in this tab will be disabled.

Figure 8-12 shows an example of when inputs 5, 6 and 7 are configured as analog inputs and input 8 as analog output.

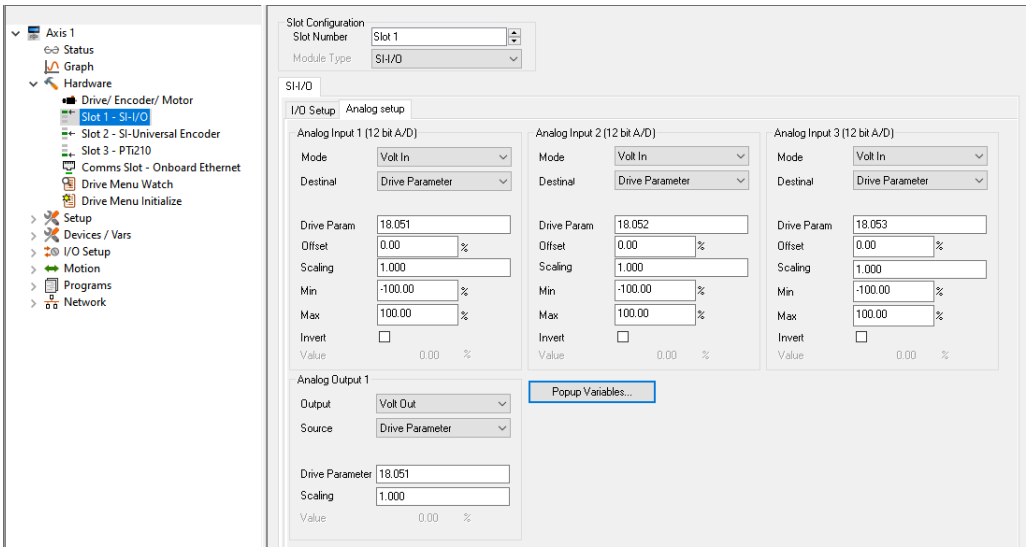


Figure 8-12: Inputs 5, 6 and 7 are configured as analog inputs and input 8 as analog output

Analog Inputs

Analog inputs 1 and 2 can be either a single ended voltage input (Volt In) or combined together to provide a differential voltage input (Differential), in this mode the Analog input 2 configuration is disabled.

Analog input 3 can be either a voltage (Volt In) or a current input, with various current input modes available.

For analog inputs, the Destinal setting specifies the parameter (Drive or Module) the analog input value will be written to, Offset and Scaling values may also be applied to this value. The value may also be inverted, this will change the polarity of the value from positive to negative or vice versa.

In addition to the Offset and Scaling values, the accepted analog input value range may be adjusted by specifying the Min and Max % values.

When online, each analog input value after all scaling and offsets have been applied is shown as a percentage.

Analog Output

The analog output can be either a voltage (Volt Out) or current output, the source of which can be either a Drive Parameter or Module Variable.

The specified analog output source (Drive Parameter or Module Variable) is used as the source of analog signal to provide the analog output value. This value may be scaled using the Scaling setting.

When online, the analog output value after scaling is shown as a percentage.

For more information on the SI-I/O analog input/output settings and operation please refer to the SI-I/O User Guide.

8.5.4 SI-Universal Encoder Module View

If SI-Universal Encoder is selected in the Slot # Module list box, the remainder of the view should have configuration parameters to define what type of encoder is being used and to define the encoder properties for both the P1 and P2 encoder interfaces. The hierarchy tree automatically updates to show that a SI-Universal Encoder module is populated in that specific slot, see Figure 8-13.

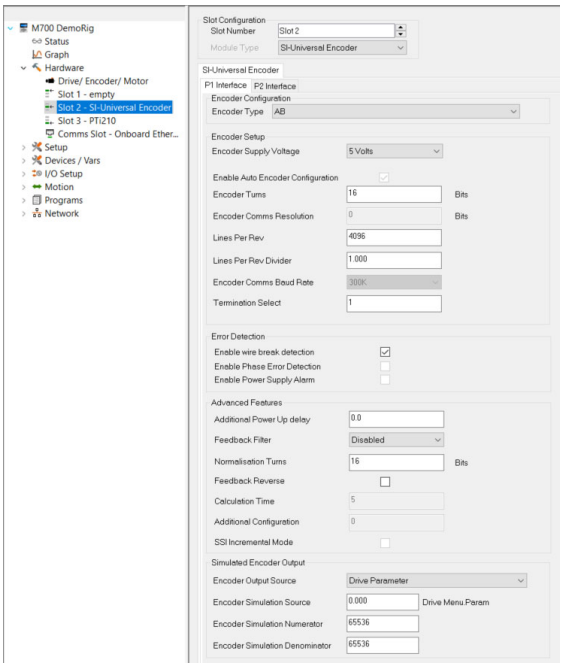


Figure 8-13: Slot # View (SI-Universal Encoder Module)

Encoder Interface Selection (P1 or P2)

The SI-Universal Encoder module provides two encoder interfaces designated P1 and P2, each interface is configured from the appropriate tab which contains the configurable parameters relevant to that particular interface. The SI-Universal Encoder setup menus (**x**) for each interface is shown in the following table.

SI-Universal Encoder Module Setup Menu

Slot #	Setup Menu (x)	
	P1	P2
1	15	25
2	16	26
3[1]	17	27

[1] – Not available on Digitax HD.

Not all encoders are supported on both P1 and P2 interfaces at the same time.

Encoder Configuration - Encoder Type

This parameter is available only when using a SI-Universal Encoder module. The Encoder Type list box provides direct access to the relevant interface encoder Device Type setting (parameter **x.038**) from the SI-Universal Encoder module configuration menu. This parameter allows one SI-Universal Encoder module to support many different encoder types. Select the desired type of encoder from this list box. See the *SI-Universal Encoder User Guide* for more information.

Sin Cos Hiperface (Stegmann comms)

Encoder Setup Parameters

Encoder Supply Voltage

Because of the wide variety of encoders supported by the Unidrive M/Digitax HD with the SI-Universal Encoder module, the user must have the ability to define the voltage supplied to the encoder hardware. The available voltage levels are 5 V, 8 V, and 15 V and are selectable from the list box on the Encoder view.

NOTE

Be sure not to configure a supply voltage greater than that supported by the encoder. The Unidrive M/Digitax HD and SI-Universal Encoder cannot protect against possible damage to the encoder if the supply voltage is set too high.

Enable Auto Encoder Configuration Check box

When a SC.Hiper, SC.EnDat, or EnDat encoder is being used, the Unidrive M/Digitax HD and/or SI-Universal Encoder module can interrogate the encoder on power-up, and acquire many of the encoder setup parameters automatically. To enable this feature, select the Enable Auto Encoder Configuration check box. When the check box is selected (enabled) the Encoder Turns, Encoder Comms Resolution, and Equivalent Lines Per Revolution parameters will appear dimmed implying that the user no longer needs to configure those parameters. If the check box is cleared, then the user must specify the correct values for these three parameters (Encoder Turns, Encoder Comms Resolution, and Equivalent Lines Per Revolution).

The aforementioned encoder parameter values are stored in non-volatile memory (NVM) embedded in the encoder. The values stored in NVM cannot be changed by the user.

Encoder Turns

This parameter determines the maximum number of revolutions before the Rev Counter register (Pr **03.028** or **x.028**) rolls over. If using an absolute encoder, this parameter should be set to the maximum number of turns or lower. The maximum number of turns is defined as follows:

Maximum # of Turns = 2^N, where N is the Encoder Turns parameter

The Encoder Turns box will appear dimmed when the Enable Auto Encoder Configuration check box is selected because the information will be read from the encoder.

Encoder Comms Resolution

Encoder communications is used to initially read the absolute encoder position (SC.Hiper or SC.EnDat), the comms resolution must be set to the maximum resolution of the absolute position data.

If the Enable Auto Encoder Configuration check box is selected, then the user does not need to enter this parameter, and the Encoder Comms Resolution box will appear dimmed.

Equivalent Lines Per Revolution

When sin/cos signals are used, the equivalent number of encoder lines per revolution must be set up correctly to give the correct speed and position feedback. The Equivalent Lines Per Revolution (ELPR) is defined as follows:

Position Feedback Device	ELPR
Ab	Number of lines per revolution
Fd, Fr	Number of lines per revolution / 2
SC.Hiper, SC.EnDat, SC.SSI	Number of sine waves per revolution

For SC.Hiper, SC.EnDat and SC.SSI encoders, the sine wave signal frequency can be up to 500 kHz, but the resolution is reduced at higher frequencies. The table below shows the number of bits of interpolated information at different frequencies and with different voltage levels at the drive encoder port. The total resolution in bits per revolution is the ELPR plus the number of bits of interpolated information.

Volt / Freq	1,000	5,000	50,000	100,000	150,000
1.2	11	11	11	10	10
1.0	11	11	10	10	9
0.8	10	10	10	10	9
0.6	10	10	10	9	9
0.4	9	9	9	9	8

If the Enable Auto Encoder Configuration check box is selected, then the user does not need to enter this parameter, and the Equivalent Lines Per Rev box will appear dimmed.

Lines Per Rev Divider

The Equivalent Lines Per Revolution (ELPR) parameter is divided by this value. This can be used when an encoder is used with a linear motor where the number of counts or sine waves per pole is not an integer.

Example:

The true number of encoder lines per rev (or electrical cycle) is 128.123. Since the Equivalent Lines Per Rev (ELPR) parameter must be a whole number, ELPR should be set to 128123, and then the Lines Per Rev Divider would be set to 1000 resulting in the following equation:

Actual Encoder Lines Per Rev = ELPR/Lines Per Rev Divider

= 128123 / 1000 = 128.123

Encoder Comms Baud Rate

This parameter defines the baud rate for the encoder communications. When using Hiperface encoders, the baud rate is fixed at 9600 and cannot be changed.

Simulated Encoder Output Parameters

Encoder Simulation Source

The SI-Universal Encoder module has a feature that allows the user to send out a simulated encoder output signal for use by an external device. The Encoder Simulation Source is used to define the source of the encoder signal. Any drive parameter in the form of a 0-65535 rollover counter can be used as the source parameter. Use this field to enter the desired drive source parameter (between 00.000 and 59.999).

By default, the PTi210 module configures the simulated output to work in Quadrature mode. In order to change the mode, the user will have to change the Drive Menu Initialization file.

Encoder Simulation Numerator

To add some flexibility to the Simulated Encoder Output signal, the SI-Universal Encoder module allows the user to scale the output signal by multiplying the source with a scaling factor. The scaling factor is made up of a Numerator and Denominator allowing the user to achieve nearly any ratio. By default, the Numerator and Denominator are both set to 65536 implying that the actual output value is equal to the Source value. Following is an equation that defines the use of the Numerator and Denominator parameters.

Simulated Encoder Output Signal = Simulated Encoder Source * (Numerator/Denominator)

Encoder Simulation Denominator

To add some flexibility to the Simulated Encoder Output signal, the SI-Universal Encoder module allows the user to scale the output signal by multiplying the source with a scaling factor. The scaling factor is made up of a Numerator and Denominator allowing the user to achieve nearly any ratio. By default, the Numerator and Denominator are both set to 65536 implying that the actual output value is equal to the Source value. Following is an equation that defines the use of the Numerator and Denominator parameters.

Simulated Encoder Output Signal = Simulated Encoder Source * (Numerator/Denominator)

SC.EnDat (Heidenhain®)

Encoder Setup Parameters

Encoder Supply Voltage

Because of the wide variety of encoders supported by the Unidrive M/Digitax HD and the SI-Universal Encoder module, the user must have the ability to define the voltage supplied to the encoder hardware. The available voltage levels are 5 V, 8 V, and 15 V and are selectable from the list box on the Encoder view.

NOTE

Be sure not to configure a supply voltage greater than that supported by the encoder. The Unidrive M/Digitax HD and SI-Universal Encoder cannot protect against possible damage to the encoder if the supply voltage is set too high.

Enable Auto Encoder Configuration

When a SC.Hiper, SC.EnDat, or EnDat encoder is being used, the Unidrive M/Digitax HD and/or SI-Universal Encoder module can interrogate the encoder on power-up, and acquire many of the encoder setup parameters automatically. To enable this feature, the Enable Auto Encoder Configuration check box must be selected. When the check box is selected (active) the Encoder Turns, Encoder Comms Resolution, and Equivalent Lines Per Revolution parameters will appear dimmed, implying that the user no longer needs to configure those parameters. If the check box is clear, then the user must specify the correct values for these three parameters (Encoder Turns, Encoder Comms Resolution, and Equivalent Lines Per Revolution).

The aforementioned encoder parameter values are stored in non-volatile memory embedded in the encoder. The values stored in NVM cannot be changed by the user.

Encoder Turns

This parameter determines the maximum number of revolutions before the Rev Counter register (Pr **03.028** or **x**.**028**) rolls over. If using an absolute encoder, this should be set at the maximum number of turns of the absolute encoder or lower. The maximum number of turns is defined as follows:

Maximum Turns = 2^N, where N is the Encoder Turns parameter

If the Enable Auto Encoder Configuration check box is selected, then the user does not need to enter this parameter, and the Encoder Turns box will appear dim.

Encoder Comms Resolution

Where encoder communications is used to initially read the absolute encoder position (SC.Hiper or SC.EnDat), the comms resolution must be set to the maximum resolution of the absolute position data.

If the Enable Auto Encoder Configuration check box is selected, then the user does not need to enter this parameter, and the Encoder Comms Resolution box will appear dim.

Equivalent Lines Per Revolution

When sin/cos signals are used, the equivalent number of encoder lines per revolution must be set up correctly to give the correct speed and position feedback. The Equivalent Lines Per Revolution (ELPR) is defined as follows:

Position Feedback Device	ELPR
Ab	Number of lines per revolution
Fd, Fr[1]	Number of lines per revolution / 2
SC.Hiper, SC.EnDat, SC.SSI	Number of sine waves per revolution

[1] - Not currently supported in PowerTools Studio.

For SC.Hiper, SC.EnDat and SC.SSI encoders, the sine wave signal frequency can be up to 500 kHz, but the resolution is reduced at higher frequencies. The table below shows the number of bits of interpolated information at different frequencies and with different voltage levels at the drive encoder port. The total resolution in bits per revolution is the ELPR plus the number of bits of interpolated information.

Volt / Freq	1,000	5,000	50,000	100,000	150,000
1.2	11	11	11	10	10
1.0	11	11	10	10	9
0.8	10	10	10	10	9
0.6	10	10	10	9	9
0.4	9	9	9	9	8

If the Enable Auto Encoder Configuration check box is selected, then the user does not need to enter this parameter, and the Equivalent Lines Per Rev box will appear dim.

Lines Per Rev Divider

The Equivalent Lines Per Revolution parameter is divided by this value. This can be used when an encoder is used with a linear motor where the number of counts or sine waves per pole is not an integer.

Example:

The true number of encoder lines per rev (or electrical cycle) is 128.123. Since the Equivalent Lines Per Rev parameter must be a whole number, ELPR should be set to 128123, and then the Lines Per Rev Divider would be set to 1000 resulting in the following equation:

$$\text{Actual Encoder Lines Per Rev} = \text{ELPR/Lines Per Rev Divider} = 128123 / 1000 = 128.123$$

Encoder Comms Baud Rate

This parameter defines the baud rate for the encoder communications. The list box allows the user to select from various baud rates between 100 k baud and 4 M baud.

Simulated Encoder Output Parameters

Encoder Simulation Source

The SI-Universal Encoder module has a feature that allows the user to send out a simulated encoder signal for use by an external device. The Encoder Simulation Source is used to define the source of the encoder signal. Any drive parameter in the form of a 0-65535 rollover counter can be used as the source parameter. Use this field to enter the desired drive source parameter (between 00.000 and 59.999).

By default, the PTi210 module configures the simulated output to work in Quadrature mode. In order to change the mode, the user will have to change the Drive Menu Initialization file.

Encoder Simulation Numerator

To add some flexibility to the Simulated Encoder Output signal, the SI-Universal Encoder module allows the user to scale the output signal by multiplying the source with a scaling factor. The scaling factor is made up of a Numerator and Denominator allowing the user to achieve nearly any ratio. By default, the Numerator and Denominator are both set to 65536 implying that the actual output value is equal to the Source value. Following is an equation that defines the use of the Numerator and Denominator parameters.

Simulated Encoder Output Signal = Simulated Encoder Source * (Numerator/Denominator)

Encoder Simulation Denominator

To add some flexibility to the Simulated Encoder Output signal, the SI-Universal Encoder module allows the user to scale the output signal by multiplying the source with a scaling factor. The scaling factor is made up of a Numerator and Denominator allowing the user to achieve nearly any ratio. By default, the Numerator and Denominator are both set to 65536 implying that the actual output value is equal to the Source value. Following is an equation that defines the use of the Numerator and Denominator parameters.

Simulated Encoder Output Signal = Simulated Encoder Source * (Numerator/Denominator)

Encoder Setup Parameters

Encoder Supply Voltage

Because of the wide variety of encoders supported by the Unidrive M/Digitax HD and the SI-Universal Encoder module, the user must have the ability to define the voltage supplied to the encoder hardware. The available voltage levels are 5 V, 8 V, and 15 V and are selectable from the list box on the Encoder view.

NOTE

Be sure not to configure a supply voltage greater than that supported by the encoder. The Unidrive M/Digitax HD and SI-Universal Encoder cannot protect against possible damage to the encoder if the supply voltage is set too high.

SSI Binary Format Select

This parameter is unique to the SSI encoder format. When using SSI encoders, the data is transmitted over the SSI network in one of two different formats. The user needs to select whether they wish to use the default format of Graycode, or switch to Binary format. Use this list box to select the desired format.

Encoder Turns

This parameter determines the maximum number of revolutions before the Rev Counter register (Pr **03.028** or **x**.**028**) rolls over. If using an absolute encoder, this should be set at the maximum number of turns of the absolute encoder or lower. The maximum number of turns is defined as follows:

Maximum # of Turns = 2^N, where N is the Encoder Turns parameter

Encoder Comms Resolution

Where encoder communications is used to initially read the absolute encoder position (SC.Hiper or SC.EnDat), the comms resolution must be set to the maximum resolution of the absolute position data.

Equivalent Lines Per Revolution

When sin/cos signals are used, the equivalent number of encoder lines per revolution must be set up correctly to give the correct speed and position feedback. The Equivalent Lines Per Revolution (ELPR) is defined as follows:

Position Feedback Device	ELPR
Ab	Number of lines per revolution
Fd, Fr[1]	Number of lines per revolution /2

[1] - Not currently supported in PowerTools Studio.

SC.Hiper, SC.EnDat, SC.SSI Number of singe waves per revolution

For SC.Hiper, SC.EnDat and SC.SSI encoders, the sine wave signal frequency can be up to 500 kHz, but the resolution is reduced at higher frequencies. The table below shows the number of bits of interpolated information at different frequencies and with different voltage levels at the drive encoder port. The total resolution in bits per revolution is the ELPR plus the number of bits of interpolated information.

Volt / Freq	1,000	5,000	50,000	100,000	150,000
1.2	11	11	11	10	10
1.0	11	11	10	10	9
0.8	10	10	10	10	9
0.6	10	10	10	9	9
0.4	9	9	9	9	8

If the Enable Auto Encoder Configuration check box is selected, the user does not need to enter this parameter and the Equivalent Lines Per Rev box will appear dimmed.

Lines Per Rev Divider

The Equivalent Lines Per Revolution parameter is divided by this value. This can be used when an encoder is used with a linear motor where the number of counts or sine waves per pole is not an integer.

Example:

The true number of encoder lines per rev (or electrical cycle) is 128.123. Since The Equivalent Lines Per Rev parameter must be a whole number, ELPR should be set to 128123, and then the Lines Per Rev Divider would be set to 1000 resulting in the following equation:

$$\text{Actual Encoder Lines Per Rev} = \text{ELPR/Lines Per Rev Divider} = 128123 / 1000 = 128.123$$

Encoder Comms Baud Rate
This parameter defines the baud rate for the encoder communications. The list box allows the user to select from various baud rates between 100 kbaud and 4 Mbaud.

Simulated Encoder Output Parameters

Encoder Simulation Source
The SI-Universal Encoder module has a feature that allows the user to send out a simulated encoder signal for use by an external device. The Encoder Simulation Source is used to define the source of the encoder signal. Any drive parameter in the form of a 0-65535 rollover counter can be used as the source parameter. Use this field to enter the desired drive source parameter (between 00.000 and 59.999).

By default, the PTi210 module configures the simulated output to work in Quadrature mode. In order to change the mode, the user will have to change the Drive Menu Initialization file.

Encoder Simulation Numerator
To add some flexibility to the Simulated Encoder Output signal, the SI-Universal Encoder module allows the user to scale the output signal by multiplying the source with a scaling factor. The scaling factor is made up of a Numerator and Denominator allowing the user to achieve nearly any ratio. By default, the Numerator and Denominator are both set to 65536 implying that the actual output value is equal to the Source value. Following is an equation that defines the use of the Numerator and Denominator parameters.

Simulated Encoder Output Signal = Simulated Encoder Source * (Numerator/Denominator)

Encoder Simulation Denominator
To add some flexibility to the Simulated Encoder Output signal, the SI-Universal Encoder module allows the user to scale the output signal by multiplying the source with a scaling factor. The scaling factor is made up of a Numerator and Denominator allowing the user to achieve nearly any ratio. By default, the Numerator and Denominator are both set to 65536 implying that the actual output value is equal to the Source value. Following is an equation that defines the use of the Numerator and Denominator parameters.

Simulated Encoder Output Signal = Simulated Encoder Source * (Numerator/Denominator)

8.5.5 SI-DeviceNet Module View
If DeviceNet is selected in the Slot # Module list box, the remainder of the view will have configuration parameters to define the properties of the DeviceNet device and network. The hierarchy tree will automatically update to show that a DeviceNet module is populated in that specific slot, see Figure 8-14.

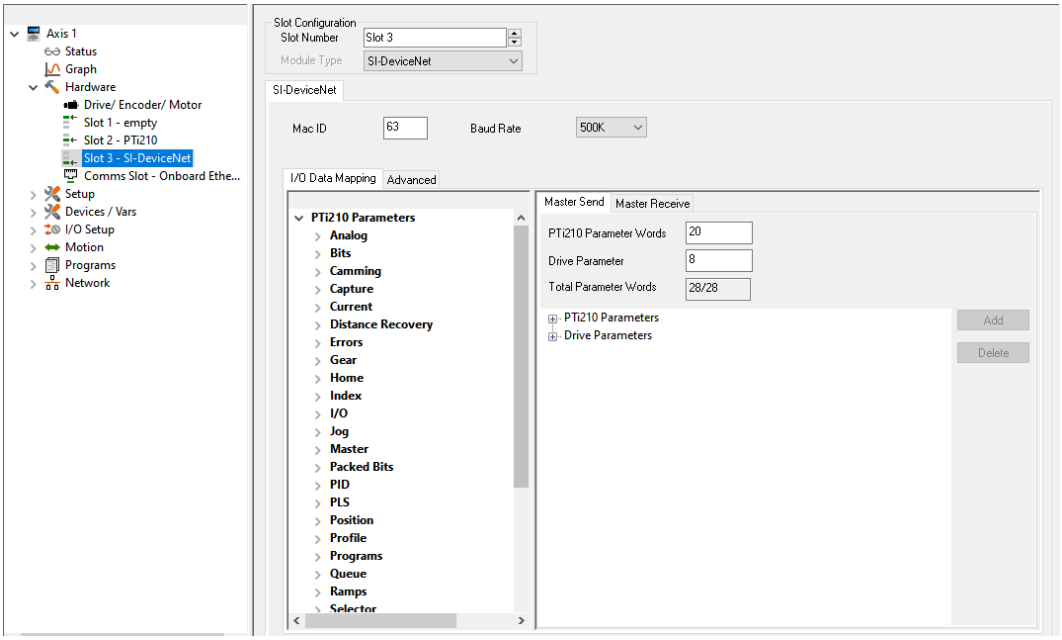


Figure 8-14: Slot # View (SI-DeviceNet Module)

Mac ID
The MacID is the number assigned to a particular DeviceNet node. Every node on a DeviceNet network must have a unique MacID. The range is 0-63.

Baud Rate
One of three standard baud rates can be configured for the DeviceNet network: 125 k, 250 k, and 500 k.

I/O Data Mapping Tab

The left side of the view contains a list of the PTi210 module and drive parameters that may be mapped to the words on the Master Send or Master Receive tabs on the right by dragging and dropping. PTi210 module parameters from the variables parameters list can only be mapped to PTi210 module parameter words, and drive parameters to drive parameter words.

PTi210 Parameter Words

There are a maximum of 20 PTi210 parameter words available, the default value is 20.

Drive Parameter Words

There are a maximum of 8 Drive parameter words available, 8 is the default.

Total Parameter Words

The total number of parameter words available is 28, these may be divided between PTi210 parameter words and Drive parameter words. The text box shows the number of words allocated vs. the total number available.

Advanced Tab

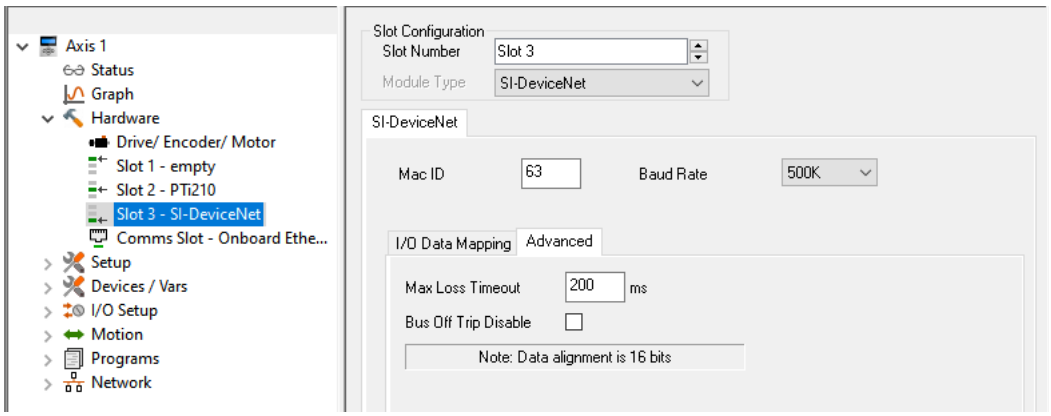


Figure 8-15: Slot # View (SI-DeviceNet Advanced Tab)

Max Loss Timeout

This provides a method on the drive to ensure that communication with the DeviceNet network is still present. The SI-DeviceNet module resets an internal timer when a valid message is received from the DeviceNet network, if a message is not received within the specified period of time, in ms, the network loss trip is triggered. Default value is 200 ms, but the range is 0 to 3000 ms.

WARNING	The network loss trip can be disabled by setting this parameter to 0. In this case, the drive will continue to operate using the last received values. It is the user's responsibility to ensure that adequate safety precautions are taken to prevent damage or injury by disabling the drive in the event of a loss of communications.

Bus Off Trip Disable

When selected the SI-DeviceNet module will not trip the drive when there is a network fault. See the *SI-DeviceNet User Guide* for more information.

8.5.6 SI-Profibus Module View

If Profibus DP is selected in the Slot # Module list box, the remainder of the view should have configuration parameters to define the properties of the Profibus device and network. The hierarchy tree automatically updates to show that a Profibus module is populated in that specific slot, see Figure 8-16.

For SI-Profibus module installation instructions, or other SI-Profibus module information, please refer to the *SI-Profibus User Guide*.

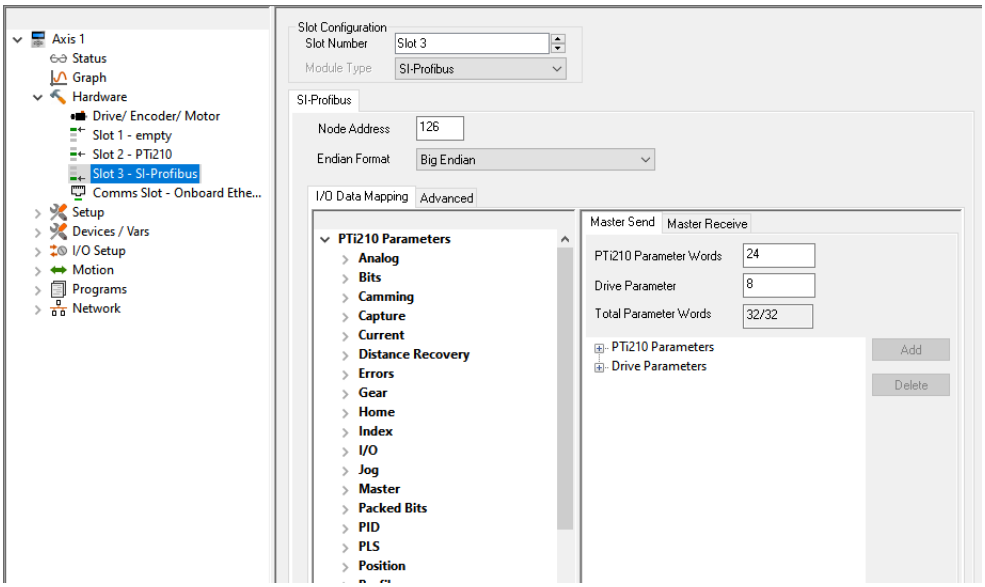


Figure 8-16: Slot # View (SI-Profibus Module)

Node Address

The Node Address is the number assigned to a particular node on the Profibus network. Every node on a Profibus network must have a unique node address with a range between 0 to 126.

Endian Format

When data is sent over the Profibus network it is transmitted as 8-bit bytes. Therefore when a 32-bit word or 16-bit word is transmitted it is split into four or two 8-bit bytes. It is important that the receiving node reconstruct the received 8-bit bytes in the correct order to arrive at the 32-bit or 16-bit data value that was originally transmitted, this order is known as the "Endian Format".

Data Endian Format	16-bit Value	32-bit Value	
	Byte Order	Word Order	Byte Order
Big	High byte first Low byte second	High word first Low word second	High byte first Mid high byte second Mid low byte third Low byte fourth
Little	Low byte first High byte second	Low word first High word second	Low byte first Mid low byte second Mid high byte third High byte fourth

Most Profibus master controllers use big endian format by default, many also support little endian. The default configuration of Big Endian is consistent with the way most Profibus Master PLCs transfer their data.

I/O Data Mapping Tab

The I/O Data Mapping tab is used to configure the data that will be sent and received from the Profibus Master (PLC) to the Profibus slave (SI-Profibus module). The Master Receive tab configures the data from the PTi210 module drive to the PLC.

Master Send Tab/Master Receive Tab

Individual parameters are mapped by dragging and dropping the parameter from the Variables list to the desired word. PTi210 parameters from the variables list can only be mapped to PTi210 parameter words, and Drive parameters to Drive parameter words.

PTi210 Parameter Words

There are a maximum of 32 PTi210 parameter words available, 24 is the default.

Drive Parameter Words

There are a maximum of 8 Drive parameter words available, the default is 8.

Total Parameter Words

The total number of parameter words available is 32, These may be divided between PTi210 parameter words and Drive parameter words. This text box shows the number of words allocated vs. the total number available.

Advanced Tab

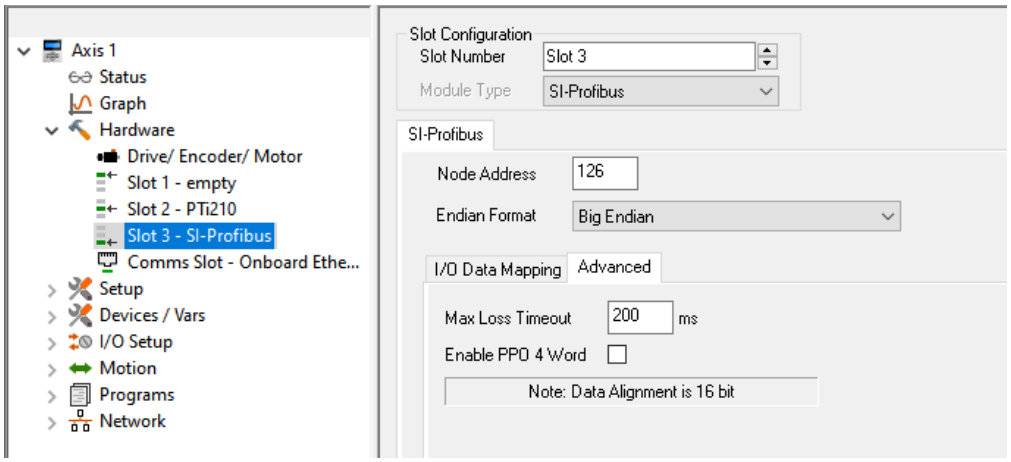


Figure 8-17: Slot # View (SI-Profibus Advanced Tab)

Max Loss Timeout

This provides a method on the drive to ensure that communication with the Profibus master is still present. The SI-Profibus module resets an internal timer when a valid message is received from the profibus network, if a message is not received within the specified period of time, in ms, the network loss trip is triggered. Default value is 200 ms, but the range is 0 to 3000 ms.

WARNING	The network loss trip can be disabled by setting this parameter to 0. In this case, the drive will continue to operate using the last received values. It is the user's responsibility to ensure that adequate safety precautions are taken to prevent damage or injury by disabling the drive in the event of a loss of communications.

Enable PPO 4 Word

When this check box is selected the PPO 4 Word mode is enabled. For more information see the *SI-Profibus User Guide*.

8.5.7 SI-I/O 24 Plus Module View

As mentioned previously, the SI-I/O 24 Plus option module provides an additional sixteen digital inputs, eight digital outputs and an encoder input, however, currently only AB and AB Servo encoders are supported.

If SI-I/O 24 Plus is selected in the Slot # Module list box. The hierarchy tree automatically updates to show that an SI-I/O 24 Plus module is populated in that specific slot, see Figure 8-18.

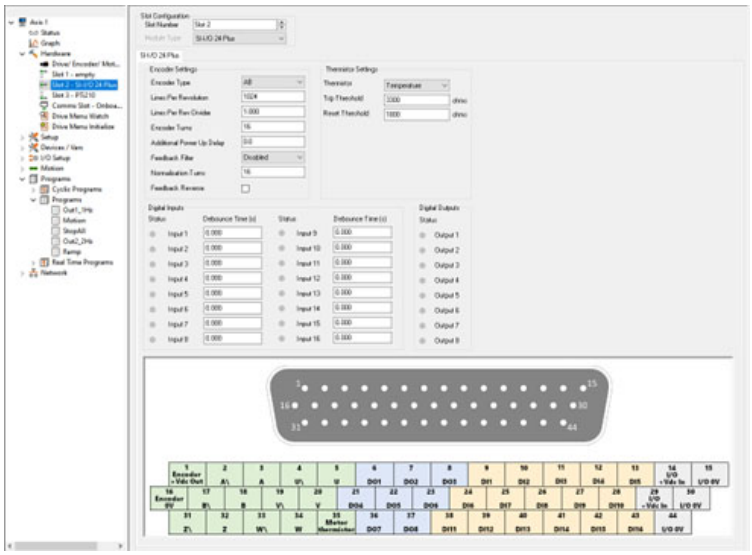


Figure 8-18: Slot # View (SI-I/O 24 Plus Module)

SI-I/O 24 Plus tab

The SI-I/O 24 Plus tab contains the parameters to configure the digital inputs and encoder input.

To the left of each digital input is an indication of the state of that input, this indication will be green when the input is active.

Each digital input has an associated Debounce Time (s) setting, this setting determines the amount of time (seconds) that the digital input must be active for the digital input state to be accepted. The maximum time that can be set is 2.000 seconds with a resolution of 1 millisecond.

The digital output states are also indicated in this view, green being active.

For more information on the SI-I/O 24 Plus please refer to the SI-I/O 24 Plus User Guide.

8.5.8 Comms Slot - Onboard Ethernet (Unidrive M700/M702 and Digitax HD M750)

If the configured drive is fitted with an onboard Ethernet interface (Unidrive M700/M702 and Digitax HD M750), this option will be shown in the hierarchy tree. If selected the remainder of the view should have configuration parameters for the Ethernet communications, see Figure 8-19.

Config

This section allows the user to configure the Ethernet communication IP address.

By default, the IP address settings are disabled and set to a fixed (DHCP disabled) address of 192.168.1.100 with a subnet mask of 255.255.255.0, if these need to be changed then the Change Ethernet Settings check box must be selected, this will enable the IP Address configuration settings allowing the user to specify if the Ethernet configuration should use a DHCP server to assign the IP Addresses or whether they will be specified by the user.

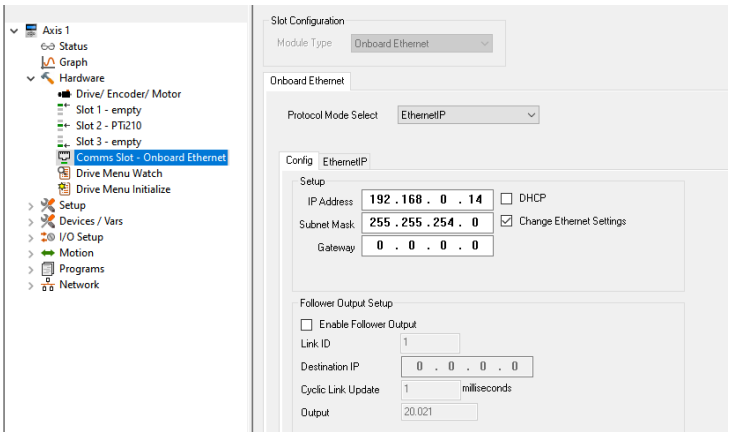


Figure 8-19: Comms Slot View (Onboard Ethernet)

IP Address

This is a 32-bit identification number for each node on an Ethernet (Internet Protocol) network. These addresses are represented as four 8-bit numbers (0 to 255), with periods between them. Each node on the Ethernet network must have a unique IP address. For detailed information see the relevant drive *User Guide*.

Subnet Mask

This 32-bit parameter indicates the Subnet mask used for this node. The subnet mask is used to group devices that are connected on the same physical connection. For a detailed description of Subnet mask refer to the relevant drive *User Guide*.

Gateway

This 32-bit parameter indicates the default Gateway address for the Ethernet interface. When attempting to communicate with a device on a different Subnet, the message must go through this gateway to reach its destination. For a detailed description of the Gateway address refer to the relevant drive *User Guide*.

Change Ethernet Settings Check Box

The Change Ethernet Settings check box is used to determine if the user wants to use the scanner to determine the application's Ethernet address. If Change Ethernet Settings check box is selected then the scanner selected address becomes the new application address (saved with application). This applies for Downloads, Uploads into an existing application and Change connection path. The scanner's IP Address Scan Range will also be loaded to select only the application's IP Address.

When Change Ethernet Settings check box is clear the Ethernet Address scanner range is the last scan range entered using the scanner's "Stop Scan".

The Change Ethernet Settings check box must be selected to change the IP Address, Subnet Mask, Gateway and DHCP selection.

DHCP

The IP Address settings used when PowerTools Studio communicates with the drive can be manually configured from the Config section settings or automatically by a DHCP server connected on the same subnet. This check box when selected and the application downloaded to the PTi210 module, will force the drive to search for a DHCP server to obtain the IP Address settings. If this check box is not selected then the configured IP Address settings will be used.

Please note that if the drive IP Address fails to be configured from a DHCP server, then the PTi210 module must be defaulted to delete the application from the PTi210 module, and the drive IP Address must also be configured to allow communication with PowerTools Studio.

For more information on the DHCP configuration please see the relevant drive User Guide.

Follower Output Setup

PowerTools Studio allows the user to configure a Master/Follower Digital Lock application based on the RTMoE communication protocol using an Easy Mode cyclic data link to transfer the master position value to the follower drive.

Enable Follower Output

Checking this box causes the PTi210 to set up the onboard Ethernet interface to transmit the demanded axis position on an RTMoE easy mode cyclic link. The transmitted position can then be used on a follower PTi210 as a master reference.

The master PTi210 module will configure the Tx1 publishing link parameters on the master drive, the follower drive PTi210 module will need to be configured to use the same Easy Mode cyclic link number and will configure the Rx1 subscribing link parameters on the follower drive, all other Easy Mode cyclic links on both the publishing and subscribing drives will remain unchanged.

NOTE

Care should be taken to ensure no other Easy Mode cyclic link (or user program) is configured to write to the source parameter used for the master or destination parameter on the follower, this may adversely affect the master/follower setup.

The PTi210 will configure the onboard Ethernet interface to transmit the position to a follower drive using the settings described below.

Link ID

This is the RTMoE link number (1 to 255) used to transmit the axis position to the follower drive.

This link number must match the link number used on the follower drive to receive the master reference.

Destination IP

This is the IP address of the follower drive.

Cyclic Link Update

This is the cyclic data rate (ms) for transmission of the axis position to the follower.

The PTi210 will create a non-synchronous cyclic data link, the update rate range is from 1 ms to 100 ms.

Output

This is the parameter on the master drive that the PTi210 will place the axis position into for the onboard Ethernet interface to transmit to the follower drive.

NOTE

It is recommended that the local parameter used in both the master and follower drives are of the same type (e.g. 32-bit), this will avoid any scaling problems which may be encountered if different parameter types are used.

Digital Lock Example

In this example of a master/follower digital lock system the master drive is a Unidrive M700 fitted with a PTi210 module in slot 3, and the follower drive is a Digitax HD (M750) with the PTi210 module in slot 1.

For both the master and follower drives, the Ethernet communication must be manually configured (the PTi210 does not configure this) and the local application parameter Pr **20.021** is used to store the position value which will be transmitted every 1 millisecond by the master drive.

Master - Unidrive M700

Ethernet Configuration

The master drive Ethernet communication is configured with an IP address of 192.168.1.100 and a subnet mask of 255.255.255.0 with DHCP disabled.

For the follower drive Ethernet communication, an IP address of 192.168.1.101 will be used with the cyclic link ID of 1.

The master drive Ethernet configuration is shown in Figure 8-20.

Figure 8-20: Digital Lock Master Ethernet Settings

The master PTi210 will create a unicast cyclic data link (number 1) to the follower drive IP address 192.168.1.101 and write the position value to Pr **20.021** every 1 millisecond (e.g. a value of 36000 will equate to 3600.0 degrees).

User Units Configuration

The master drive User Units is configured for 360 degrees per revolution with a 0.1 degree resolution as shown in Figure 8-21.

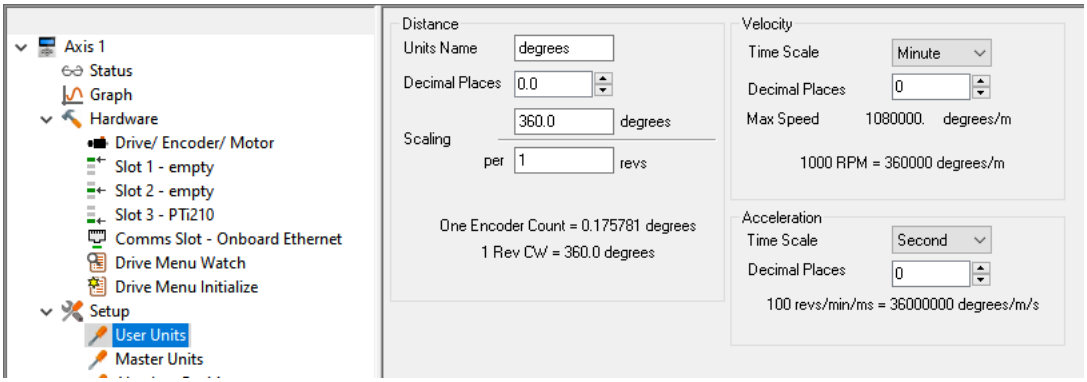


Figure 8-21: Digital Lock Master User Units Configuration

Assignments

The master drive digital inputs are used to set the home position and initiate a repeating series of indexes, each index is initiated upon completion of the previous index.

Digital input 2 (terminal 25) is used to set the axis home position, and digital input 3 (terminal 26) is used to start the motion sequence.

Motion Sequence Assignments

The motion sequence consists of four index motion sequences (0 to 3) as follows:

- Index0.CommandComplete → Index1.Initiate Active on
- Index1.CommandComplete → Index2.Initiate Active on
- Index2.CommandComplete → Index3.Initiate Active on
- Index3.CommandComplete → Index0.Initiate Active on

Digital Input Assignments

The master drive digital input DIO2 is used to set the axis home position by assigning it to the destination *Position > DefineHome*.

- DriveIO.2.In → DefineHome Active on
- → UndefineHome Active Off

The master drive digital input DIO3 is used to start the motion sequence.

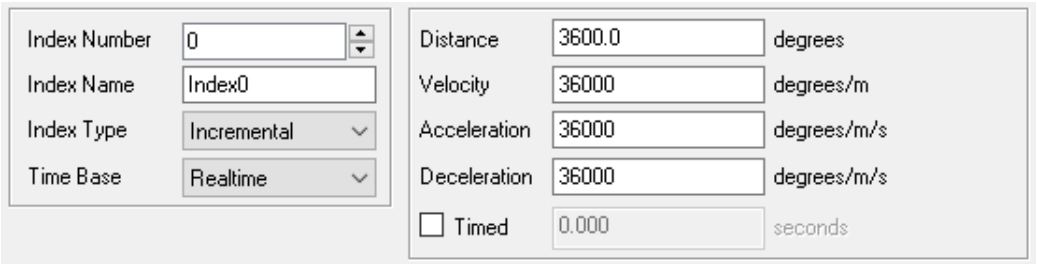
- DriveIO.3.In → Index0.Initiate Active on

Index Configuration

Index0

Index0 has a distance of 3600 degrees with a velocity of 36000 degrees/min and acceleration/deceleration values of 36000 degrees/m/s, which equates to 100 rpm.

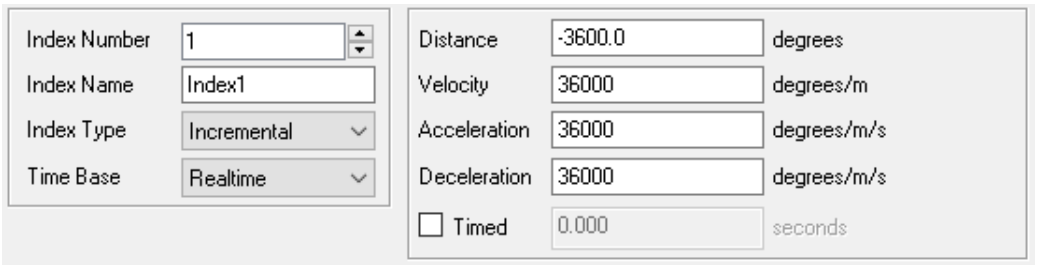
The motor will turn CW.



Index1

Index1 has a distance of -3600 degrees with a velocity of 36000 degrees/min and acceleration/deceleration values of 36000 degrees/m/s, which equates to 100 rpm.

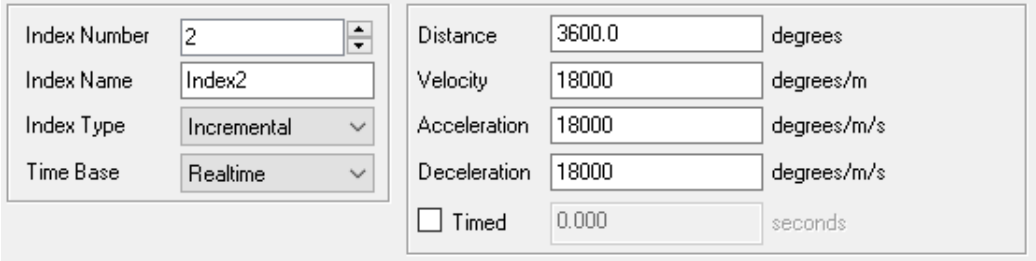
The motor will turn CCW.



Index2

Index2 has a distance of 3600 degrees with a velocity of 18000 degrees/min and acceleration/deceleration values of 18000 degrees/m/s, which equates to 50 rpm.

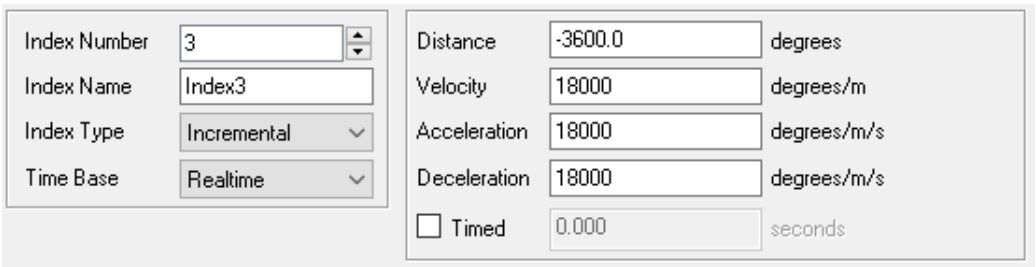
The motor will turn CW.



Index3

Index3 has a distance of -3600 degrees with a velocity of 18000 degrees/min and acceleration/deceleration values of 18000 degrees/m/s, which equates to 50 rpm.

The motor will turn CCW.



Follower - Digitax HD M750

Ethernet Configuration

The follower drive Ethernet communication is configured with an IP address of 192.168.1.101 and a subnet mask of 255.255.255.0 with DHCP disabled.

No Follower Output Settings need to be configured.

The master drive Ethernet configuration is shown in Figure 8-22

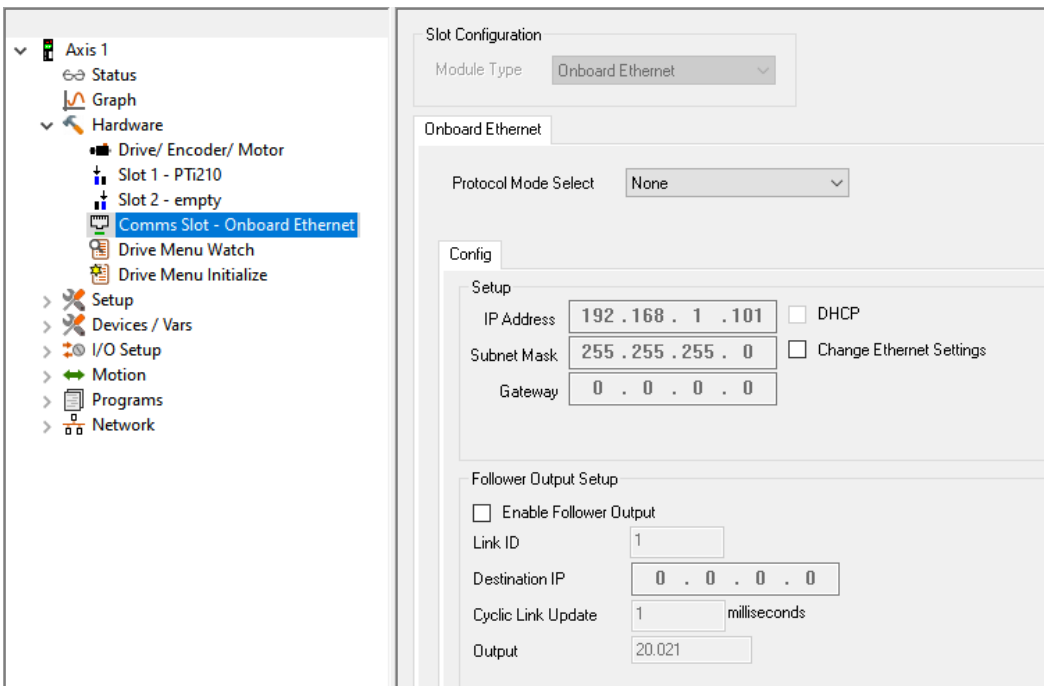


Figure 8-22: Digital Lock Follower Ethernet Settings

User Units Configuration

The follower drive User Units is configured for 360 degrees per revolution with a 0.1 degree resolution as shown in Figure 8-23.

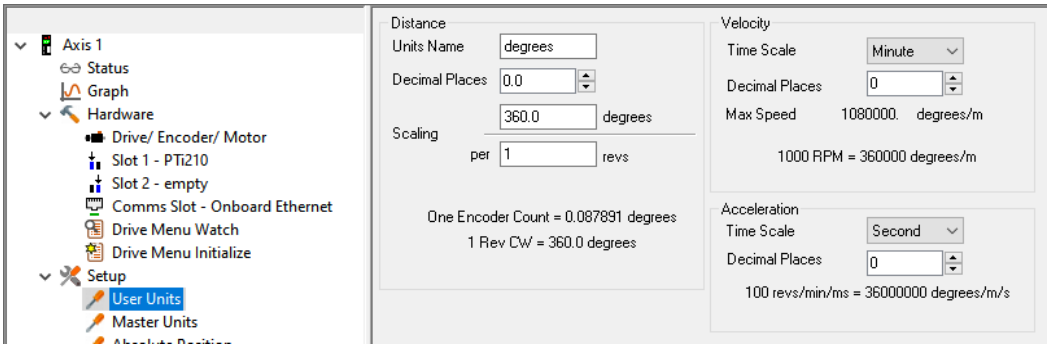


Figure 8-23: Digital Lock Follower User Units Configuration

Master Units Configuration

The follower drive Master Units is configured to use the Easy Mode RTMoE Master cyclic data link (number 1) as shown in Figure 8-24.

The Master Distance Units is 1 decimal place precision and the scaling is 0.1 degrees per 1 feedback source unit. This means that for 1 unit in Pr **20.021** from the master, this equates to 0.1 degrees on the follower.

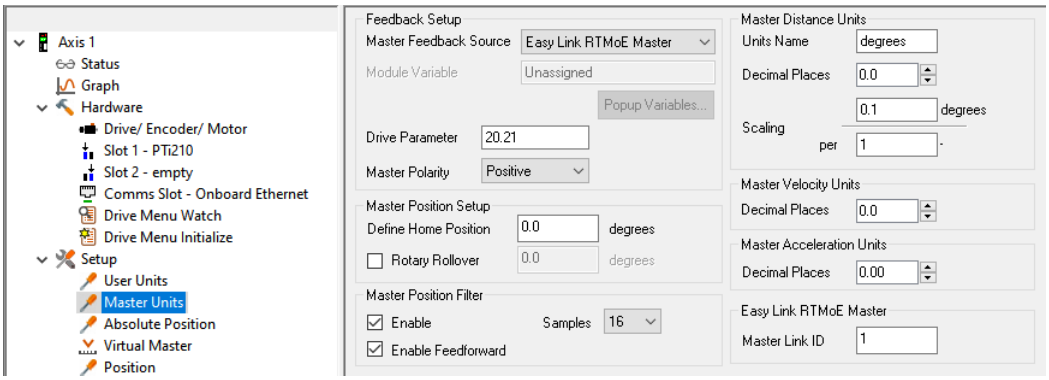


Figure 8-24: Digital Lock Follower Master Units Configuration

Assignments

The follower drive digital inputs are used to set the home position and enable the gearing (following the master position).

Digital input 4 (terminal 11) is used to set the axis home position, and digital input 5 (terminal 13) is used to enable the gearing motion.

Digital Input Assignments

The follower drive digital input DI4 is used to set the axis home position by assigning it to the destination Position > DefineHome.

• DriveInput.4	→	DefineHome	Active on
•	→	UndefineHome	Active Off

The follower drive digital input DI5 is used to enable the gearing.

• DriveInput.5	→	Gear.Active	Active on

Gearing Configuration

The follower drive gearing configuration is set to a Gear Ratio of 1:1 as shown in Figure 8-25.

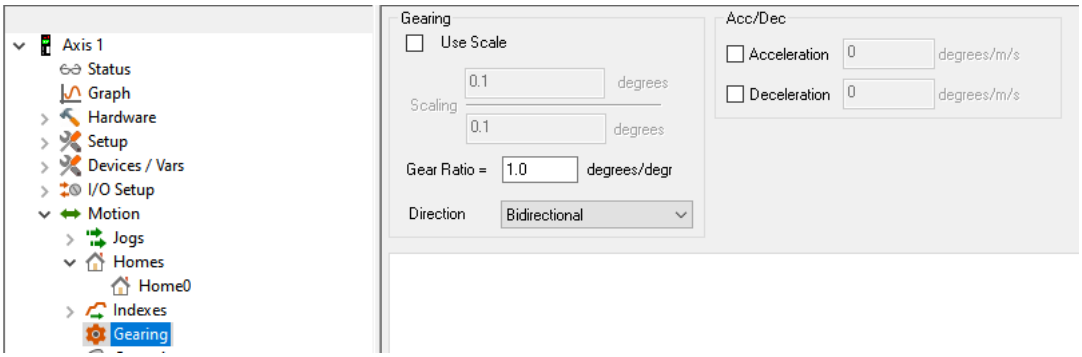


Figure 8-25: Digital Lock Follower Gearing Configuration

Operation

After downloading each project to the appropriate drive:

- Enable both drives
- Enable gearing on the follower drive by activating digital input 5 (terminal 13)
- Start the motion sequence on the master drive by activating digital input 3 (terminal 26)

The follower drive should follow the master position.

Protocol Mode Select

The Unidrive M700/M702 and Digitax HD M750 Ethernet interfaces support one real-time cyclic communication task in addition to standard Ethernet, the supported options are “None”, “EthernetIP” and “ProfiNet”.

This setting directly controls the Protocol Mode Select parameter in the drive (Pr **S.02.018**).

If “EthernetIP” is selected then the EtherNet/IP cyclic mapping configuration will be shown where the user can configure the EtherNet/IP cyclic mapping details.

EtherNet/IP Mapping

When Protocol Mode Select is set to “EthernetIP”, the EtherNet/IP mapping page will be available, see Figure 8-26.

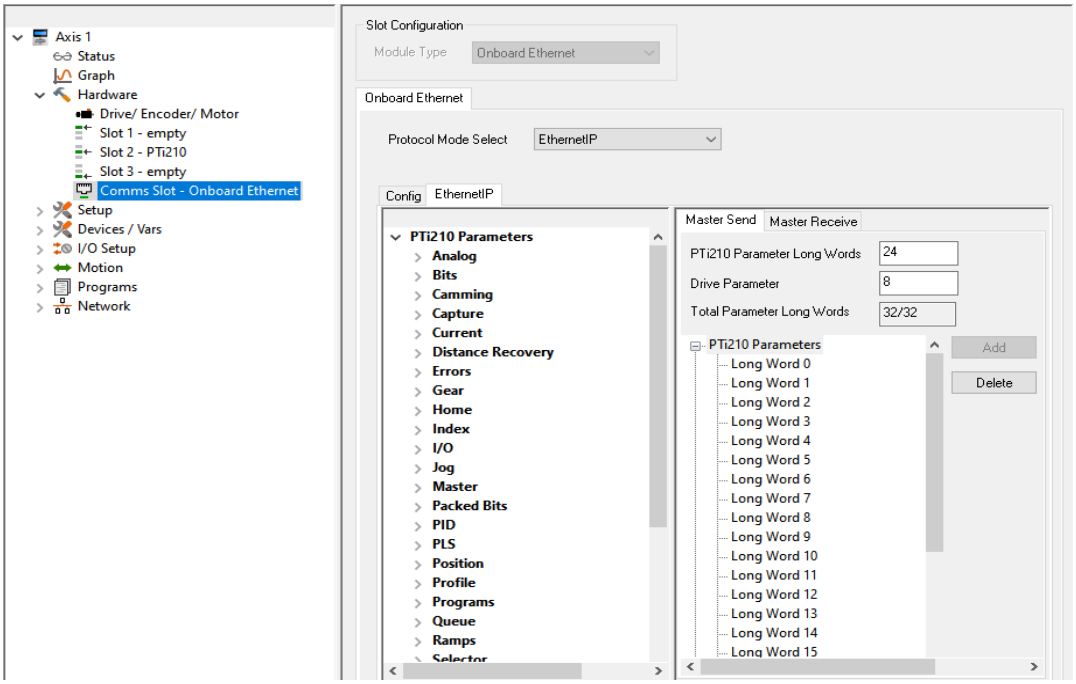


Figure 8-26: EtherNet/IP Default Mapping View

Master Send Group

The Master Send group contains the EtherNet/IP cyclic output mapping configuration for the EtherNet/IP master data sent to the drive or PTi210.

PTi210 Parameter Words

This parameter specifies the number of EtherNet/IP cyclic Long Words allocated to the PTi210 Module parameters. The default number of PTi210 Long Words is 24.

Drive Parameter

This parameter specifies the number of EtherNet/IP cyclic Long Words allocated to the drive parameters. The default number of Drive Parameter Long Words is 8.

Total Parameter Words

This displays the total number of cyclic Long Words configured for both the PTi210 module and the drive, together with the maximum number of Long Words available. The displayed format is CC/MM, where CC represents the total number of configured words and MM is the maximum number of Long Words available.

A maximum number of 32 output Long Words are possible shared between the PTi210 and drive parameters.

Master Receive Group

The Master Receive group contains the EtherNet/IP cyclic input mapping configuration for the EtherNet/IP master data sent from the drive or PTi210.

PTi210 Parameter Words

This parameter specifies the number of EtherNet/IP cyclic Long Words allocated to the PTi210 Module parameters. The default number of PTi210 Long Words is 24.

Drive Parameter

This parameter specifies the number of EtherNet/IP cyclic Long Words allocated to the drive parameters. The default number of Drive Parameter Long Words is 8.

Total Parameter Words

This displays the total number of cyclic Long Words configured for both the PTi210 module and the drive, together with the maximum number of Long Words available. The displayed format is CC/MM, where CC represents the total number of configured Long Words and MM is the maximum number of Long Words available.

A maximum number of 32 input Long Words are possible shared between the PTi210 and drive parameters.

Adding a mapping

To add a mapping, the user has two choices:

- Drag and drop the required parameter data from the left hand pane parameters to the required parameter in the right hand Master mapping pane. The cursor icon will change from the unavailable icon (⃠) to the normal cursor arrow with a plus sign below it () when the cursor is over a valid parameter item or
- Select the required parameter data from the left hand pane parameters
    - Select the required parameter in the right hand Master mapping pane
    - Click the Add button, the selected item will be added to the Master mapping pane at the selected location.

NOTE

Please note, PowerTools Studio will automatically map the source parameter to two words if it is larger than 16-bit.

Deleting a mapping

To delete a mapping, the user must select the required parameter data in the right hand Master mapping pane and click the Delete button.

When the application is downloaded to the PTi210 module, the relevant EtherNet/IP parameters will be configured accordingly and the drive parameters will be saved before the Ethernet interface is reset to activate the changes.

8.5.9 Drive Menu Watch View

The Drive Menu Watch View is only available when the **Show Advanced Views** is selected in the **Options** > **Preferences** menu.

This view allows the user to view the online value of all the Unidrive M/Digitax HD menu parameters as well as modify the value of a menu parameter. Figure 8-27 shows an example of the Drive Menu Watch view.

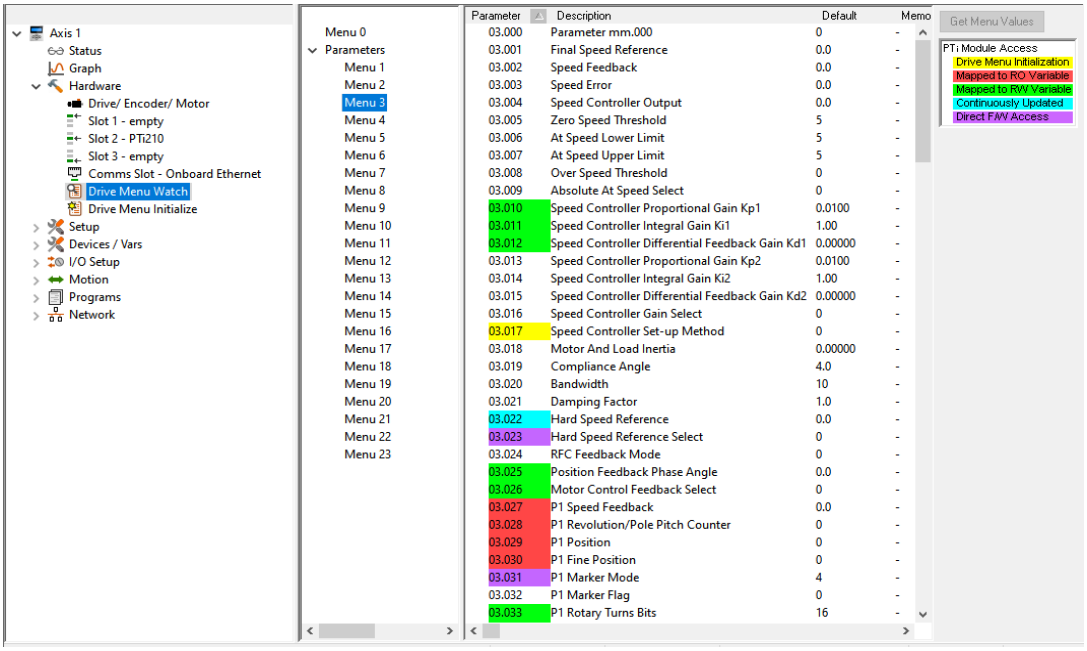


Figure 8-27: Drive Menu Watch View - PTi210 Application

To view the menu parameters, PowerTools Studio must upload the parameter values. To upload the parameters, click ***Get Menu Values*** on the right side of the view. The values displayed are only the values at the time the ***Get Menu Values*** button was selected. The values are NOT continuously updating.

Get Menu Values Button

Click ***Get Menu Values*** and PowerTools Studio will read the current value of all the parameters in the selected menu and display them in the memory column. If the value in the memory is different from the default value. The parameter value will be highlighted in yellow in the default column.

8.5.10 Drive Menu Initialize View

The Drive Menu Initialize View is only available when the **Show Advanced Views** is selected in the **Options** > **Preferences** menu.

The Drive Menu Initialize View is a utility to aid the user in configuring the Unidrive M/Digitax HD drive setup. Because the Unidrive M/Digitax HD can operate in many different modes, and has many different features, it must be put into a known state so that the PTi210 module can control it. To get into this state, certain menu parameters must be set to specific values. The Drive Menu Initialize View is simply a list of parameters that the PTi210 module writes to the drive on powerup so that the drive is in a known state so the PTi210 module can control it.

A default list of parameters is included so the user does not need to enter each of these parameters by hand. The user can modify the default list, adding new parameters or removing some of the parameters. If the user makes changes, and then wishes to revert back to the original default list, the ***Reset to Defaults*** button will restore the original list.

Figure 8-29 shows an example of the Drive Menu Initialize view.

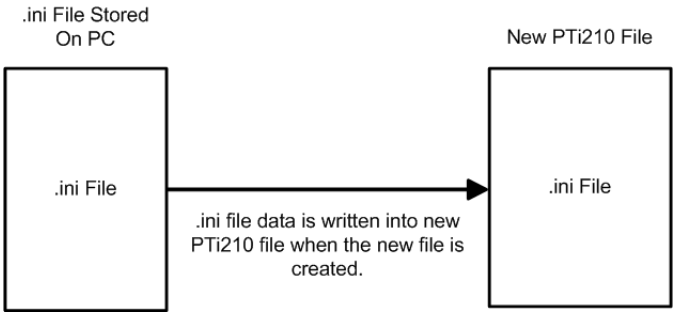


Figure 8-28: Initialization File Example

The default contents of the Drive Menu Initialize view is generated from a file titled DriveInitialize.ini. This file is installed to your PC as part of the PowerTools Studio installation. When a new PTi210 module configuration file is created using PowerTools Studio, the contents of DriveInitialize.ini is read and written into the configuration file.

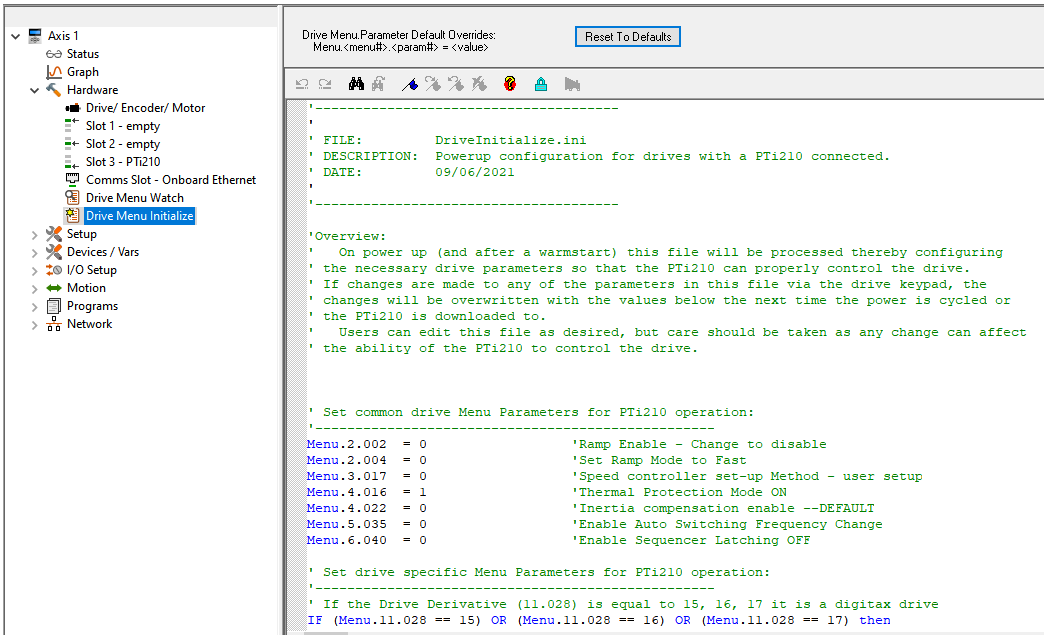


Figure 8-29: Drive Menu Initialize View

Once the data is written from the DriveInitialize.ini file into the new configuration, it is stored as part of the configuration. Therefore, if the file is downloaded to the PTi210 module, the contents of the Drive Menu Initialize view resides in the module. If the file is uploaded using PowerTools Studio, the contents of the Drive Menu Initialization list is uploaded as part of the configuration file.

The Drive Menu Initialize data can be modified just like a user program. If the user wishes, new parameters can be added to the default data, or files can be removed from the default data. Changes made to the Drive Menu Initialize data in PowerTools Studio will change the initialization for that configuration file only (not for another new configuration created later).

If the user wishes to make a change to the initialization for every new file they create, then changes can be made directly to the source DriveInitialize.ini file. Using a text editor (i.e. Microsoft™ Notepad), the .ini file can be modified to include new parameters or remove existing default parameters. Once the modified .ini file is saved, those changes will be included in every new configuration file created in the future.

Care should be taken when editing the default list. Changes made to this file will directly impact the functionality of the system. It could be possible to cause a condition where the PTi210 module can not control the drive if an incorrect change is made to the initialization list. Consult CT Applications Engineering with questions.

8.6 Configure Setup Parameters

Following is a list of the different views in the Setup group on the hierarchy tree. The Setup group is dedicated to configuring the operation parameters for the system such as User Units, Position Limits, Torque Limits, Tuning Values, PLS points, etc. To complete the application, start with the Setup view and work down to the last view in the Setup group (User Bits).

8.6.1 Setup View

The Setup view allows the user to setup various parameters related to how the overall system operates. Figure 8-30 shows an example of the Setup view.

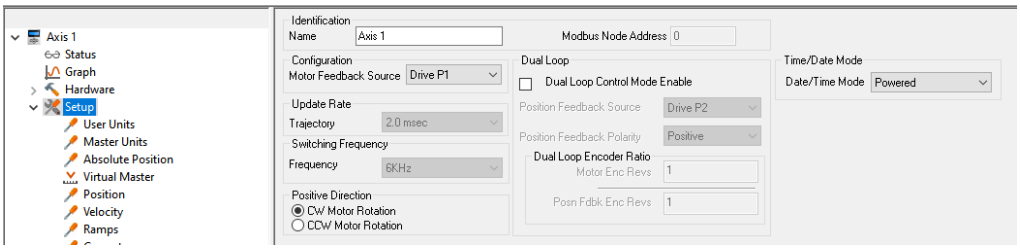


Figure 8-30: Setup View

Identification Parameters

Name

This is a 12-character alpha/numeric user-configured name for this axis. Enter the name for the device you are currently setting up. Assigning a unique name for each device in your system allows you to quickly identify a device when downloading, editing, and troubleshooting. All keyboard characters are valid. This will default to Axis 1.

Modbus Node Address

This is the Modbus serial address of the target drive to which you will download the configuration. The default target drive address is 1.

Configuration

Motor Feedback Source

Motor Feedback Source allows the user to specify where the motor feedback device is connected to the drive or SI-Universal Encoder module.

NOTE

When the Dual Loop Mode feature is used, this specifies the location of connection for the motor feedback device even though a different feedback device is being used to close the position loop.

Update Rate

Trajectory

This parameter configures the interrupt interval for the processor. This defines how often the motion program is interrupted and the Control Loop is processed. In the Control Loop, the feedback information is processed and a new position command is generated. Also in the Control Loop, the I/O is scanned.

Available selections for Trajectory Update Rate are 1, 1.25, 1.5, 1.75, 2, 2.25 and 2.5 milliseconds. The longer the update rate, the more time is dedicated to the user programs, and the less time dedicated to servo performance. The shorter the update rate, the more precise the servo performance, but less time is available to process user programs. Diagnostics are available on the Status Online tab when online with the device to help select the ideal setting. (See description of Control Loop Group of online parameters for further information).

Switching Frequency

Frequency

This parameter defines the switching frequency of the drive. Available switching frequencies are 2 kHz, 3 kHz, 4 kHz, 6 kHz, 8 kHz, 12 kHz and 16 kHz. For more information on how the switching frequency effects drive performance refer to the relevant drive *User Guide*.

Positive Direction

The Positive Direction settings consists of a CW (clockwise) Motor Rotation radio button and a CCW (counter-clockwise) Motor Rotation radio button.

The motion will move in either CW direction or CCW direction depending on the direction selected. Perspective of rotation is defined as you face the motor shaft from the front of the motor.

CW Motor Rotation Radio Button

Select this radio button for applications in which CW motor rotation is considered to be motion in the positive direction (increasing absolute position).

CCW Motor Rotation Radio Button
Select this radio button for applications in which CCW motor rotation is considered to be motion in the positive direction (increasing absolute position).

Dual Loop Group

Dual Loop Control Mode Enable Check Box
Select this check box to enable the Dual Loop Mode feature which accepts a secondary feedback device to close the position loop while still using the motor encoder to close the velocity loop.

Position Feedback Source
This list box is used to define the physical connection location of the Position Feedback Encoder to the drive system. This is different from the Motor Feedback Source.

Position Feedback Polarity
This list box is used to define which direction the Position Feedback Encoder counts when the feedback device moves in the "positive" direction. Positive = Counts Up, Negative = Counts Down.

Dual Loop Encoder Ratio

Motor Enc Revs
This parameter is the numerator in the ratio used to define the mechanical ratio between the Motor Encoder and the Position Feedback Encoder. This parameter is only used when Dual Loop Control Mode is enabled, and must be set correctly to achieve the correct target velocity.

Posn Fdbk Enc Revs
This parameter is the denominator in the ratio used to define the mechanical ratio between the Motor Encoder and the Position Feedback Encoder. This parameter is only used when Dual Loop Control Mode is enabled, and must be set correctly to achieve the correct target velocity.

Time/Date Mode

Date/Time Mode
This parameter specifies which clock the drive's Date and Time parameters, (Pr **06.016** and Pr **06.017** respectively), use as the source clock for the PTi210 RunTime and PowerUpTime parameters. Options available are Powered and Running.

8.6.2 User Units View
The User Units view allows the user to configure the distance, velocity, and acceleration units to be used for the motor axis throughout the application. Figure 8-31 shows an example of the User Units View.

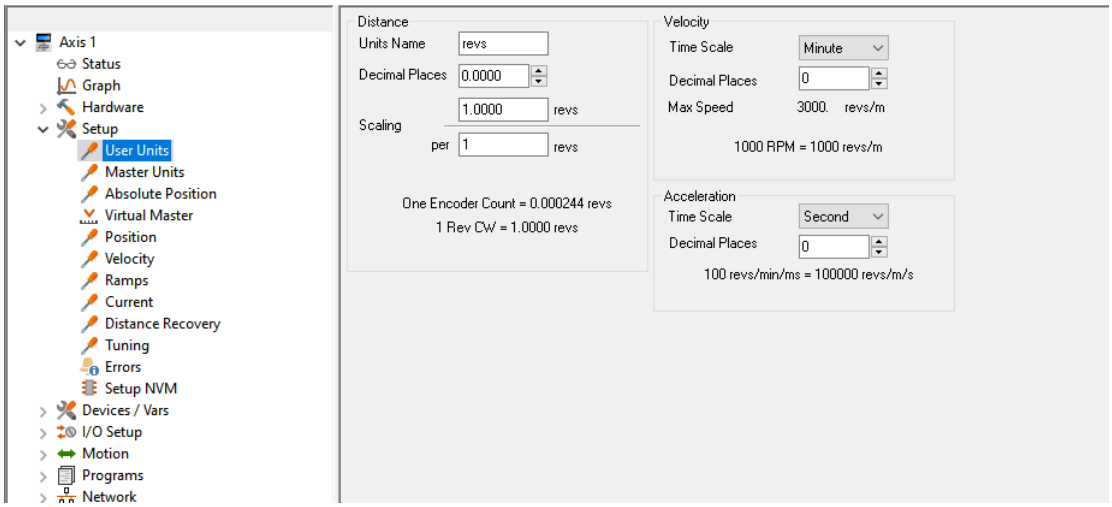


Figure 8-31: User Units View

Distance

Units Name
This is a 12 character name for the distance user units you want to use in your application.

Decimal Places
The number of decimal places set in this parameter determines the number of digits after the decimal point used in all distance and position parameters throughout the software. Using a high number of decimal places will improve position resolution, but will also limit the range of absolute position. You can select from zero to six decimal places of accuracy.

Scaling
A Characteristic Distance and Length must be established to allow the module to scale user units back to actual motor revolutions. This scaling factor is as follows:

$$\text{Scaling} = \frac{\text{Characteristic Distance}}{\text{Characteristic Length}}$$

Scaling - Characteristic Distance
This parameter is the distance the load travels (in user units) when the motor travels the characteristic length (in motor revolutions). This parameter is used along with the DistUnits.CharacteristicLength to establish the relationship between user distance and actual motor travel distance.

Scaling - Characteristic Length
This parameter is the distance the motor travels (in whole number of revolutions) to achieve one characteristic distance of load travel. This parameter is used along with the DistUnits.CharacteristicDist to establish the relationship between user distance and motor travel distance.

Velocity

Time Scale List Box
The time can be one of two values: seconds or minutes. This selection sets the real-time velocity time scale.

Decimal Places
The number of decimal places defined in this parameter determines the maximum resolution of all real-time velocity parameters found throughout the PowerTools Studio software. Set between 0 and 6 decimal places. Higher number of decimal places allows higher velocity resolution, but can limit the maximum speed allowed by the application.

Acceleration

Time Scale List Box
From this list box, select the acceleration time scale to be used for all real-time profiles. The time scale selected will be used for both acceleration and deceleration parameters. You can select from milliseconds or seconds.

Decimal Places
The number of decimal places defined in this parameter determines the maximum resolution of all real-time acceleration and deceleration parameters found throughout the software. Set between 0 and 6 decimal places.

8.6.3 Master Units View
The Master Units view is used to configure the parameters for the master axis used in synchronized motion applications. The master axis is most often a second encoder, or possibly another upstream drive. Figure 8-32 shows an example of the Master Units view.

Feedback Setup

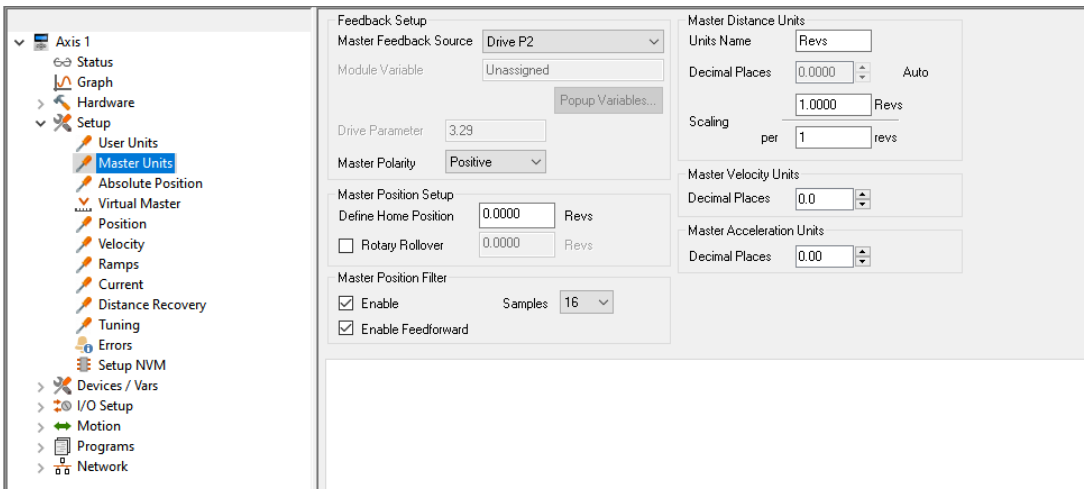


Figure 8-32: Master Units Setup View

Master Feedback Source
Master Feedback Source indicates the source of the master encoder input. Select from Drive P1/P2, Slot1 P1/P2, Slot2 P1/P2, Slot3 P1/P2, Module Parameter, Drive Parameter, and Virtual.

Select Drive Parameter, the Drive Parameter text box will become available to enter the drive parameter that will be the source of the master.

Select Module Parameter, the Module Variable text box will become available to enter a parameter. The module variable can be entered in one of two ways: type the parameter into the text box using the program format for the variable, or click the Popup

Variables button and the Select Variables from Tree window will open. Select the variable and drag it over to the Module Variable text box.

When Virtual is selected as the Master Feedback Source, the parameters on the Virtual Master view in the application will need to be setup to generate VirtualMaster.Counts.

Drive Parameter
When Drive Parameter is selected as the Master Feedback Source, this text box becomes available to enter the parameter number that will be used as the feedback source.

Master Polarity
This interprets what the encoder will consider positive or negative movement.

Master Position Setup

Define Home Position
Define Home Position is the value that the Master Position Feedback will be set to when the MasterAxis.DefineHome destination is activated. After the MasterAxis.DefineHome has been activated, the MasterAxis.AbsolutePosnValid source will activate.

Rotary Rollover Check Box
When selected, the rotary rollover feature for the Master Axis will be enabled.

Rotary Rollover
If enabled, the Master Position will rollover to zero at the value specified here. As the master encoder counts up, the master position feedback will increase until it reaches the Rotary Rollover value and then reset to zero and continue to count up. If rotating in the negative direction, the master position feedback will decrease until it reaches zero, and then start over at the Rotary Rollover value.

Master Distance Units
The parameters in this group are used to establish the scaling of the master axis into user units.

Units Name
This is a text string up to 12 characters that will be used to define the units of distance traveled by the master axis for incoming synchronization signals.

Decimal Places
The number of decimal places set in this parameter determines the number of digits after the decimal point used in all distance and position parameters used in synchronized motion throughout the software. Using a high number of decimal places will improve position resolution, but will also limit the maximum position. You can select from zero to six decimal places of programming resolution.

Master Distance Units Scaling
The scaling factor is defined as following:

$$\text{Scaling} = \frac{\text{MasterDistUnits.CharacteristicDistance}}{\text{MasterDistUnits.MasterRev OR MasterDistUnits.ModuleVarScalingDenominator}}$$

The numerator (top value of the scaling fraction) is the Characteristic Distance (MasterDistUnits.CharacteristicDistance).
The Characteristic Distance is the number of Master Distance Units that will be traveled per number of units defined in the bottom value of the fraction (denominator).

When the Master Feedback Source is set to one of the drive's or slot's P1/P2 ports then the denominator (MasterDistUnits.MasterRev) is the # of encoder revolutions. The master revs parameter is the number of incoming whole revolutions it takes to travel the specified characteristic distance.

When the Master Feedback Source is set to Drive Parameter or Module Parameter the denominator (MasterDistUnits.ModuleVarScalingDenominator) value has units based on the parameter chosen. If Virtual is chosen the denominator (MasterDistUnits.ModuleVarScalingDenominator) units will be virtual counts. The denominator is the number of incoming whole revolutions it takes to travel the specified characteristic distance.

These two scaling parameters can be modified within a user program. This allows users to create a series of user programs to automatically change the scaling factor to handle multiple products on the same machine, without requiring the user to download a new configuration. Extreme caution should be used to prevent the change of scaling when synchronized motion is active. The motor should be stopped and the drive disabled when changing these scaling factors in a program.

Changing the scaling within a user program or from an HMI will automatically clear the AbsolutePosnValid signal, and for Absolute encoders the AbsoluteHomeDefined signal. This means that when the values are changed, the machine must be homed again.

Master Position Filter

Master Position Filter Enable Check Box
The Master Position Filter Enable check box is used to turn on or turn off the Master Position Filter. When the check box is selected, the filter is active and the user must select the number of samples, from the Samples list box, used by the filter. If the check box is clear, the filter is not used

Master Position Filter Samples

Defines the number of samples used by the filter to smooth the master signal. Increasing the number of samples increases smoothness, but also increases lag. See Filter table below to select proper setting.

		Feedforward OFF	Feedforward ON
# of Samples	Disabled	One update of phase shift (not velocity dependent) No Filtering	No delay, No Filtering
	4	Small Lag (function of speed), Low Filtering	Poor at low speed, Low Filtering
	8	Medium Lag (function of speed), Medium Filtering	Good at low speed, Medium Filtering
	16	Large Lag (function of speed), High Filtering	Best at low speeds, High Filtering

Smoother

Increasing Lag with FF Off

Reduced Lag

Enable Feedforward

The Enable Feedforward check box is used to turn on or turn off feedforward. When the check box is selected, feedforward is active. If the check box is clear, feedforward is not used.

Master Velocity Units

Decimal Places

The number of decimal places defined in this parameter determines the maximum resolution of all synchronized time base velocity parameters found throughout the PowerTools Studio software. Set between 0 and 6 decimal places. Higher number of decimal places allows higher velocity resolution, but can limit the maximum speed allowed by the application.

Master Acceleration Units

Decimal Placed

The number of decimal places defined in this parameter determines the maximum resolution of all synchronized time base acceleration and deceleration parameters found throughout the software. Set between 0 and 6 decimal places.

8.6.4 Absolute Position View

NOTE

If not using an absolute encoder type in your application, this view can be skipped altogether.

The primary reason for using an absolute encoder is that position is not lost when power to the machine is cycled. The absolute encoder does not require that you maintain logic power, nor do most absolute encoders require any type of battery power backup. Therefore, on power-up, the motor encoder can provide its position feedback to the motion controller without the need to "re-home" the machine to a sensor or encoder marker pulse. Since the machine does not need to be re-homed, the customer saves time and in many cases reduces product waste.

This view is used to define how the position feedback from the absolute encoder is to be interpreted by the PTi210 module on power-up or after a warm-start.

Figure 8-33: Absolute Position View

Motor Axis and Master Axis Tabs

The Absolute Position view allows the user to configure the necessary parameters for both the Motor Axis (this is the axis being controlled by the PTi210 module), and the Master Axis (if applicable). If the encoder selected for use on either the Motor Axis or Master Axis are not of the absolute type, then the setup parameters on that tab will be grayed-out and cannot be modified.
The following descriptions of the Absolute Position Setup parameters are duplicated for both the Motor Axis and Master Axis.

Absolute Position Setup

Absolute Position Auto-Calculate Enable

This check box is used to enable or disable the automatic calculation of the position for PTi210 module, based on a defined absolute home position, on power-up.

If this check box is selected, the PTi210 module will only read the position from the absolute feedback device on power-up, and then set the position feedback equal to that value. However, the PTi210 module will NOT take into account any previously defined home position. If clear, the Absolute Position Mode parameters will remain unavailable.

If the check box is selected, the PTi210 module will internally calculate the correct feedback position of the machine in user units based on a previously defined home position and the position feedback from the absolute feedback device. The previous sentence is valid only if the system has been previously homed (either using a homing routine or the DefineHome function). The Application Type radio buttons are then used to determine the method used for calculating the position feedback on power-up or after a warm-start (see below).

Absolute Position Mode

One-Sided Mode

One-Sided Mode implies that once the absolute home position has been defined, the user wishes to utilize the full multi-turn resolution of the encoder in ONE DIRECTION from the home point. Using the example of a 12-bit multi-turn absolute encoder, the motor could then travel 4096 revs in the positive direction without experiencing any problems due to absolute rollover.

NOTE

In One-Sided mode, if the motor moves in the negative direction from the home position, the PTi210 module cannot detect this condition, and therefore the absolute position would be incorrectly calculated on the next power up.

EXAMPLE: The user homes the motor to a sensor. When the sensor activates, the motor is at the machine home position PM = 0 revs (where PM is the Position feedback of the Machine). When the motor is at the home position of 0 revs, the absolute encoder reads PA = 1375 revs (where PA is the Position feedback of the Absolute Encoder). In this example, 1375 is just a randomly chosen number for demonstration purposes. Therefore, the PTi210 module now uses this relationship of PM and PA at the home position to calculate the machine position (PM) on power up.

Position Feedback of Machine when at Home = PMH = 0

Position Feedback of Absolute Encoder when at Home = PAH = 1375

If the motor then moves 1000 revs in the positive direction, PM will be PMH + 1000 or PM = 1000, and PA is PAH + 1000 or PA = 2375. If the machine is powered-down and then back up, the PTi210 module would read the position from the absolute encoder PA, subtract the position of the absolute encoder at the home position PAH, and then add the position feedback of the machine at the home position PMH, to get PM.

Calculation #1: Used when $P_A > P_{AH}$

$P_M = P_A - P_{AH} + P_{MH}$

$= 2375 - 1375 + 0$

$= 1000$ revs

If the motor then moves an additional 2000 revs in the positive direction, P_M will be P_{MH} + 3000 (because we had already moved 1000) or $P_M = 3000$. P_A would be P_{AH} + 3000 or $P_A = 4375$. However, we know that since this is a 12-bit multi-turn absolute encoder, the absolute encoder rev counter will reach 4096 and rollover to 0. Therefore, we will never get a value of 4375 on the absolute encoder. In this case, the absolute encoder would read P_A = 4375 - rollover position = 4375 - 4096 = 279 revs. Since the user has defined the application type to use One-Sided Mode, the PTi210 module can interpret this position from the absolute encoder to properly calculate the correct machine position. If the machine was powered-down, and then back up, the P_M would be calculated as follows:

Calculation #2: Used when $P_A < P_{AH}$

P_M = (Maximum Encoder Multi-Turn Resolution - P_{AH}) + P_A + P_{MH}

$= (4096 - 1375) + 279 + 0$

$= 3000$ revs

3000 revs is exactly the result that we wanted! Notice that even though P_A rolled over when it reached 4096, we can still calculate the correct position since the user has defined that the application is running in One-Sided Mode.

If the motor moves an additional 2000 revs in the positive direction, P_M will be P_{MH} + 5000 (because we already moved 3000 before) or P_M = 5000. P_A would be P_{AH} + 5000 or P_A = 6375. Again, since the 12-bit encoder cannot exceed 4096, we need to recalculate P_A as P_A = 6375 - 4096 = 2279. Notice that when P_A = 2279, we should use Calculation #1 above since 2279 > 1375. Therefore, our P_M is as follows:

P_M = P_A - P_{AH} + P_{MH}
= 2279 - 1375 + 0
= 904 revs

Note that this value is NOT correct. This is because after 5000 revs, we have exceeded the absolute resolution of the encoder in a single direction.

So you can see that when using One-Sided Mode, as long as the motor does not move more than 4096 revolutions in the positive direction from the home position, the PTi210 module can correctly calculate the absolute machine position on power up.

Two-Sided Mode

Two-Sided Mode implies that once the absolute home position has been defined, the user wishes to distribute the full multi-turn resolution of the encoder evenly in BOTH DIRECTIONS from the home point. Using the example of a 12-bit multi-turn absolute encoder, the motor could then travel 2048 revs in either the positive or negative direction without experiencing any problems due to absolute rollover. IMPORTANT NOTE: In Two-Sided mode, if the motor moves in excess of 2048 revs in either direction from the home position, the PTi210 module cannot detect this condition, and therefore the absolute position would be incorrectly calculated on the next power up.

Example:

The user homes the motor to a sensor. When the sensor activates, the motor is at the machine home position P_M = 0 revs (where P_M is the Position feedback of the Machine). When the motor is at the home position of 0 revs, the absolute encoder reads P_A = 1375 revs (where P_A is the Position feedback of the Absolute Encoder). In this example, 1375 is just a randomly chosen number for demonstration purposes. Therefore, the PTi210 module now uses this relationship of P_M and P_A at the home position to calculate the machine position (P_M) on power up.

Position Feedback of Machine when at Home = P_{MH} = 0

Position Feedback of Absolute Encoder when at Home = P_{AH} = 1375

We then need to calculate the value equal to half the resolution of the encoder (E_{half}), and the position that distributes the encoder resolution in half. This is the position feedback of the absolute encoder at the rollover point (or P_{AR}).

E_{half} = Max Resolution / 2
= 4096 / 2
= 2048

P_{AR} = P_{AH} + E_{half}
= 1375 + 2048
= 3423

P_{AR} must be within the max resolution of the encoder, so if P_{AR} calculated above is > Max Encoder Resolution we need to subtract the max resolution:

P_{AR} = P_{AR} - Max Resolution

If the motor then moves 1000 revs in the positive direction, P_M will be P_{MH} + 1000 or P_M = 1000,and P_A is P_{AH} + 1000 or P_A = 2375. If the machine is powered-down and then back up, the PTi210 module would read the position from the absolute encoder P_A, subtract the position of the absolute encoder at the home position P_{AH}, and then add the position feedback of the machine at the home position P_{MH}, to get P_M.

Calculation #1: Used when P_{AR} > E_{half} AND P_A < P_{AR}

P_M = P_A - P_{AH} + P_{MH}
= 2375 - 1375 + 0
= 1000 revs

If the motor is again at the home position and then moves 1000 revs in the negative direction, P_M will be P_{MH} - 1000 or P_M = -1000. If we power down, and then back up, P_A would be P_{AH} - 1000 or P_A = 375. Once again, the conditions for Calculation #1 are met, so we will use it again to find P_M.

P_M = P_A - P_{AH} + P_{MH}
= 375 - 1375 + 0
= -1000 revs

If we now start from the home position again, and move the motor 2000 revs in the negative direction, P_M will be P_{MH} - 2000 or P_M = -2000. If we power down and then back up, P_A should be P_{AH} - 2000, or P_A = -625. However, we know that since this is a 12-bit multi-turn absolute encoder, when traveling in the negative direction, the absolute encoder rev counter will reach 0 and rollover to 4096. Therefore, we will never get a value of -625 on the absolute encoder. In this case, the absolute encoder would read P_A = (P_{AH} - 2000) + Max Encoder Resolution = 3471. Since the user has defined the application type to utilize Two-Sided Mode, the PTi210 module can

interpret this position from the absolute encoder to calculate the correct machine position. If the machine was powered-down, and then back up, the P_M would be calculated as follows:

Calculation #2: Used when $P_{AR} > E_{half}$ AND $P_A > P_{AR}$

P_M = P_A - P_{AH} - Max Enc Resolution + P_{MH}

= 3471 - 1375 - 4096 + 0

= -2000 revs

-2000 revs is exactly the result that we wanted! Notice that even though P_A rolled over when it reached 0, we can still calculate the correct position since we know that the application is running in Two-Sided Mode.

If the motor moves an additional 200 revs in the negative direction, P_M will be P_{MH} - 2200 (because we already moved -2000 before) or P_M = -2200. P_A would be P_{AH} - 2200 + Max Encoder Resolution or P_A = 3271. Notice that with P_A = 3271 we no longer match the conditions of Calculation #2, but now instead match those of Calculation #1. Therefore, P_M is calculated as follows:

P_M = P_A - P_{AH} + P_{MH}

= 3271 - 1375 + 0

= 1896 revs

Note that this value is NOT correct. The desired result of the calculations is -2200. This is because after 2048 revs in either given direction from the home position, we have exceeded half of the absolute resolution of the encoder in that direction.

We could test the same scenario in the positive direction. If the motor starts from the original home position and moves 2200 revs in the positive direction. P_M at that point would be P_{MH} + 2200 or P_M = 2200. If we power the system down and then back up, we find P_A = P_{AH} + 2200 = 3575. In this case, we match the conditions for Calculation #2 and therefore use it to calculate P_M.

P_M = P_A - P_{AH} - Max Enc Resolution + P_{MH}

= 3575 - 1375 - 4096 + 0

= -1896 revs

Again, -1896 is NOT the correct position since we expected to get 2200 revs instead.

So you can see that when using Two-Sided Mode, as long as the motor does not move more than 2048 revolutions in either direction from the home position, the PTi210 module can correctly calculate the absolute machine position on power up.

It should be noted that there is a different set of calculations used to find P_M when P_{AR} is < E_{half}. These calculations are as follows:

Calculation #3: Used when $P_{AR} < E_{half}$ AND $P_A < P_{AR}$

P_M = Max Enc Resolution - P_{AH} + P_A + P_{MH}

and

Calculation #4: Used when $P_{AR} < E_{half}$ AND $P_A > P_{AR}$

P_M = P_A - P_{AH} + P_{MH}

8.6.5 Reasons for Re-Homing

Once an absolute home routine has been executed, either using a Home motion profile or a DefineHome, the system can automatically calculate its position after a download or power-cycle without the need to re-home the machine. Under normal circumstances, the system should never need to be homed again.

However, it is important to note that there are a few reasons or actions that could require the system to be re-homed. These are as follows:

1. Flash Upgrade

If the PTi210 module is flash upgraded, the contents of NVM is lost and hence, the system needs to be re-homed to re-learn the absolute home position of the machine. The absolute home position is not stored as part of the user configuration, so downloading the previous user configuration will not re-load the absolute home position.

2. UndefineHome or MasterAxis.UndefineHome

The UndefineHome function (for the motor or follower axis) and the MasterAxis.UndefineHome function (for the master axis) are used to tell the system to no longer use the previously defined Absolute Home Position. Additionally, the AbsolutePosnValid and AbsoluteHomeDefined signals will deactivate when the UndefineHome is used. The UndefineHome and MasterAxis.UndefineHome functions by themselves do not cause a new absolute home routine to be performed. The user must initiate a new Home motion profile or use the DefineHome to re-home the system. When the UndefineHome or MasterAxis.UndefineHome functions are activated, they do not set the previously stored values of for the absolute home position to zero. The previously stored values will remain in the absolute home position registers until another absolute home routine is performed.

3. Exceeding the range of the absolute encoder

Absolute Encoders fall into two categories called Single-Turn Absolute and Multi-Turn Absolute.

Single-Turn Absolute encoders are only absolute within a single revolution of the motor. Once the motor turns more than a single revolution, the absolute position provided on power-up could now be off by one or more revolutions. For example:

A Single-Turn Absolute encoder reads a value of 0.25 revolutions on power up. If the device moves 0.5 revolutions in the positive direction, the position feedback should read 0.75 revolutions. If the system is powered down and back up, the encoder would still read 0.75 revolutions, just as it should. Now, if the system is powered-down, then moved one full revolution in the positive direction, the position feedback should be 1.75 revolutions. However, since the device is only a single-turn absolute device, the encoder would read 0.75 revolutions on power up. Therefore, the encoder cannot differentiate between 0.75, 1.75, 2.75, and so on. The encoder is only absolute within a single revolution.

Multi-Turn Absolute encoders are absolute for more than a single revolution, and up to some maximum number of revolutions defined by the design of the encoder. For example a multi-turn encoder contains 12 bits of encoder turns information which equates to 2^12 revolutions or 4096 revolutions absolute. This means the absolute encoder position can range from 0 to 4096 revolutions and still know exactly where it is through a power cycle. This is sufficient for most applications, however multi-turn devices suffer the same fate as the single-turn absolute encoder if the encoder moves past the absolute range of the multi-turn device (4096 revs in this case). For example:

A 12-bit Multi-Turn Encoder reads a value of 5.35 revs on power up. If the device moves 10 revs in the positive direction, the position feedback reads 15.35 revs. If the system is powered down and back up, the encoder would read 15.35 revs since it is a multi-turn absolute encoder. If the system is powered-down again, and this time is moved 4096 revs in the positive direction (causing us to exceed the absolute resolution of the encoder), the desired position feedback would be 4096 revs + 15.35 revs = 4111.35 revs. However, since the encoder in question only contains 12-bits (or 4096) revs of absolute turns information, the feedback position reaches 4096 and starts over at zero. So the position feedback on power-up would read 4111.35 revs - 4096 revs = 15.35 revs. Therefore, the encoder cannot differentiate between the 15.35 revolutions position that it was at after the first short move, or the 15.35 position that we are at after moving 4096 additional revs.

So, regardless of the type of encoder (Single-Turn or Multi-Turn), if the motor moves outside the range of the absolute encoder, the system must be homed again to establish the desired absolute home position of the machine.

NOTE

It is the user's responsibility to design the machine such that the encoder does not move outside of its supported absolute range.

4. Disabling of Absolute Position Auto-Calculate Enable

If the user un-checks the Absolute Position Auto-Calculate Enable check box after the system has been homed, and then downloads the configuration, the AbsoluteHomeDefined signal will be automatically reset. The previously stored home position registers will not be cleared. Note that simply re-selecting the Absolute Position Auto-Calculate Enable check box and downloading will not allow you to continue using the previously defined home position information. In order to again use the automatic position calculation it is necessary to re-home the system.

5. Motor/Encoder Replacement

If the motor/encoder is repaired or replaced, it is necessary for the user to undefine the absolute home and then repeat the homing procedure. This can be done using an actual Home motion profile, or using the UndefineHome/DefineHome functions. This is necessary because the new or repaired motor/encoder almost certainly will not be mounted with the exact same absolute position when at the machine home position. There is no way for the PTi210 module to monitor that this scenario has occurred, so it is up to the user to manually undefine the home, and then repeat the homing procedure.

6. Change in User Unit Scaling

If the user changes the User Unit Scaling for either the motor axis or the master axis, the respective AbsHomeDefined signal will automatically be cleared. This must be done since the relationship between the stored values of the Absolute Home Rev Count/Posn and Absolute Home Position in User Units is no longer valid.

8.6.6 Virtual Master View

The Virtual Master View is used to create a simulated encoder output. It generates an encoder stream of counts without the actual operation of a motor. This count can be used by the drive itself as an input to the Master Source (MasterAxis.Source). It can also be transmitted to other drives through the Sync Encoder Output connector and into their Master Sync Input connector.

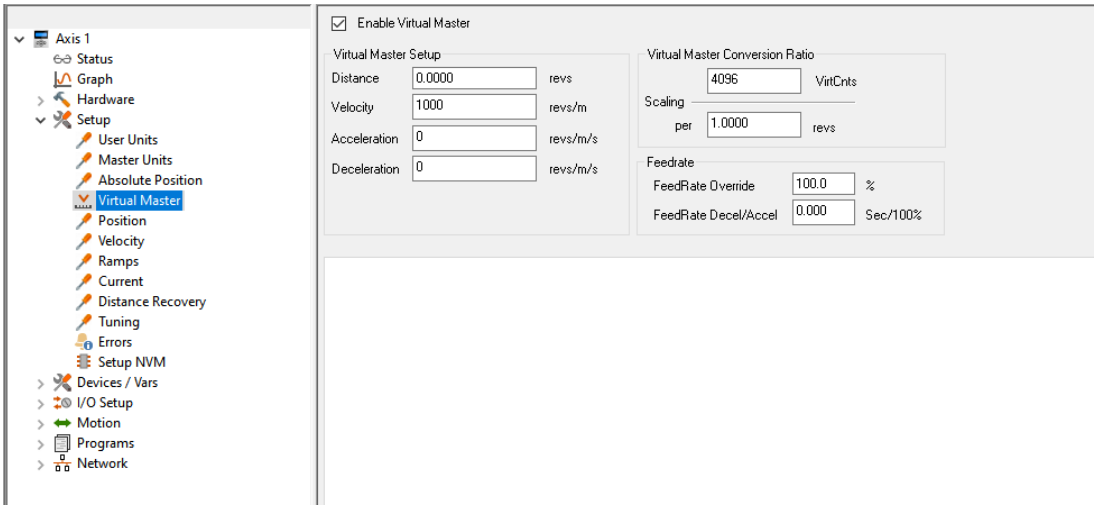


Figure 8-34: Virtual Master View

Enable Virtual Master Check Box

Enable Virtual Master check box (VirtualMaster.VirtualMasterEnable) by default is clear. Select the check box to enable the virtual master feature.

Virtual Master Setup Group

Distance
Distance (VirtualMaster.Dist) is the incremental distance virtual master will move, in user units, if the virtual master is initiated as an index.

Velocity
Velocity (VirtualMaster.Vel) is the maximum virtual velocity that will be attained by the virtual master. This parameter is in user units.

Acceleration
Acceleration (VirtualMaster.Accel) is the acceleration rate, in user units, that the virtual master will use to accelerate. This parameter is used when in either jog or indexing mode.

Deceleration
Deceleration (VirtualMaster.Decel) is the deceleration rate, in user units, that the virtual master will use to decelerate in either jog or index mode.

Virtual Master Conversion Ratio Group

Scaling
Converts the user units distance into virtual counts.

VirtCnts
The numerator (top value of the scaling fraction) is the VirtualMaster.CharacteristicLength. The characteristic length is the number of virtual counts that will be generated per the distance, in user units, defined by the denominator (bottom number of the scaling fraction).

Distance (UserUnits)
The denominator (bottom value of the scaling fraction) is VirtualMaster.CharateristicDistance, in user units, and is used with VirtualMaster.CharacteristicLength to create the virtual master conversion ratio.

If the user sets the numerator to 10,000 and the denominator to 1 revs, then 10,000 virtual counts will be sent out when the virtual master produces 1 rev of virtual motion.

Feedrate Group

FeedRate Override
This parameter (VirtualMaster.FeedRateOverride) is used to scale the Virtual Master counts. It can be described as “scaling in real time”. The default setting of 100 % will allow all counts to occur in real time. A seeing of 50 % will scale time so that all counts are half as fast as they are at 100 %. A setting of 200 % will scale time so that all count run twice as fast as they would at 100 %. Feed Rate Override is always active, and this parameter may be modified via Modbus, Ethernet, or in a program.

FeedRate Decel/Accel
FeedRate Decel/Accel (VirtualMaster.FeedRateDecelerationTime) specifies the ramp used when velocity changes due to a change in the FeedRate Override value. The units of FeedRate Decel/Accel are seconds/100 %. Therefore, the user must specify the amount of time (in seconds) to accelerate or decelerate 100 % of programmed feedrate.

8.6.7 Position View
The Position view allows the user to configure parameters related to position control of the PTi210 module. Figure 8-35 shows a sample of the Position view.

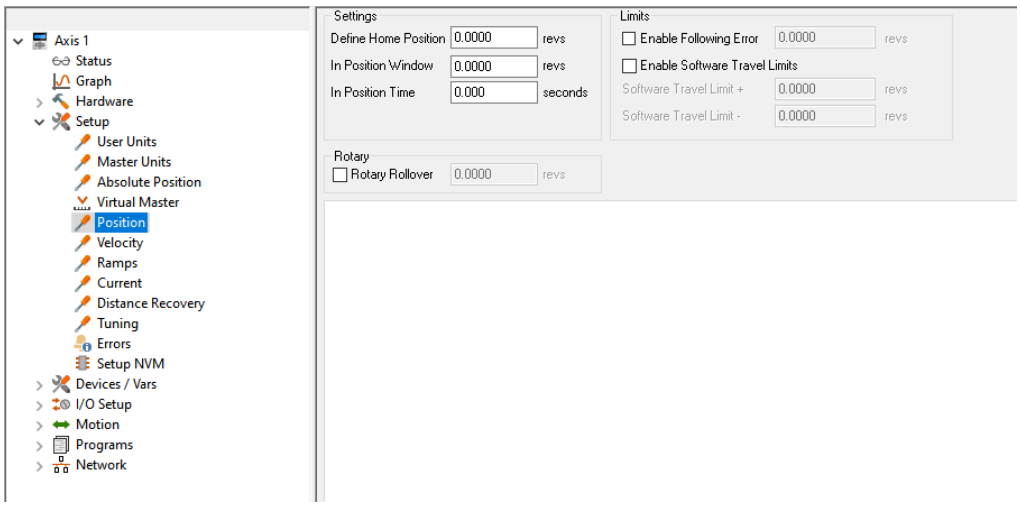


Figure 8-35: Position View

Settings

Define Home Position

This is the value to which the position command will be set when the Define Home destination is activated. This is used in applications which do not use a home routine, but require a known reference point. The units are defined on the User Units view.

In Position Window

The absolute value of the Following Error must be less than or equal to this value at the end of an index in order for the InPosn source to activate. This window is set in units specified in the User Units view.

Example:

The In Position window is set to 0.0025 revs. At the end of an index, the following error is calculated to be 0.0012 revolutions. Therefore, the InPosn source will activate.

In Position window is set to 0.001 inches. If at the end of an index, the following error is calculated to be 0.0015 inches, then the InPosn source will not activate.

In Position Time

This is the amount of time in seconds that commanded motion must be complete and the absolute value of the following error must be less than the In Position Window for the InPosn source to activate. If set to zero (default), then InPosn will activate as soon as motion stops and the following error is less than the In Position Window parameter.

Limits

Enable Following Error

Select this check box to enable (or disable if clear) the Following Error Limit. If enabled, a fault will be generated if the absolute value of the following error ever exceeds the value in the following error parameter. If disabled, a fault will never be generated.

Following Error Limit

Following Error is the difference between the Position Command and the Position Feedback. It is positive when the Position Command is greater than the Position Feedback. If the absolute value of the following error exceeds the value you enter here, the drive will generate a Following Error Fault. All accumulated Following Error will be cleared when the drive is disabled.

The Following Error Limit is defined in user units.

Enable Software Travel Limits

Select this check box to enable (or disable if clear) the software travel limits. If clear, the software travel limits are not monitored.

Software Travel Limit Plus

If the absolute position is greater than or equal to this value the Software Travel Limit Plus Active source shall activate.

A rising edge occurs when the absolute position is greater than or equal to the parameter Software Travel Limit +. A falling edge will be generated as soon as the above is not true.

Software Travel Limit Minus

If the absolute position is less than or equal to this value the Software Travel Limit Minus Activate shall activate.

A rising edge occurs when the absolute position is less than or equal to the parameter Software Travel Limit -. A falling edge will be generated as soon as the above is not true.

Rotary

Rotary Rollover

Select this check box to enable (or disable if clear) the rotary rollover feature.

Rotary Rollover Position

This parameter is used in rotary applications and determines the position at which the internal position counter will be reset to zero.

Example:

The user has a rotary table application with distance user units of degrees, 360.00 degrees/1 rev. The Rotary Rollover would be set to a value of 360°.

The motor is traveling in the positive direction. As the feedback position reaches 359.999 and continues on, the feedback position will reset (or roll-over) to zero. If the motor changes direction and travels in the negative direction, the position will rollover at 0 to 359.999 degrees and count down. The resolution of the rotary rollover point is determined by the Distance Units Decimal Places parameter on the User Units view in the PowerTools Studio software.

If an absolute index is used with a non-zero rotary rollover point, the PTi210 module will calculate the shortest path to its destination and move in the required direction.

To force the motor to run a certain direction, use the Rotary Plus or Rotary Minus type of indexes.

8.6.8 Velocity View

The Velocity view allows the user to define parameters related to the velocity control of the PTi210 module. Figure 8-36 shows an example of the Velocity view.

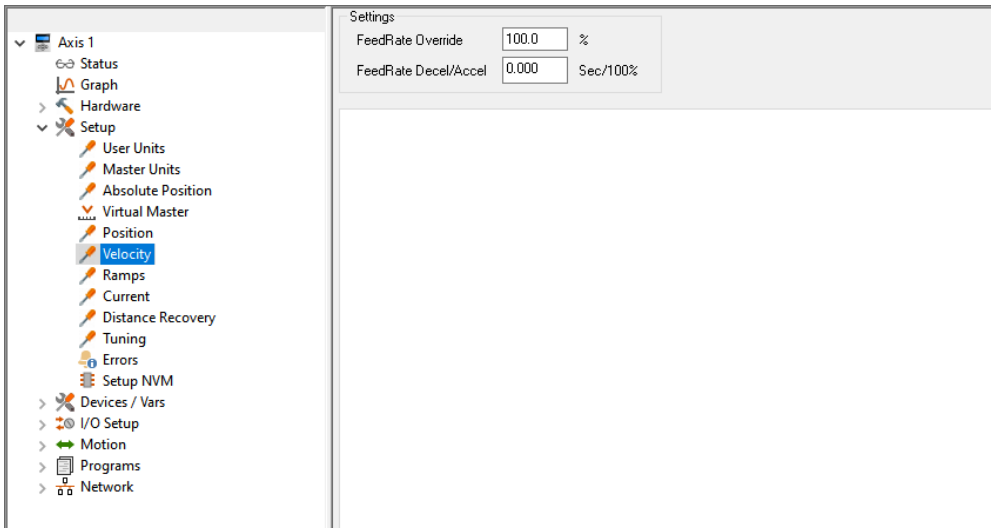


Figure 8-36: Velocity View

FeedRate Override

This parameter is used to scale all motion. It can be described as scaling in real time. The default setting of 100 % will allow all motion to occur in real time. A setting of 50 % will scale time so that all moves run half as fast as they do at 100 %. A setting of 200 % will scale time so that all moves run twice as fast as they would at 100 %. FeedRate Override is always active and affects all motion, including accels, decels, dwells, and synchronized motion. This parameter may be modified via the communication interface or in a program.

FeedRate Accel/Decel

The FeedRate Decel/Accel parameter specifies the ramp used when velocity changes due to a change in the FeedRate Override value. The units of feedrate decel/accel are Seconds/100 % of FeedRate. Therefore, the user must specify the amount of time (in seconds) to accelerate or decelerate 100 % of FeedRate.

8.6.9 Ramps View

The Ramps view allows the user to define various accel/decel ramps used under typical application conditions. Figure 8-37 shows an example of the Ramps view.

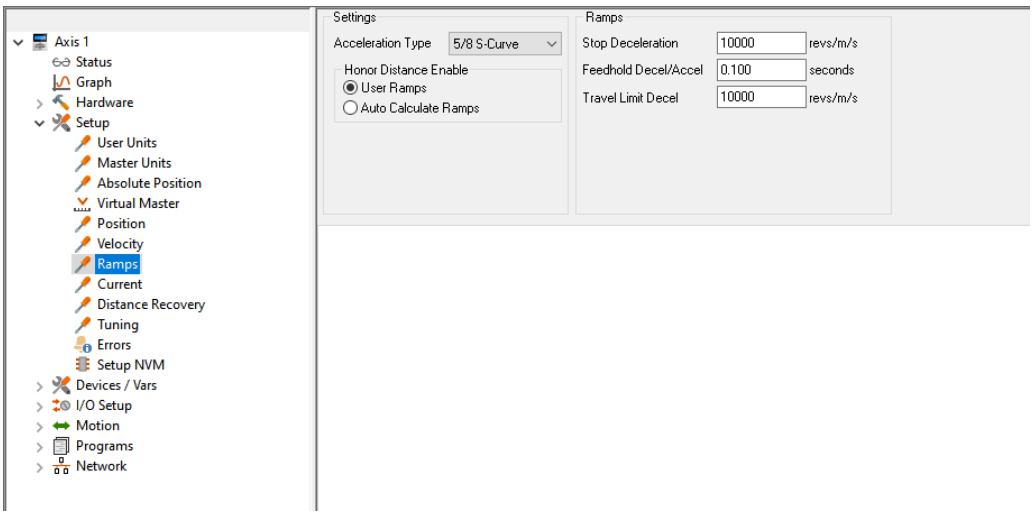


Figure 8-37: Ramps View

Settings

Acceleration Type

Press the arrow by the Acceleration Type list box. It will display the various acceleration types: 5/8 S-Curve, 1/4 S-Curve, Linear, and S-Curve.

This is used to select the acceleration/deceleration type for all motion (homes, jogs and indexes). The “S-Curve” ramps offer the smoothest motion, but lead to higher peak acceleration/deceleration rates. “Linear” ramps have the lowest peak acceleration/ deceleration rates but they are the least smooth ramp type. “5/8 S-Curve” ramps and “1/4 S-Curve” ramps use smoothing at the beginning and end of the ramp but have constant (linear) acceleration rates in the middle of their profiles. The “5/8 S-Curve” is less smooth than the “S-Curve” but smoother than the “1/4 S-Curve”.

S-Curve accelerations are very useful on machines where product slip is a problem. They are also useful when smooth machine operation is critical. Linear ramps are useful in applications where low peak torque is critical. Below is a comparison of the 4 ramp types:

S-Curve:	Peak Acceleration =	2 x Average Acceleration
5/8 S-Curve:	Peak Acceleration =	1.4545 x Average
1/4 S-Curve:	Peak Acceleration =	1.142857 x Average Acceleration
Linear:	Peak Acceleration =	Average Acceleration

Honor Distance Enable

User Ramps/Auto Calculate Ramps

The user has the ability to select one of two ramp control types for the entire motion control system. By default, User Ramps is selected. The user can change the ramp controls in PowerTools Studio and perform a download to make the change, or the parameter AutoCalcRampsEnable can be turned On or Off within a program. To enable User Ramps, AutoCalcRampsEnable should be turned Off, and to enable Auto Ramps, AutoCalcRampsEnable should be turned On. Once a motion profile is in progress, changes to this parameter will be ignored until the next motion is initiated.

See the description of each of the ramp types below.

User Ramps

When User Ramps are enabled, the Acceleration or Deceleration ramp entered by the user will ALWAYS be used during a motion profile, even if that means the motor must overshoot the entered stopping position. Under this circumstance, the acceleration or deceleration ramp would be honored, and therefore the motor may need to reverse directions after coming to a stop in-order to reach the user entered target position. This scenario most often occurs when using Compound or Blended Index instructions within a program. During Compound or Blended indexes, the user occasionally does not enter an aggressive enough acceleration or deceleration ramp to reach the target velocity within the specified distance. See the Figures 8-38 and 8-39 below for examples of how User Ramps work. For more information on Index.#.CompoundInitiate and/or Index.#.BlendInitiate, see the programming section of this User Guide.

Auto Calculate Ramps

When Auto Calculate Ramps is selected the PTi210 module will automatically calculate the necessary ramp to reach the target velocity within the user specified distance without any overshoot. In this scenario, the user entered acceleration or deceleration rate is ignored. See the Figures 8-38 and 8-39 below for examples of how Auto Calculate Ramps work.

	Distance (Revs)	Velocity (RPM)	Accel (RPM/sec)	Decel (RPM/sec)
Index 0	**20**	**1250**	**2000**	**3000**
Index 1	**3**	**500**	**500**	**500**

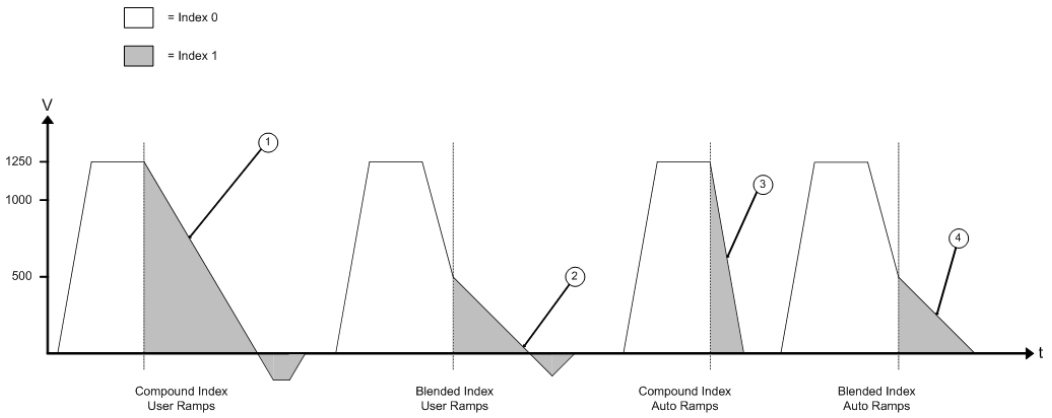


Figure 8-38: Ramps Examples of a Fast Index to a Slower Index

1. Index.1.Accel specified by user is used to decelerate from Index.0.Vel, to Index.1.Vel, but overshoots since ramp is not aggressive enough to reach Index.1.Vel within Index.1.Dist of 3 Revs.

2. Index.1 begins at Index.1.Vel but since Index.1.Decel specified by user is not aggressive enough to decelerate to zero velocity within Index.1.Dist of 3 Revs slight overshoot occurs.

3. Index.1 begins at Index.0.Vel and ramp is automatically calculated to reach zero speed within Index.1.Dist of 3 Revs without any overshoot. If Index.1.Accel and or Decel were aggressive enough to reach zero speed within 3 Revs, they would have been used instead of automatically calculating the ramp.

4. Index.1 begins at Index.1.Vel and ramps is automatically calculated to reach zero speed within Index.1.Dist of 3 Revs without any overshoot. If Index.1.Decel was aggressive enough to reach zero speed within 3 Revs, it would have been used instead of automatically calculating the ramp.

	Distance (Revs)	Velocity (RPM)	Accel (RPM/sec)	Decel (RPM/sec)
Index 0	**5**	**500**	**1000**	**1000**
Index 1	**20**	**1250**	**2000**	**2000**

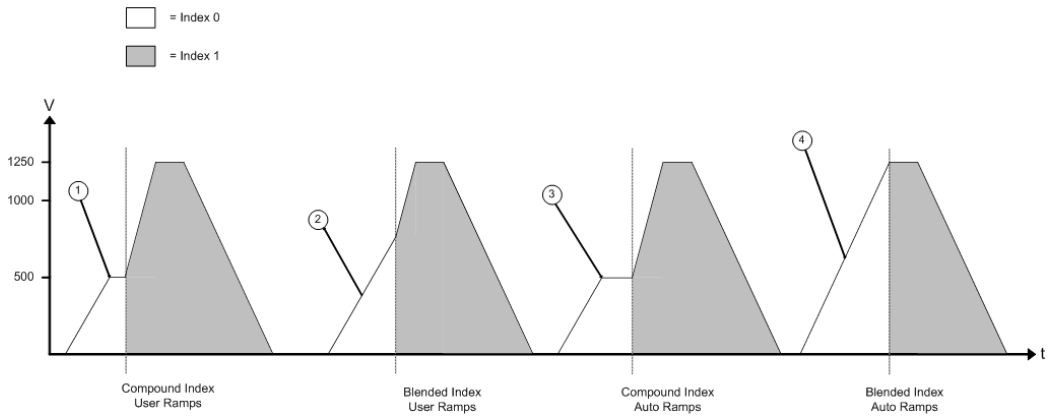


Figure 8-39: Ramps Examples of a Slow Index to a Faster Index

1. Index.0.Accel specified by user is used to accelerate up to Index.0.Vel. Index.0.Accel is aggressive enough to reach Index.0.Vel within Index.0.Dist of 5 Revs. Since indexes are compounded together, Index 1 begins at Index.0.Vel.

2. When indexes are Blended, Index 0 should end at velocity of Index 1, but Index.0.Accel is not aggressive enough to reach Index.1.Vel within Index.0.Dist of 5 Revs. Therefore, entire distance of Index 0 is used to accelerate towards Index.1.Vel.

3. Acts the same as the Compound with User Ramps because Index.0.Accel entered by user is aggressive enough to reach Index.0.Vel of 500 rpm. If Index.0.Accel entered by the user was not aggressive enough to reach 500 rpm within 5 Revs, necessary ramp would be calculated.

4. Acceleration ramp is automatically calculated to reach Index.1.Vel within Index.1.Dist of 5 Revs. If user had entered a ramp aggressive enough to reach Index.1.Vel within 5 Revs, no automatic ramp calculation would be required, and the user entered acceleration rate would be followed.

Ramps

Stop Deceleration

The value you enter here defines the deceleration rate which is used when the Stop destination is activated. The default is 100 rpm/second.

The Stop destination is found in the Ramps Group in the Assignments view.

Feedhold Decel/Accel

When the Feedhold destination is activated, the motor will decelerate to a stop in the time specified by the FeedholdDecelTime parameter. When feedhold is cleared, the motor will accelerate back to speed in the same specified period of time.

Feedhold is a means to halt the motor within a velocity profile and then return to the profile later at the exact same place in the profile. Feedhold does not ramp and does not decelerate in terms of velocity. Instead, it stops by decelerating in terms of time. For example, if the motor is running at 50 revs/second and feedhold is activated with 2 seconds specified in the FeedholdDecelTime parameter, then the motor will actually slow and stop in 2 seconds as measured time (on a time/velocity profile) goes from 100 % to 0 %.

Travel Limit Deceleration

The value entered here is the deceleration ramp that is used when a software or hardware travel limit is hit.

8.6.10 Current View

The Current view allows the user to configure Current Level Flags and Current Limits for use in the application.

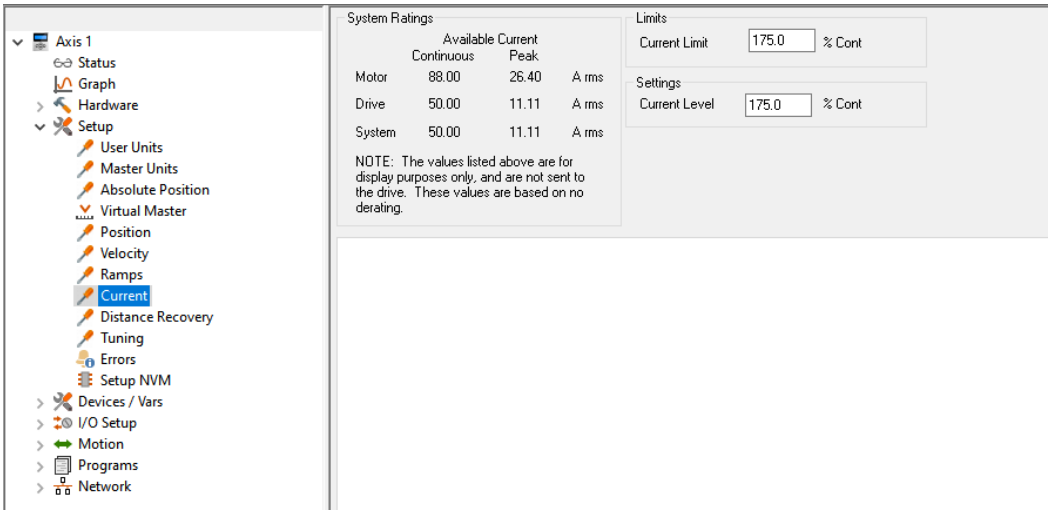


Figure 8-40: Current View

System Ratings

This box is used only to show the user what current will be available from the system based on user-entered information. There are two columns of information based on Continuous and Peak ratings for the different system components and for the system as a whole. The first row shows the Continuous and Peak current rating for the selected Motor. The next row then shows the ratings for the Drive itself, which may be higher or lower than those of the motor. The last row called “System” shows the Continuous and Peak current ratings for the motor/drive combination. The system continuous and peak ratings are the lower of the two motor and drive ratings. The Current limit on this view is entered in units of % Cont. This means that the limit entered is a percentage of the System Continuous Current rating. Therefore, if the system continuous current is equal to 15 Amps, and the Current Limit is set to 145 %, the current will actually be limited to 15 x 1.45 = 21.75 Amps (remember the Current Limit Enable must be active to limit the current).

If the Motor current rating is lower than the Drive current rating, then the system is said to be “Motor Limited”. If the Drive current rating is lower than the Motor current rating, then the system is called “Drive Limited”.

Limits

Current Limit

This parameter sets the value to which the Current Command will be limited when the Current Limit Enable destination is active. To make the Current Limit always active, assign the Current Limit Enable destination to the Initially Active source on the Assignments view.

Settings

Current Level

This parameter sets the activation point for the Current Level Active source. If set to 100 %, the Current Level Active source will activate any time the Current Command reaches or exceeds 100 % continuous.

8.6.11 Distance Recovery View

When initiating gear or cam using the Master Axis to provide synchronization, distance is lost as we must follow the acceleration limitations. This distance can be recovered when at velocity by adding a distance recovery index on top of the gear operation or cam. This view defines the limits of that recovery index.

Gearing uses distance recovery after accelerating from zero to a locked state. Camming uses distance recovery after using the Resume process.

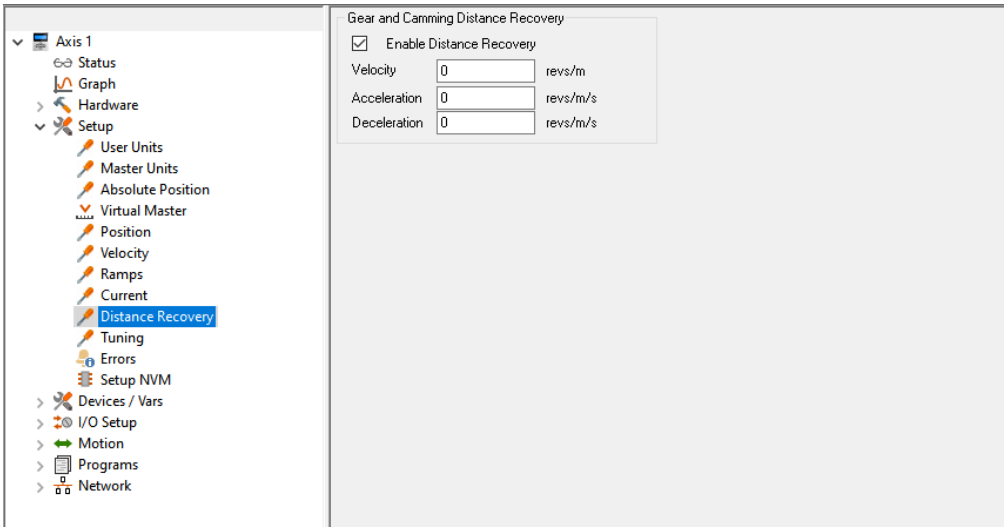


Figure 8-41: Distance Recovery View

Gear and Camming Distance Recovery Group

Enable Distance Recovery Check Box

The Enable Distance Recovery Check box (DistanceRecovery.DistanceRecoveryEnable) is clear (disabled) by default. Select the check box to enable the additive distance recovery index feature.

Velocity

This parameter (DistanceRecovery. Vel) is the velocity limit of the distance recovery index in user units.

Acceleration

Acceleration (DistanceRecovery. Accel) is acceleration rate for the distance recovery index in user units.

Deceleration

This (DistanceRecovery. Decel) is the deceleration rate for the distance recovery index in user units.

8.6.12 Tuning View

The Tuning view contains all the parameters that are adjusted to properly tune the motor/drive system. Figure 8-42 shows an example of the Tuning View.

Figure 8-42: Tuning View

Load

Inertia Ratio

This parameter is the ratio between the load inertia and the motor rotor inertia. The formula is defined as follows:

$$\text{InertiaRatio} = \frac{\text{LoadInertia}}{\text{RotorInertia}}$$

In order to calculate the Inertia Ratio, the user must first calculate the load inertia. The load inertia is the inertia of the load reflected to the motor shaft.

When entering the inertia ratio, if the exact ratio is not known, it is better to enter a lower (more conservative) value than to enter a “higher-than-actual” value. Higher numbers in the Inertia Ratio amplify the gains of the Velocity and Position Loop parameters described below.

The motor rotor inertia used in this calculation can be found on the Motor Tab on the Motor/Encoder view in PowerTools Studio or in the specified motor .ddf file that is installed with PowerTools Studio software.

Tuning

Velocity Loop Bandwidth

The Velocity Loop Bandwidth is the theoretical bandwidth of the velocity controller. It is important for the motor data file to be accurate for ideal velocity performance, in particular the Motor Inertia and Motor Current Constant (Ke). Higher Velocity Loop Bandwidth values will result in better response to change in velocity command, however setting the bandwidth too high can result in ringing (audible ringing) in the velocity loop. The units for Velocity Loop Bandwidth are Hz.

Position Loop Response

The Position Loop Response directly impacts the stiffness of the position loop. Position Loop Response is effectively the Proportional gain term for the position loop controller. Higher Position Loop Response values will result in less following error throughout a motion profile, however setting the response too high can result in instability in the position loop.

Enable Velocity Feedforward

The Velocity Feedforward applies the calculated velocity command directly to the drive’s velocity loop. Enabling the Velocity Feedforward will generally yield faster velocity response, however the feedforward will introduce some overshoot. It is vital in applications that require the use of Jog profiles to enable the Velocity Feedforward signal.

Filter

Current Demand Filter

The Current Demand Filter is a first-order low pass filter applied to the current command of the drive. The parameter entered by the user is a time constant, t, in the filter formula. The frequency of the filter, f, can be derived as follows:

$$f = \frac{1}{(2\pi t)}$$

The units for the parameter are milliseconds. This parameter is written directly to parameter Pr **04.012** of the drive.

Feedback Filter

The Feedback Filter checkbox is used to apply a filter to the feedback source, possible values are Disabled, 1 ms, 2 ms, 4 ms, 8 ms, or 16 ms.

When the velocity loop gains are high, without a feedback filter, it is possible for the velocity loop output to change constantly from one current limit to the other and lock the integral term of the speed controller. The units for the filter are milliseconds. The higher the selected value, the more filtering is applied. When this filter is used, it may be necessary to lower the Current Demand Filter proportionally to this filter.

Module Gains Enable Radio Button

Under normal operation, the PTi210 module calculates Current Loop and Velocity Loop gains based on information provided by the user on the Motor tab and Tuning view. These calculated gains are sent by the PTi210 module to the drive on every power-up or warmstart, overwriting any gain values stored in the drive. In some cases, a user may decide to use their own values for Current Loop and Velocity Loop gains rather than the values calculated by the PTi210 module. Therefore, it is necessary to be able to prevent the PTi210 module from sending its calculated gain values to the drive.

By default, the Module Gains Enable radio button will be selected so that the values calculated by the PTi210 module are sent to the drive. To disable this function, and allow the user to enter specific values for the Current and Velocity loop gains, select the User Gains Enable radio button, this action will enable the Current and Velocity Loop gains allowing the user to modify them as necessary.

If the user decides to enter their own gain values, the values are entered using the Unidrive M keypad (Unidrive M) or the remote keypad (Unidrive M/Digitax HD) (or Drive Menu Watch Window in Power Tools Studio). Once the values are entered, the values must be saved so that they are not lost the next time power is cycled. This is done by navigating to parameter number zero in any menu (Pr **mm.000**), entering the value 1001, and then pressing the red Reset button on the front of the keypad. This will save all drive parameters to NVM.

The gain parameters in the drive written to by the PTi210 module are as follows:

Velocity Loop (Speed Controller) Gains

Kp (Velocity Loop Proportional Gain) – Pr **03.010**

Ki (Velocity Loop Integral Gain) – Pr **03.011**

Current Loop (Current Controller) Gains

Kp (Current Loop Proportional Gain) – Pr **04.013**

Ki (Current Loop Integral Gain) – Pr **04.014**

For more information on the gain parameters, please refer to the relevant drive *User Guide*.

The Module Gains Enable and User Gains Enable radio buttons can be seen on the Tuning view as shown in Figure 8-42.

Simple Tuning Procedure Example

1. Select the desired Drive Type and Motor Type on the Drive/Encoder view in PowerTools Studio.
2. Verify that the values of each of the motor parameters have been entered correctly, paying close attention to the Motor Rotor Inertia and the Motor Ke.
3. Enter the calculated Inertia Ratio into the text box on the Tuning view.
4. Set the Position Loop Response to zero.
5. Enable the Velocity Feedforward parameter.
6. Configure an index that will simulate a step signal (make sure not to configure accel/decel values that you cause the motor to be current limited)
7. Initiate the index while recording velocity feedback on an oscilloscope.
8. Gradually increase the Velocity Loop Bandwidth until the velocity plot shows less than 20 % velocity overshoot with no ringing (instability).
9. Audible noise may be introduced by increasing the Velocity Loop Bandwidth due to velocity jitter. The Current Demand Filter can be used to reduce the noise. Begin with a 1 ms value in the Current Demand Filter and gradually increase until the noise is reduced.
10. After adding the Current Demand Filter, you may be able to slightly increase the Velocity Loop Bandwidth again.
11. Gradually increase the Position Loop Response until Following Error is minimized. Often, a value approximately 1/4th of the Velocity Loop Bandwidth will work well. Increase until oscillation is introduced and then slightly decrease.
12. Save file.

8.6.13 Errors View

The Errors view contains information about Drive Trips and Module Errors that are currently active as well as a log of the last ten drive trips. PowerTools Studio must be online with the PTi210 module to obtain this information. Figure 8-43 shows an example of the Errors view when online with the PTi210 module.

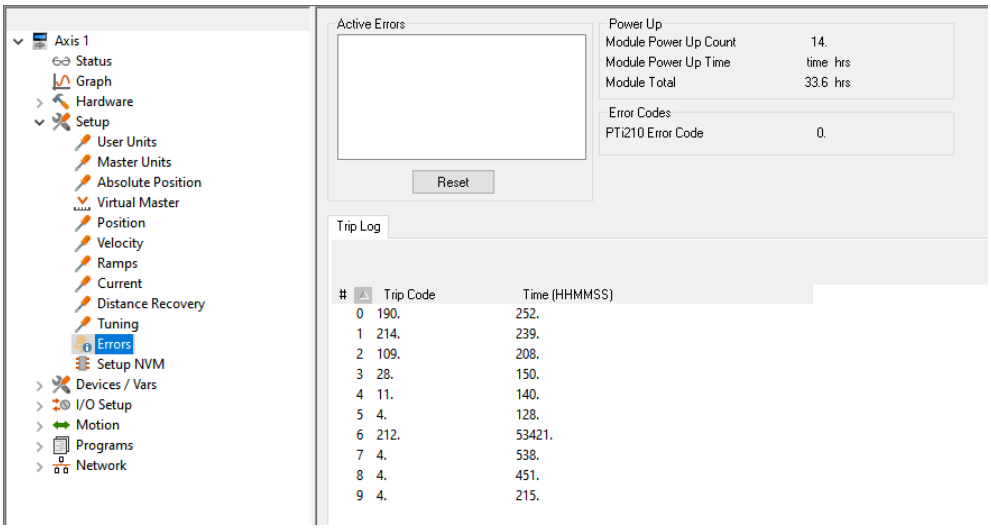


Figure 8-43: Errors View

Active Errors

The Active Errors box shows any drive trips or module errors that are currently active. The user can reset any drive trip or module error by clicking on the ***Reset*** button underneath the Active Errors box. If the situation that caused the trip or error has not been fixed, then the trip or error will activate again immediately.

Power Up

The Power Up box contains information related to the lifetime of the drive and the PTi210 module. Following are the parameters listed in the Power Up information box:

Module Power Up Count

Module Power Up Count is a current value of how many times the PTi210 module has been powered up. Each time power is cycled to the system, this number increments by one. This parameter is stored in the PTi210 module, and is not reset if the module is switched to another drive.

Module Power Up Time

The Module Power Up Time is the time elapsed since power has been cycled to the PTi210 module. The units for the parameter are Hours with a resolution of 0.1 Hours.

Module Total

Module Total is the total elapsed time that the PTi210 module has been powered up (since reset by the factory). The units for the parameter are Hours with a resolution of 0.1 Hours. This parameter is stored in the PTi210, and is not reset if the module is switched to another drive.

Drive Power Up Time

The Drive Power Up Time is the time elapsed since power has been cycled to the drive. The format for this parameter is Years.Days and Hours.Minutes. These values must be used in combination to find the actual time.

Example:

Drive Power Up Time = 2.123 Years.Days
05.22 Hrs.Min

In the example above, the time elapsed since the last power cycle is 2 Years, 123 Days, 5 Hours, and 22 Minutes.

Drive Run Time

The Drive Run Time is the Total Time that the drive has been powered up and the Bridge Enabled since last reset by the factory. The format for this parameter is Years.Days and Hours.Minutes. For an example of this format, please see Drive Power Up Time above.

Error Codes

The Error Codes box contains the currently active error code for the PTi210 module.

The error code is read directly from the drive parameter 50 of the relevant option slot setup menu (15, 16, or 17) for the PTi210 module. The Menu number depends on which slot is being referenced. Following is a list of the different error code references.

Parameter 15.050 – Slot 1 Error Code

Parameter 16.050 – Slot 2 Error Code

Parameter 17.050 – Slot 3 Error Code

For a list of the individual trip/error codes, please refer to the specific solutions module reference manual. Error Codes for the PTi210 module can be found in *Diagnostics* on page 239.

Trip Log

The Trip Log is history of the last 10 Trips that occurred on the Unidrive M/Digitax HD. This information is read directly from the Unidrive M/Digitax HD and is not stored in the PTi210 module. PowerTools Studio must be online to view the Trip Log.

Within the Trip Log Tab is the Parameter Trip 0 Time. The Trip 0 Time is a reference of when the most recent drive trip occurred. The time in this parameter is based off of the Drive Run Time described above. This means that the Trip time indicates when the trip happened in reference to the total time that the drive bridge has been enabled. The user can use the Drive Run Time as a gauge to determine exactly when the trip actually occurred.

The Trip 0 Time is displayed in a format of Years.Days and Hours.Minutes. These must be used in combination to find the time that the trip action happened.

Example:

Trip 0 Time = 0.005 Years.Days
17.35 Hrs.Min

In the example above, the Trip 0 occurred after 5 Days, 17 Hours, and 35 Minutes of run-time.

The rest of the Trip Log is shown as a chart. Following are the columns of data found in the chart:

Trip Number (#)

The 10 most recent Trips are stored in the drive. The most recent Trip is listed as Trip 0, and Trip 9 is the oldest. When the next fault occurs, Trip 0 will be moved down to Trip 1 and all the remaining trip numbers will also increment one. Trip number 9 will fall out of the log because it is no longer one of the ten most recent Trips.

Trip Code

The Trip Code displays the actual trip code from the Unidrive M/Digitax HD. For a reference of the Unidrive M Trip Codes, please refer to the Unidrive M *User Guide* or Unidrive M *Parameter Reference Guide*. For a reference of the Digitax HD Trip Codes, please refer to the Digitax HD *User Guide* or Digitax HD *Parameter Reference Guide*. Note that this trip code is not the same as the Trip Code or Error Code within an option module. For example, if an error occurs in a PTi210 module located in Slot 1, the Trip Log will not store the PTi210 module Error code, but instead it will store Slot1 Error (code #202) indicating that an error occurred in Slot 1. The specific error codes for the different options modules are NOT stored in the Trip Log but are stored in the Error Log.

Time Before Trip 0

All Trip Times for Trips other than Trip 0 are stored as a period of Time between Trip 0 and the specific Trip. The format for the Time is Hrs.Minutes. For example, if Trip 4 Time is shown as 1.20, that would signify that Trip 4 occurred 1 Hour and 20 Minutes prior to Trip 0. The largest value that can be stored in the Time Before Trip 0 is 600 Hours and 00 Minutes. If 600 Hours is exceeded, the 600 Hours and 00 Minutes is displayed in the Trip Log.

8.6.14 Setup NVM View

This view allows the user to configure which parameters are saved when power is cycled. If a parameter is not in this list, the value of the parameter will revert back to the value that was downloaded when power is cycled. If a parameter is in the list, the value that was being used just before power is cycled will be used when power is restored.

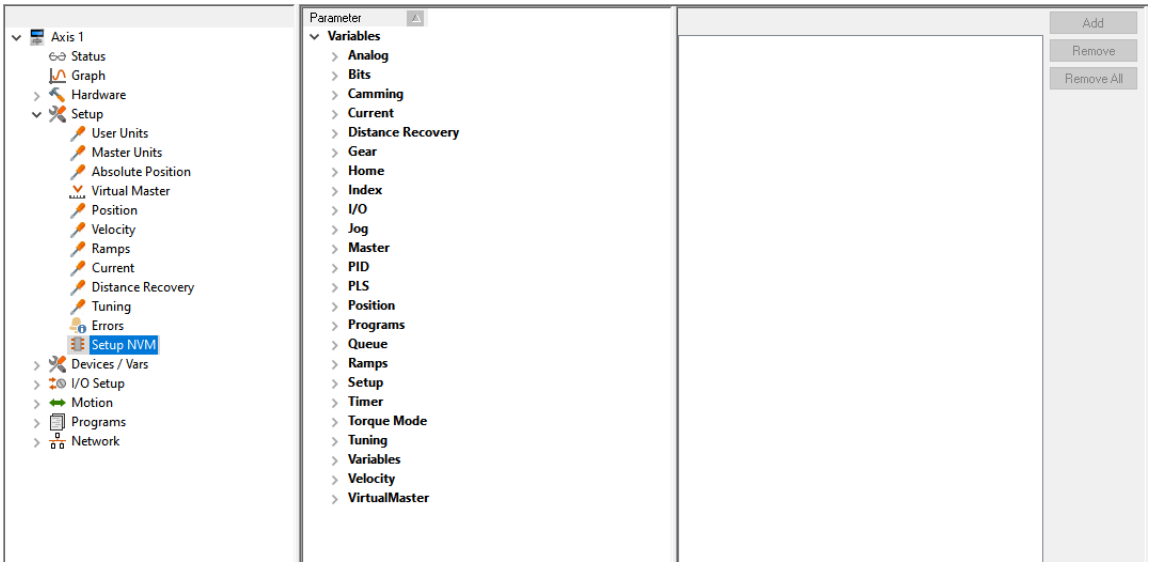


Figure 8-44: Setup NVM View

8.7 Devices / Vars

8.7.1 PLS View

The PLS view allows users to define the Programmable Limit Switches (PLS) for advanced machine operation. A PLS can be used to accurately turn on or off a bit based on the PLS Source value. There are eight global PLSs available for a single application. Figure 8-45 is an example of the PLS view.

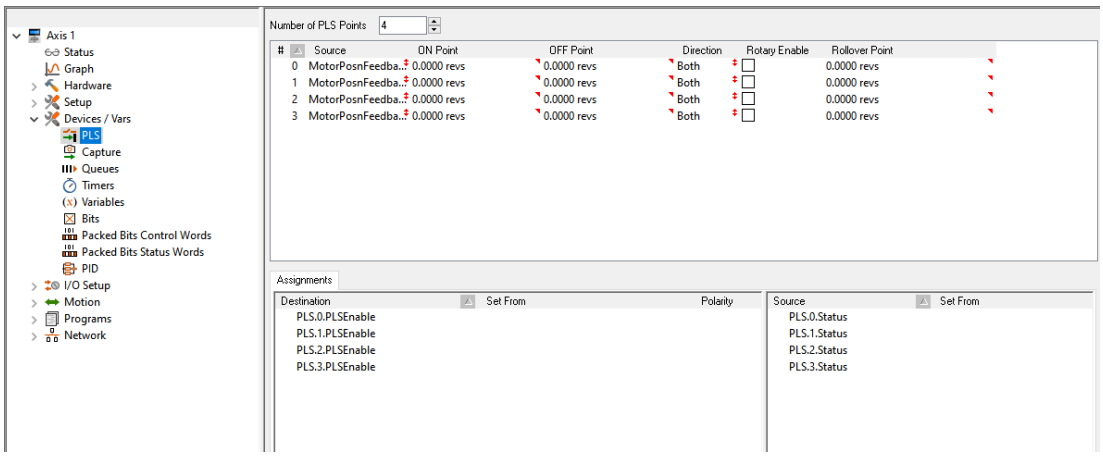


Figure 8-45: PLS View

Source

PLSs can be assigned to four different sources: the motor axis, (MotorPosnFeedback, MotorPosnCommand), a master synchronization signal, (MasterPosnFeedback) or the real time clock from the built-in microprocessor (FreeRunTime). This parameter determines which source the PLS uses to reference its OnPosn and OffPosn in order to determine the PLS.#.Status parameter.

On Point

PLS.#.Status will be active when the selected source position is between the PLS.#.OnPosn and the PLS.#.OffPosn. Assume that the PLS.#.Direction is set to "Both". When traveling in the positive direction and the feedback position executes the OnPosn, the PLS.#.Status will activate. As the motor continues in the same direction, the PLS.#.Status will deactivate when feedback position reaches or exceeds the OffPosn. If motor travel changes to the negative direction, the PLS.#.Status will activate when the feedback position reaches the OffPosn, and will deactivate when it continues past the OnPosn. All on/off positions are defined in user units.

PLS.#.Status will be active if: PLS.#.OnPosn ≤ Feedback Position ≤ PLS.#.OffPosn

Off Point

PLS.#.Status will be active when the selected source position is between the PLS.#.OnPosn and the PLS.#.OffPosn. Assume that the PLS.#.Direction is set to "Both". When traveling in the positive direction and the feedback position reaches the OnPosn, the PLS.#.Status will activate. As the motor continues in the same direction, the PLS.#.Status will deactivate when feedback position reaches or exceeds the OffPosn. If motor travel changes to the negative direction, the PLS.#.Status will activate when feedback position reaches the OffPosn, and will deactivate when it continues past the OnPosn.

PLS.#.Status will be active if: PLS.#.OnPosn ≤ Feedback Position ≤ PLS.#.OffPosn

If using negative values for your OnPosn and OffPosn, the most negative value should go in the OnPosn parameter, and the least negative value should go in the OffPosn.

If the PLS has a rollover point, and the OnPosn is greater than the OffPosn, the PLS will be active whenever the position feedback is not between the On and Off positions, and inactive whenever the position feedback is between the two positions. However, the PLS.#.Status will not turn on until it reaches the OnPosn the first time.All on/off positions are defined in user units.

Direction

This parameter specifies the direction of motion that a particular PLS output will function. If set to Both, the PLS will activate regardless of whether the motor (or master motor) is moving in the positive or negative direction. If set to Plus, the PLS will activate only when the motor is moving in the positive direction. If set to Minus, the PLS will activate only when the motor is moving in the negative direction.

For example, a flying cutoff or flying shear application may use this feature to activate the PLS to fire the knife only when the axis is moving in the positive direction.

Rotary Enable

This parameter is used to enable the RotaryRolloverPosn for this PLS.

Rollover Position

This parameter is the absolute position of the first repeat position for this PLS. When enabled it causes the PLS to repeat every time this distance is passed. The repeating range begins at an absolute position of zero and ends at the RotaryRolloverPosn.

For example, in a rotary application a PLS could be setup with an OnPosn of 90 degrees and an OffPosn of 100 degrees. If the RotaryRolloverPosn is set to 360 degrees the PLS would come on at 90, go off at 100, go on at 450 (360+90), go off at 460 (360+100), go on at 810 (2*360+90), go off at 820 (2*360+100), and continue repeating every 360 degrees forever.

8.7.2 Capture View

The Capture view allows the user to configure the various Capture object parameters. Figure 8-46 shows an example of the Capture view.

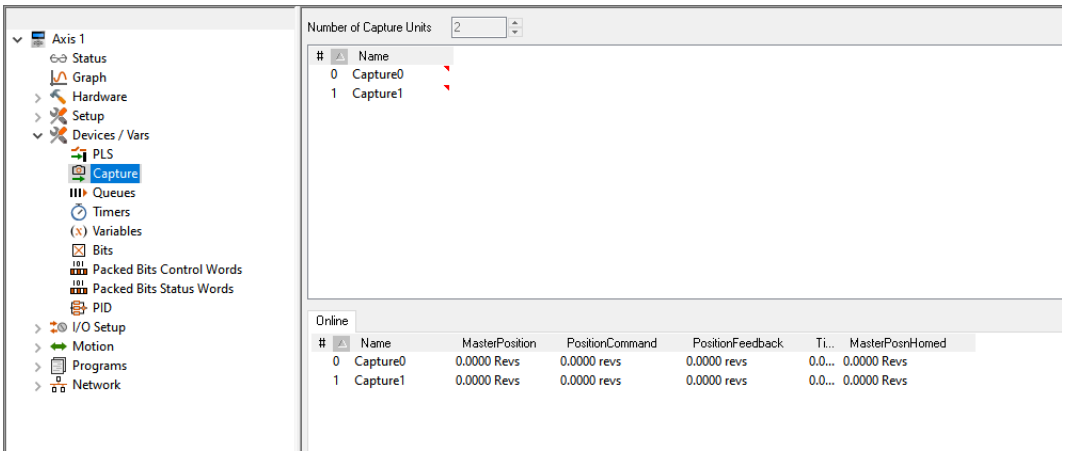


Figure 8-46: Capture View

A detailed explanation of each of the Capture components is below.

Number of Capture Units

This parameter defines the number of Capture objects available. Maximum is eight.

Name

You can assign a descriptive name to each capture, making the configuration easier to follow. The length of the text string is limited by the column width with a maximum of 12 characters.

Simply double click on the Name field of any capture line, enter the descriptive name and then enter, to assign a name to it.

Capture Parameters

The following four parameters are the data that is acquired by the position capture object. These parameters are available to be used as variables in a program. The four parameters can be accessed as follows:

Capture.#.CapturedTime - The time, in microseconds, from a free-running 32-bit binary counter at which the CaptureActivate signal activated.

Capture.#.CapturedPositionCommand - The command position, in user units, at the time when the CaptureActivate signal activated.

Capture.#.CapturedPositionFeedback - The feedback position, in user units, at the time when the CaptureActivate signal activated.

Capture.#.CapturedMasterPostion - The master axis feedback position, in master axis distance units, at the time when the CaptureActivate signal activated.

The captured data remains in these parameters until the capture component is reset and CaptureActivate is activated the next time. When the capture component is reset and CaptureActivate is activated, the data related to the previous capture will be over-written by the most recent capture data.

Capture Sources and Destinations

Figure 8-47 shows a block diagram of the Capture object

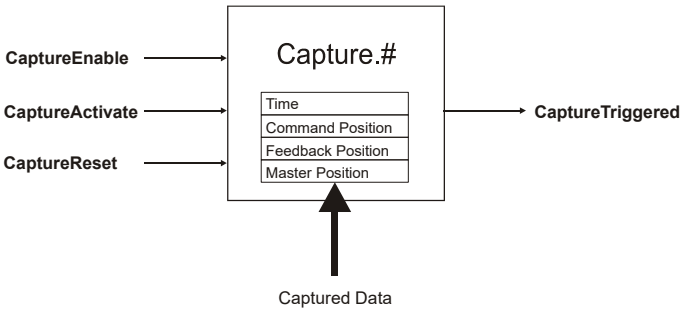


Figure 8-47: Capture Object

Sources

CaptureTriggered - The CaptureTriggered signal is read-only and indicates that the Capture component was activated, and that data has been captured. CaptureTriggered will activate on the leading edge of CaptureActivate if the Capture component is enabled and reset. CaptureTriggered will remain active until CaptureReset is activated.

Destinations

CaptureEnable - The CaptureEnable is used to enable or "arm" the capture component. If the CaptureEnable is not active, then the CaptureActivate has no effect, and the CaptureTriggered remains inactive. Once the CaptureEnable is activated, the Capture component is ready and waiting for a CaptureActivate signal to capture data. CaptureEnable is a read-only destination on the Assignments view, and is accessible through a user program.

CaptureActivate - If the Capture component is enable and has been reset (CaptureTriggered is inactive), then the rising edge of CaptureActivate will capture the four data parameters and cause CaptureTriggered to be activated. If the Capture component is not enabled, or has not been reset, the CaptureActivate will be ignored.

CaptureReset - The CaptureReset is used to reset or re-arm the capture component after it has been activated.

Once the capture has been activated, the CaptureTriggered destination will be active. The capture component cannot capture data again until it has been reset. The capture component will automatically reset itself if the CaptureEnable signal is removed.

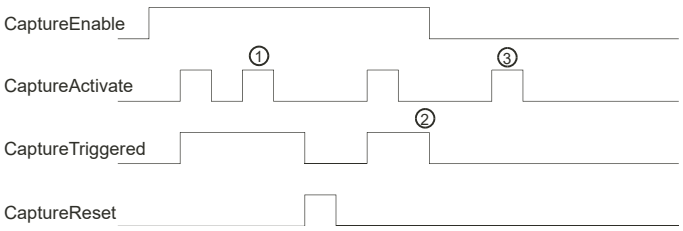


Figure 8-48: Capture Timing Diagram

Figure 8-48 is a timing diagram that shows how the different capture related sources and destinations function. The three numbers located on the diagram are associated to the following three notes respectively:

1. While CaptureTriggered is active, a rising edge on CaptureActivate will be ignored. The capture object can not activate again until it has been reset.

2. When CaptureEnable is deactivated, CaptureTriggered automatically deactivates because the capture object is automatically reset. The captured data is retained until the capture component is re-enabled and CaptureActivate is activated.

3. CaptureActivate has no effect while the capture component is not enabled.

Sources That Can Accurately Capture Data

Only a select few sources can accurately capture data. The reason that only certain signals can accurately capture data is because they are wired directly to the PTi210 module FPGA or the signal is generated internally by the processor and can be activated in the FPGA. Signals that are not wired to the FPGA or sent to the FPGA can be used to capture data, but it will only be accurate to the Trajectory Update Rate. This means that the capture will only occur the next time the signal is updated and not when it actually activates.

Sources that CAN accurately capture data:

- PTi210 module Digital Inputs 1 and 2
- Motor Encoder Marker Channel
- Index.#.CommandComplete
- Jog.#.CommandComplete
- Index.#.AtVelocity
- Jog.0.AtVelocity
- PLS.#.Status

All other Sources are only accurate to the Trajectory Update Rate.

Assignments that Automatically Use Position Capture

The sources listed above automatically capture data each time they activate even when a capture object is not configured. If any of these Sources are assigned to a destination, the captured data may be used by the destination function.

The Destinations that can use the automatically captured data are as follows:

- Index.#.Initiate
- Index.#.SensorTrigger

The automatically captured data is not available to the user like it is when a capture object is used. The processor takes care of resetting the automatic capture internally.

Index.#.Initiate - When one of the Sources listed above (Sources That Accurately Capture Data) is assigned to an Index Initiate, the captured information is automatically applied to the index starting point. This offers extremely high accuracy for initiation of motion, which is beneficial especially in synchronized applications.

Index.#.SensorTrigger - The sensor trigger destination used in registration indexes can use captured data to accurately calculate the ending position of the index based on the Registration Offset parameter. The Offset distance is added to the captured position to let the accurate stopping position for the registration index.

8.7.3 Queues View

The Queues view allows the user to configure the necessary parameters for the Queue object to function properly. Figure 8-49 shows an example of the Queue view.

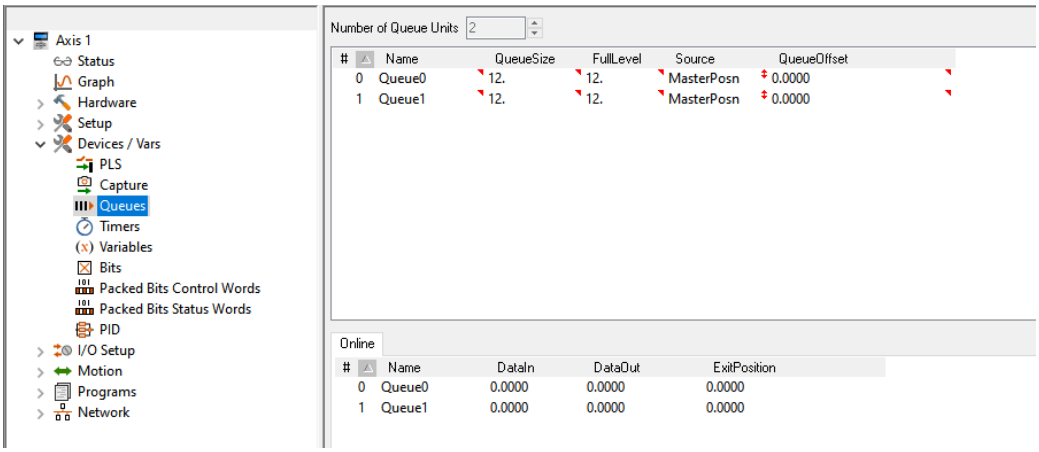


Figure 8-49: Queues View

A detailed explanation of each of the queue components is as follows:

Queue Data

The queue data is loaded into the queue by statements in the user program. Two types of data are most often used with the queue: Position Feedback, and Master Position Feedback.

Number of Queue Units

This selects the number of Queues available. Maximum of eight.

Name

You can assign a descriptive name to each queue, making the setup easier to follow. The length of the text string is limited by the column width with a maximum of 12 characters.

Simply double click on the Name field of any queue to assign a name to it.

Queue Size

This is the maximum number of elements that can be stored in the queue. If more than this number of pieces of data is in the queue at a time, then a Queue Overflow source will activate.

Full Level

The amount of data in the queue is constantly monitored and the Queue Full source will activate when the number of pieces of data in the queue exceeds the Full Level parameter.
Queue Full is only a flag and does not cause an error or a trip of any kind.

Queue Offset

The Queue Offset is the value that is added to the Queue Data and then compared to the selected source to determine when the Queue Exit event activates. For instance, if the source in selected source is set to Feedback Position, and the Queue Offset is set to 10, and the user puts the value 5 into the queue, the Queue Exit source will activate when the Feedback Position is greater than or equal to 5 + 10 or 15.

Source

The Source determines which parameter the sum of the Queue Data and Queue Offset are compared to in order to activate the Queue Exit function. If set to Position Feedback, the sum of the data and offset are compared to the Position Feedback parameter. If set to Master Position, then the sum is compared to the Master Feedback Position parameter, and if set to Command Position, then the sum is compared to the Motor Commanded Position.

Queue Sources and Destinations

Figure 8-50 shows a block diagram of the Queue object.

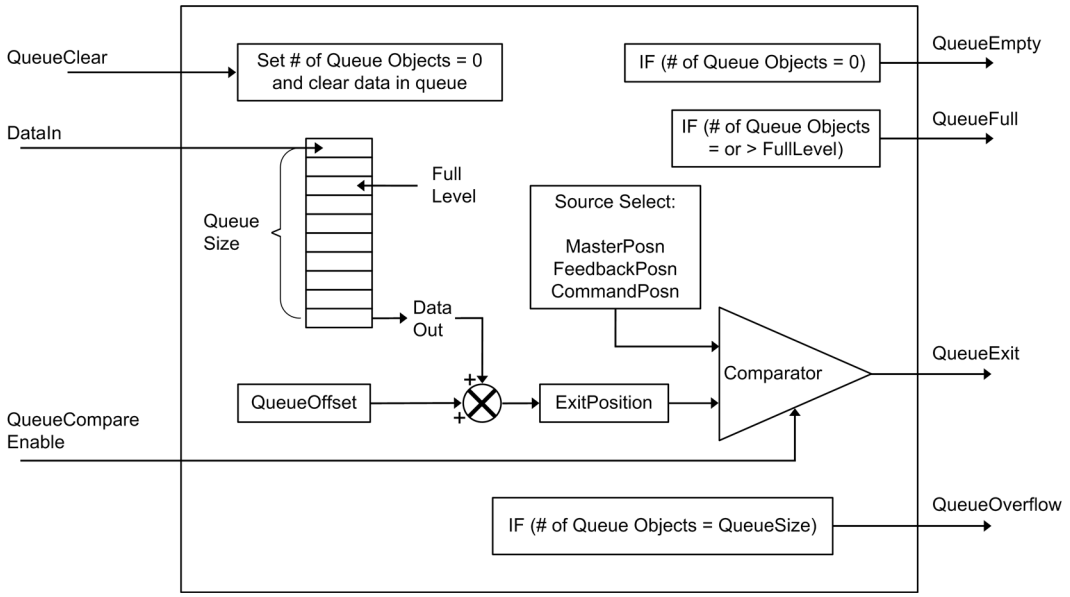


Figure 8-50: Queue Block Diagram

Sources

Queue Exit - This source activates when the Comparator Select parameter is greater to or equal to the sum of the data entered into the queue, and the queue offset.

Queue Empty - This source is active if no data is stored in the queue. It will become inactive when the first piece of data is loaded into the queue and remain inactive until all data has been removed from the queue.

Queue Full - The Queue Full source will activate if the number of pieces of data in the queue equals or exceeds the Full Level parameter. The source will deactivate when the number of pieces of data in the queue is less than the Full Level.

Queue Overflow - This source activates when there is no more room in the queue to store data. The maximum number of pieces of data is determined by the Queue Size parameter.

Destinations

Queue Clear - This destination automatically clears all of the data out of the queue. The cleared data is not saved and there is no way to recover the cleared data. This is typically activated on power-up of the system to make sure no old data remains in the queue.

Queue Compare Enable - The Compare Enable causes the comparator internal to the queue to function. If the Compare Enable is inactive, then the Queue Exit source will never activate.

If activated, then the Queue Exit source will activate when the Queue Data plus the Queue Offset is greater than or equal to the Comparator Select parameter.

To fully understand the operation of the queue, Figure 8-51 has a more detailed view of the Queue object.

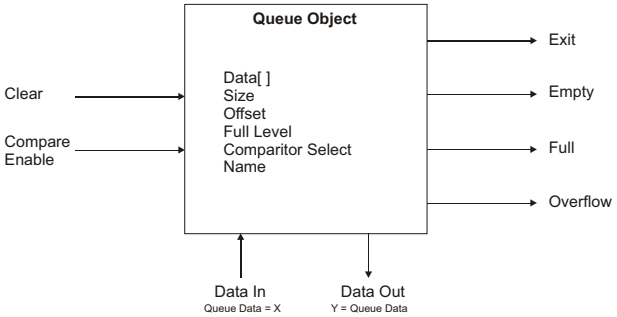


Figure 8-51: Queue Object and Components Diagram

8.7.4 Timers View

The Timers View is used to configure necessary parameters to use Timer objects within an application. Timers are used in many different types of applications to accurately control the timing of a given machine event with respect to one or more previous events. For example: When a digital input turns on, we want to start a program exactly 500 msec later. Or, when an index is complete, we want to hold an output on for exactly 2.000 seconds.

There are several different types of Timers available to the user based on the task they are trying to achieve.

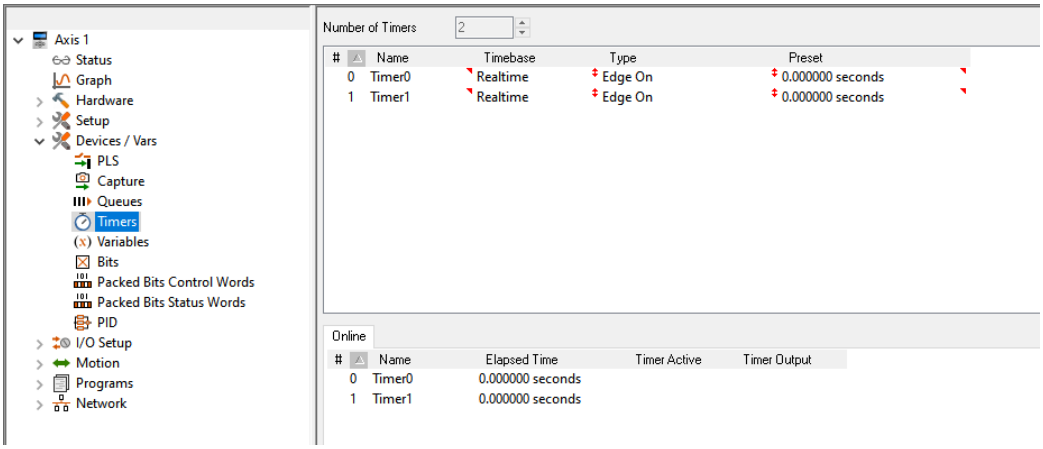


Figure 8-52: Timers View

Number of Timers

This parameter determines the number of Timer objects that are available in the configuration. Use the scroll buttons to change the number of Timers from 1 to 8. The Timers are numbered with a base number of 0. This means that if there are 5 Timers available, they will be numbered 0 through 4.

The number of Timers in the configuration can only be modified while offline.

Timer Name

The user can assign a descriptive name to each Timer, making the configuration easier to follow. The length of the text string is limited by the column width, with a maximum of 12 characters. Spaces are not allowed in the name, but underscores are. Simply click on the name field to modify the individual Timer Name.

Timer Type

There are several types of Timers available that help you to achieve different tasks. Use the combo box in the Type column to change the individual Timer Type. The Enable is not shown in the diagrams below. Assume in all cases the TimerEnable is ON. Select from these Types:

Edge ON Timer

With this type of timer, when the input of the timer activates, the internal clock begins to count. When the Preset time is reached, the Output of the timer turns ON.

In the case of the Edge type timer, once the rising-edge occurs on the Input, it doesn't matter what happens to the state of the Input after that. The Output will always turn ON when the Preset Time is reached (unless the Reset activates before the Preset Time is reached). The Timer Output will remain ON until the Reset is activated or the TimerEnable is deactivated.

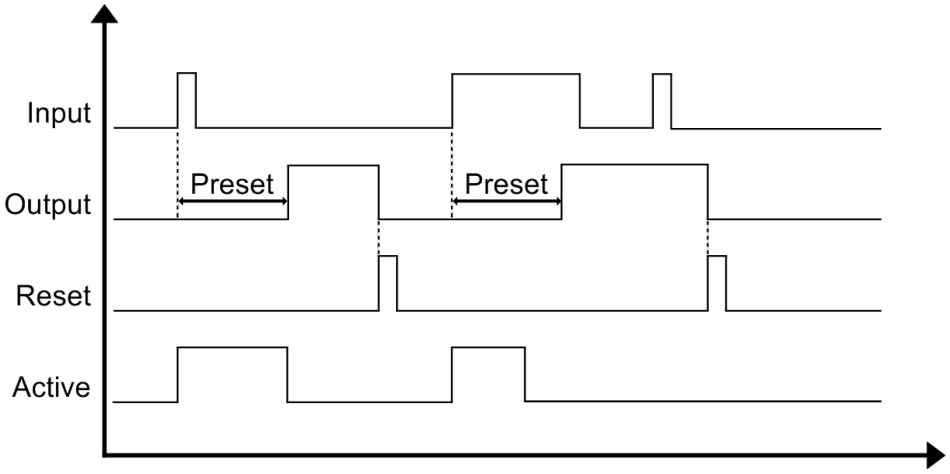


Figure 8-53: Edge On Timer - Timing Diagram

Level ON Timer

With this type of timer, when the Input of the timer activates, the internal clock begins to count. When the Preset time is reached, the output of the timer turns ON. If the Input of the timer turns OFF before the output turns ON, then the internal clock stops counting and the output will not turn ON. The whole sequence begins again the next time the Input turns ON. This means that the Input must remain ON for the duration of the Preset Time in order for the Output to turn ON.

The Level On Timer does not use the Reset, but rather, the Output will remain ON as long as the Input remains ON. When the Input turns OFF, the Output will automatically turn OFF.

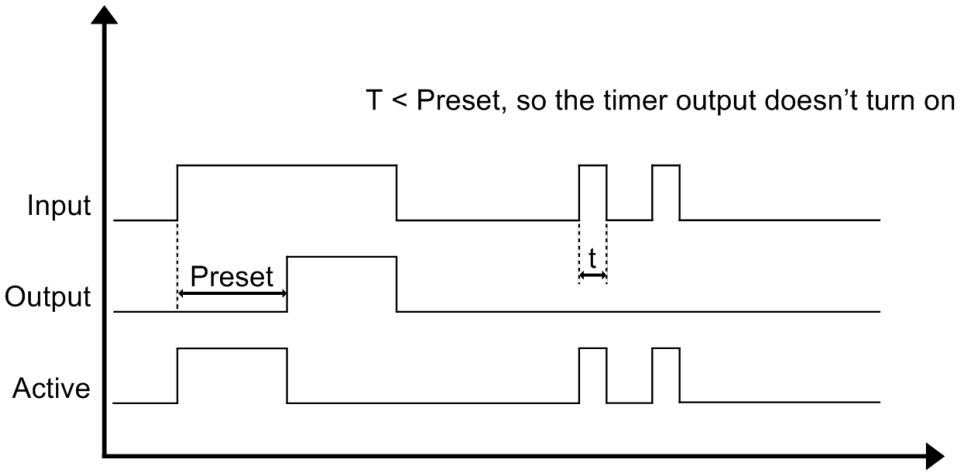


Figure 8-54: Level ON Timer - Timing Diagram

Edge OFF Timer

With this type of timer, when the Input of the timer activates, the Output turns ON automatically. When the Input turns OFF, the internal clock begins to count. Once the Preset time is reached, the Output of the timer turns OFF. As soon as the Input turns OFF, it does not matter what happens to the state of the Input, the Output will turn OFF when the Preset time is reached. The Output will remain OFF until the Reset is activated.

Note that if the Input is already ON when the Reset activates, the Output will not automatically activate at that point. The timer controller must see a rising edge on the Input to activate the Output.

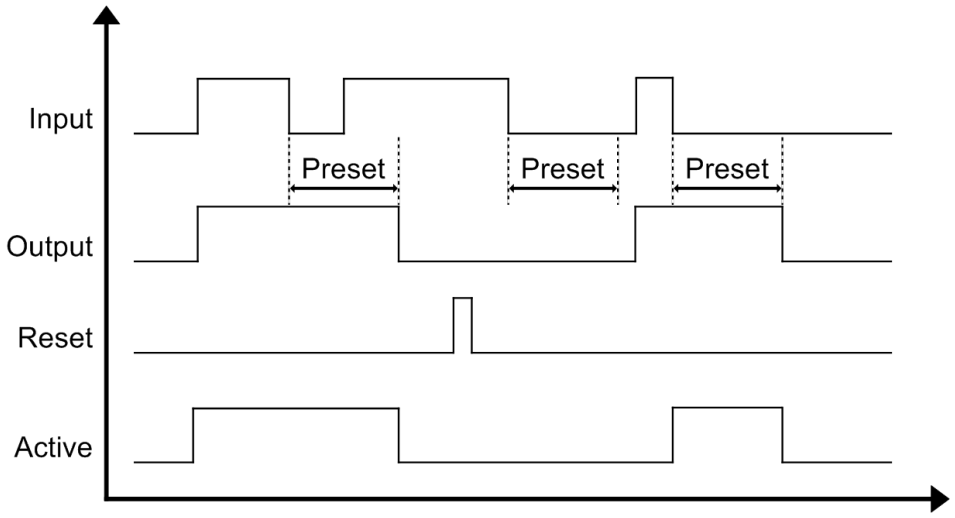


Figure 8-55: Edge Off Timer - Timing Diagram

Level OFF Timer

With this type of timer, when the Input of the timer activates, the Output turns ON automatically. When the Input turns OFF, the internal clock begins to count. Once the Preset time is reached, the Output of the timer turns OFF. If the Input turns ON again before the Preset time is reached, the clock resets to zero and waits again for the Input to turn OFF before starting the count.

The Level Off Timer does not use the Reset, but rather, the Output will remain OFF as long as the Input remains OFF.

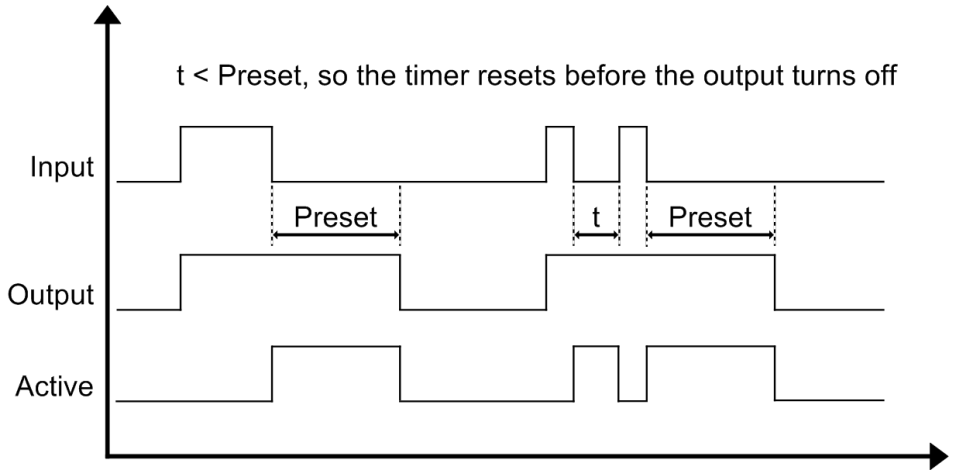


Figure 8-56: Level Off Timer - Timing Diagram

Cumulative ON Timer

This type of timer works exactly like the Level ON Timer, except that if the Input turns OFF before the Preset time is reached, the elapsed time is not reset. The total time that the Input is ON is added together such that when the total time of the Input being ON reaches the Preset time, the Output turns ON. Figure 8-57 below shows an example timing diagram of the Cumulative On Timer.

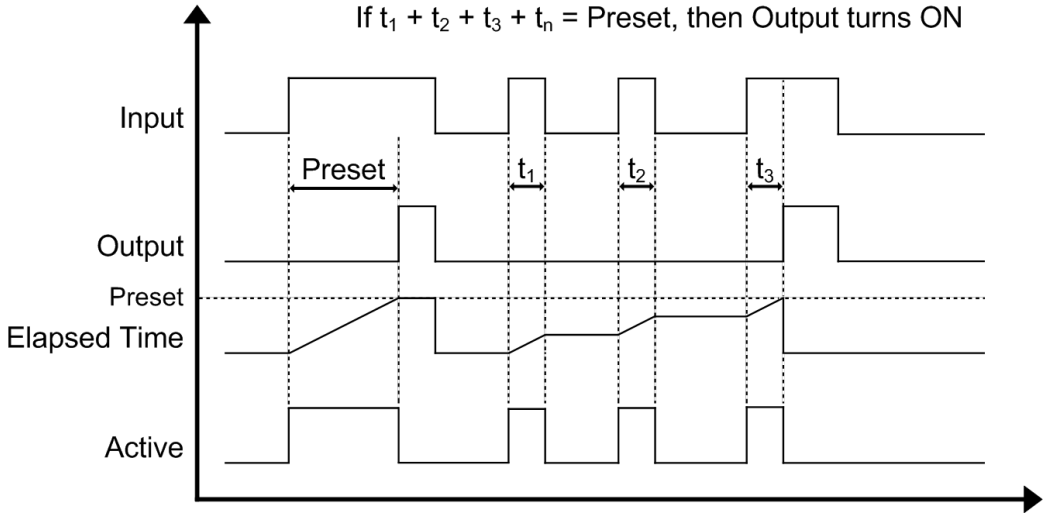


Figure 8-57: Cumulative ON Timer, No Reset - Timing Diagram

Additionally, a Reset signal can be used to clear the cumulative timer and restart it at zero. The figure below shows an example of this.

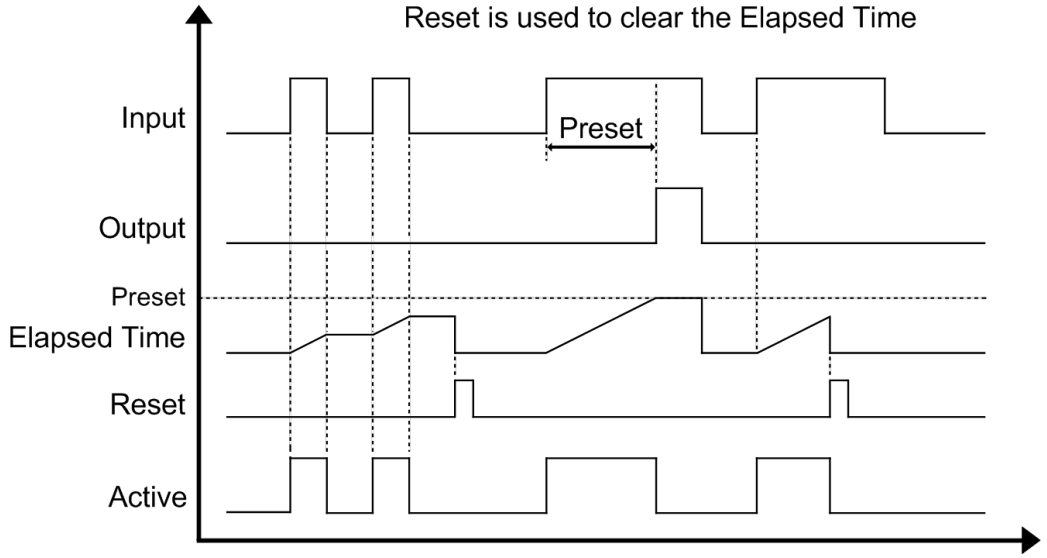


Figure 8-58: Cumulative ON Timer with Reset - Timing Diagram

Cumulative OFF Timer

This type of timer works exactly like the Level OFF Timer, except that with this type of timer, when the Input of the timer activates, the output turns ON. When the Input turns off, the internal clock begins to count. Once the Preset time is reached, the Output of the timer turns OFF. If the Input turns ON again before the Preset time is reached, the clock resets to zero and waits again for the Input to turn OFF before resuming the count again.

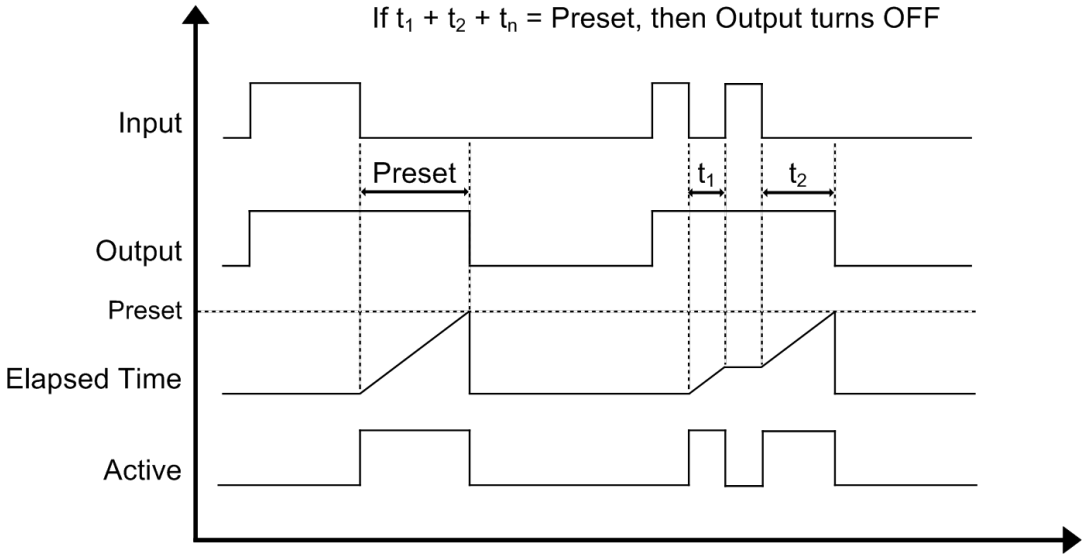


Figure 8-59: Cumulative Off Timer - Timing Diagram

Additionally, a Reset signal can be used to clear the cumulative timer and restart it at zero. The figure below shows an example of this.

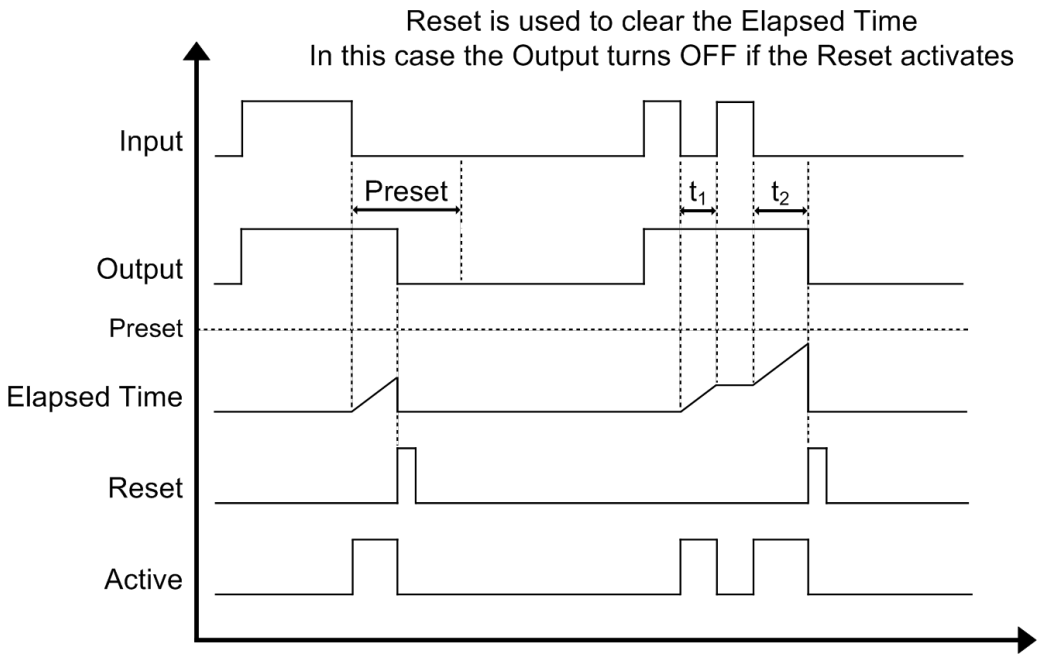


Figure 8-60: Cumulative Off Timer with Reset - Timing Diagram

Watchdog Timer

With this type of timer, when the Enable of the timer activates, the internal clock begins to count. If a rising edge is seen on the Input before the Preset Time is reached, the Elapsed Time is set to zero and the internal clock again starts counting immediately. The duration that the Input remains ON does not have any effect on the Output. If at any point, the Elapsed Timer reaches Preset time, the Output turns ON. The Output will remain ON until the Timer Reset is activated. If a rising edge is seen on the Input while the Output is ON, it will be ignored and the Elapsed Time will NOT be reset.

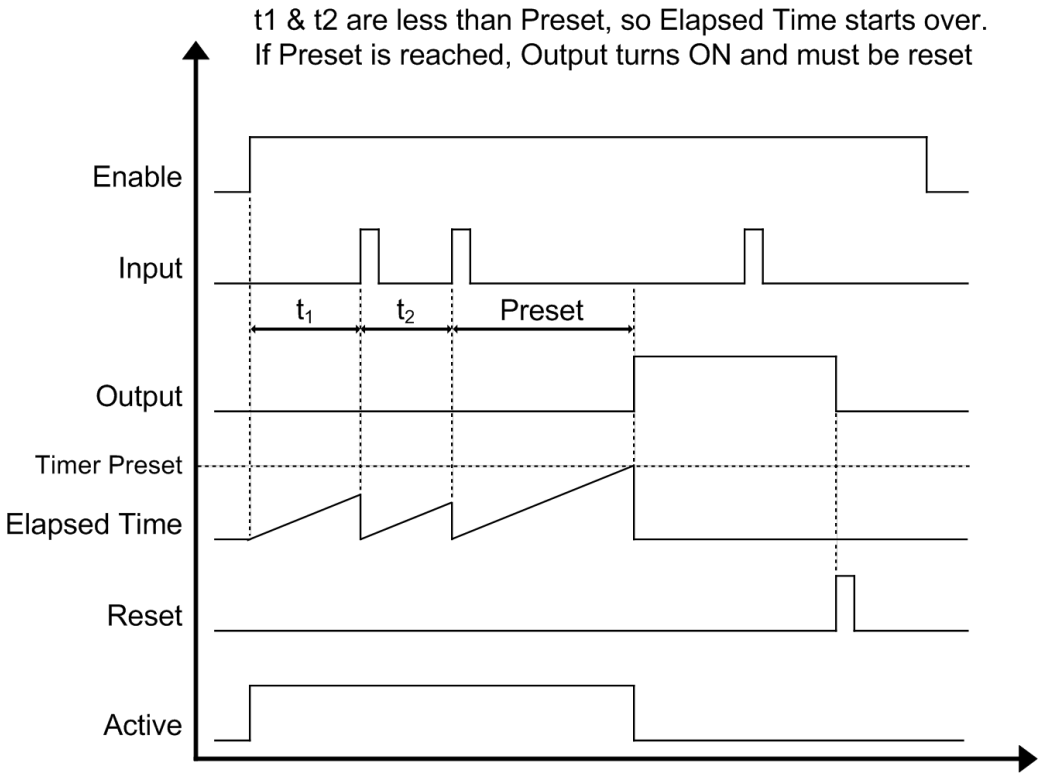


Figure 8-61: Watchdog Timer - Timing Diagram

One Shot

With this type of timer, when the Input activates, the internal clock begins to count. Once the Preset time is reached, the Output of the timer turns ON. If the Input of the timer turns OFF before the Output turns ON, then the internal clock keeps counting and the Output will turn ON when Preset time is reached. If the Output is active and a rising edge of the Input timer is received the Output will be reset. The Output will activate once the Preset time has past.

In the case of the One Shot type timer, once the rising-edge occurs on the Input, it does not matter what happens to the state of the Input after that during the Preset time. When past the Preset time and the Output is still active and a rising-edge occurs on the Input the timer will reset. The Output deactivates and will not become active until the Preset time has pasted.

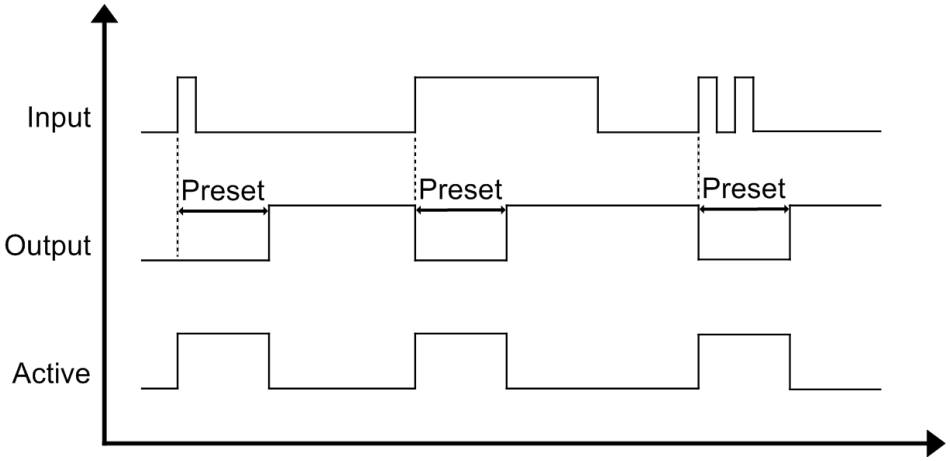


Figure 8-62: One Shot Timer - Timing Diagram

Timer Timebase

This parameter is used to select the desired Timebase for each timer object. If you want to control the Timer using actual time (seconds, or milliseconds, or microseconds) then the Timebase should be set to Realtime. Realtime is the default setting for the Timebase of each Timer object. When using Realtime, the Preset value of the Timer is entered in units of seconds. For example, after a program has run, you wish to hold an output active for ten seconds. This is an example that uses Realtime.

However, if you wanted to hold that same output active for ten revolutions of the Master Encoder instead of ten seconds, you set the Timebase to Synchronized. This means that you are using the master encoder position to control the duration of the Timer rather than actual Time. When the Timebase of the Timer is set to Synchronized, the Preset is entered in units of Master Distance rather than units of Time.

Use the combo box in the Timebase column to change the individual Timer Timebase settings.

Timer Preset

This parameter is used to control the actual duration of the timer delay.

If the Timebase is set to Realtime, the user specifies the Preset value in units of seconds with resolution of 1 microsecond (0.000001 seconds). If the Timebase is set to Synchronized, the Preset value is set in units of Master Distance and the resolution is dependant upon the Master Distance Units Decimal Places configured on the master Units view.

Once a timer has begun timing, if the Preset is modified, it is ignored until the next time the timer starts.

The Preset value must always be positive and can range from 0 to 2147483647 with the decimal place removed (example: 0.000000 to 2147.483647 with six decimal places).

Timer "Elapsed Time"

This is a read only parameter available on the Online tab that acts as feedback to the user to indicate how much of the Preset time has expired. This value starts at zero and counts up to the Preset value.

This parameter has the same resolution as the Preset parameter.

If a Timer is Reset or Enable deactivates, this value is zeroed out automatically. Once the Preset time is reached, the Elapsed Time will remain at its full preset value until the Timer is Reset, the Enable deactivates, or the Output deactivates.

8.7.5 Timer Signals/Events

The Timer objects has several inputs and outputs that are available on the Assignments view, or within a user program. A block diagram of the Timer object is seen in Figure 8-63 below.

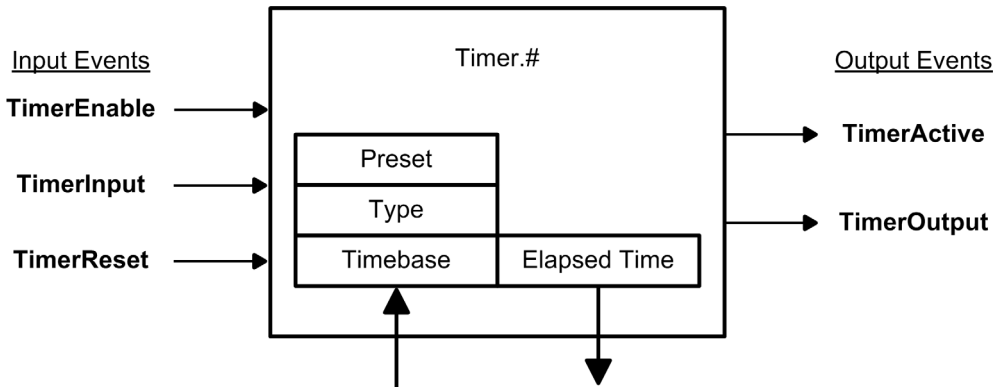


Figure 8-63: Timer Object Block Diagram

Input Events / Destinations

The following signals are available in user programs and as Destinations on the Assignments view.

Timer.#.TimerEnable

The Timer Enable event is level sensitive (not edge only) and is used to enable the Timer object. If a timer is not Enabled, the Output will never activate regardless of the state of the Input.

When the TimerEnable is deactivated, all output events associated with that Timer object will also deactivate and Elapsed Time will be set to zero.

All timer types use the Enable even though it is not shown in the individual timing diagrams. See examples of how TimerEnable affects timer functionality in Figure 8-64 and Figure 8-65 below.

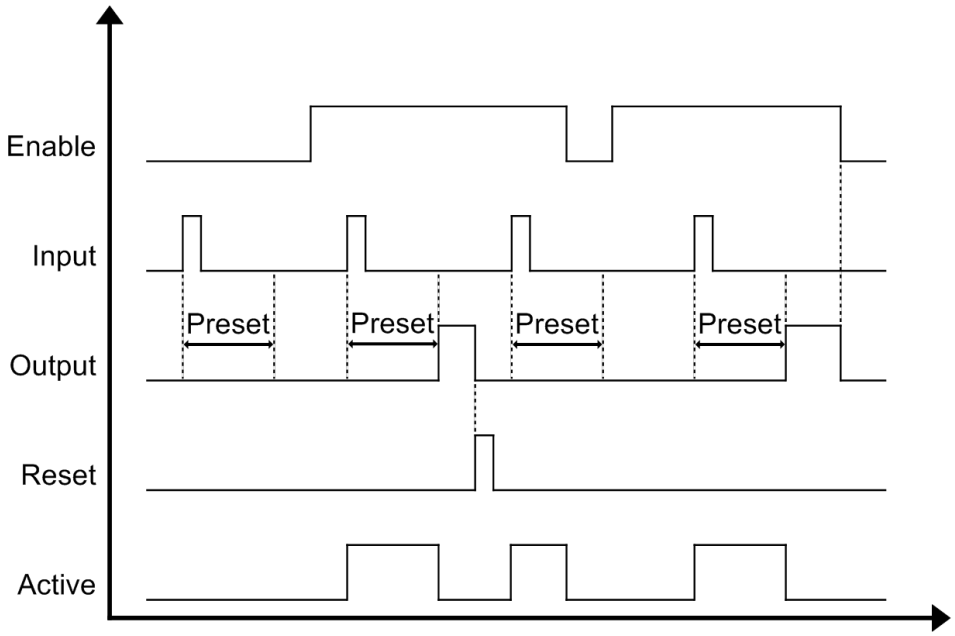


Figure 8-64: Timer Enable with Edge On Timer

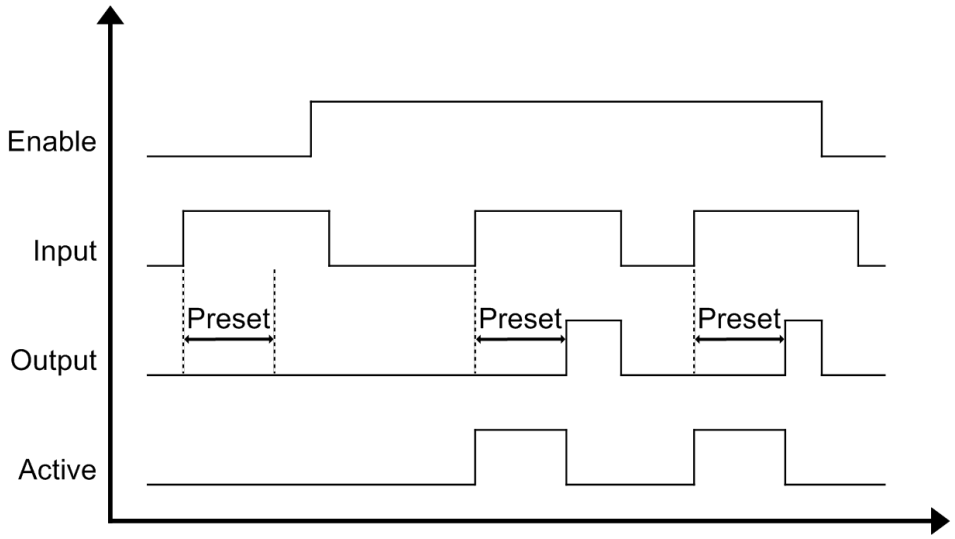


Figure 8-65: Timer Enable with Level On Timer

Timer.#.TimerInput

The Timer Input event is level sensitive (not edge only) and is used to define when the Timer starts counting against the Preset (along with the Timer Type). See the various timing diagrams to see how the Timer Input affects the Timer Output.

Timer.#.TimerReset

The Timer Reset event is used to reset the ElapsedTime to zero and deactivate the TimerOutput and TimerActive. Level On and Level Off timer types do not utilize the reset, as they reset automatically based on the state of the Input.

The Reset is a self-clearing function meaning that the reset occurs on the rising edge of the Reset input, and internally the reset is set off. Therefore a timer can immediately begin counting again even if the user holds the Reset input ON. Additionally, when resetting a timer in user program, the user must only set the Reset ON, and the controller will automatically set it off. There is no reason for the user to force the Reset OFF within the program.

Output Events / Destinations

The following signals are available in user programs and as Sources on the Assignments view.

Timer.#.TimerOutput

The Timer Output state is controlled by the Timer Input, the Timer Type, and the other configuration settings of the Timer object. See the various timing diagrams to see how Timer Output functions.

The Timer Output can be assigned to multiple Destinations on the Assignments view just as any other Output Event.

Timer.#.TimerActive

The TimerActive event is used as an indicator that the Timer is actively counting (i.e. In the case of an OFF Timer, the Active event would not activate until the Input turns OFF.).

In the case of Cumulative timers, the Active does not remain on even though the Elapsed Time may be non-zero, if the timer is not currently counting.

When a timer reaches the user defined Preset time, the Output activates, and the Active event turns OFF.

See the timing diagrams for each of the Timer Types for examples of how TimerActive behaves.

8.7.6 Using Timers within Programs

Timers can be used within user programs without the need for assignments. Following are some examples of using Timer functions within a program.

To start a Timer in a program, you must do the following:

```
Timer.#.TimerInput = ON
```

The user must also turn the Input off as necessary to control the timer:

```
Timer.#.TimerInput = OFF
```

To use the output of a Timer within a program:

```
Wait For Timer.#.TimerOutput = ON (or = OFF)
```

To change the value of a Timer Preset in a program:

```
Timer.#.TimerPreset = 12.345 'Seconds
```

To check if a timer is running:

```
If (Timer.#.TimerActive = ON) Then
    Some code
    .
    .
    Endif
```

To wait for a Timer to start running:

```
    Wait For Timer.#.TimerActive = ON
```

Other examples:

```
If((Timer.#.TimerActive=ON) AND (Timer.#.TimerElapsedTime<0.500)) Then
    Some code
    .
    .
Endif

If ((Timer.#.TimerOutput = ON) AND (Index.#.CommandComplete)) Then
    Timer.#.TimerReset = ON 'Self-resetting - No need to turn OFF
Endif
```

8.7.7 Variables View

Variables allow the user to store data related to their application into a parameter, which the user can name. The user must define each variable by giving it a Name, Resolution (number of decimal places), and Initial Value. All variables are signed 32-bit parameters. Figure 8-66 shows an example of the Variables view.

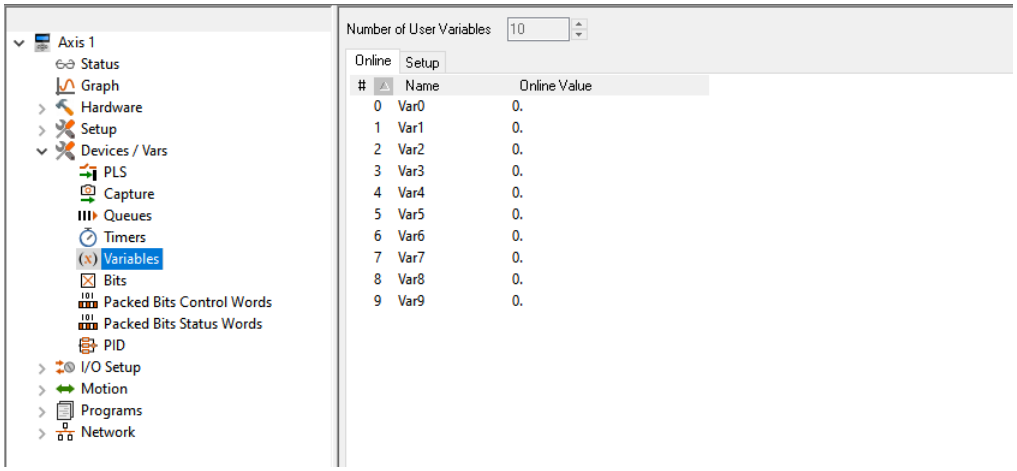


Figure 8-66: Variables View

The following parameters are part of the Variable definition:

Name

This is a twelve-character string that allows the user to assign a descriptive name to the parameter. Spaces are not allowed in the name of a variable.

Decimal

This parameter defines the number of digits (up to 6) to be used after the decimal point for the specific variable. This is the maximum resolution that the parameter will have.

Initial Value

This is the initial value of the variable that will be used on power up. If the variable has been configured as a Save to NVM parameter, then the value in NVM will overwrite the initial value on power up.

Adding and Deleting Variables

The default number of variables is ten. To add more user variables, click the up arrow next to the Number of User Variables box on the User Variables view. The maximum number of user variables is 255.

Only the last variable in the list can be deleted. To delete the last variable, simply click the down arrow next to the Number of User Variables box.

User variables are all Global variables. A Global variable means that it can be accessed from any program.

Using User Variables in a Program

Once setup, Variables can be used inside a program for calculations, motion profile setup, or any other user-desired function. To access User Variables, click ***Drag In Variables*** button in the user program toolbar. User Variables is a branch in the Select Variables box.

8.7.8 Bits View

Bits act just like Variables except that they allow the user to store bit level parameters rather than 32-bit parameters. The user may customize each Bit by giving it a Name and an Initial Value.

The Name for each Bit may be up to 12 characters in length, and must start with an alpha character (non-numeric character). Spaces are not allowed in the Name for a Bit however, the underscore character ("_") may be used.

The Initial Value for each Bit is configured using a check box for the specific Bit. To make the Initial Value "On" or "Active", simply select the check box for that Bit. The default value for each Bit will be "Off" or "Inactive", the check box will be clear.

Bits are configured on the Bits view as shown in Figure 8-67.

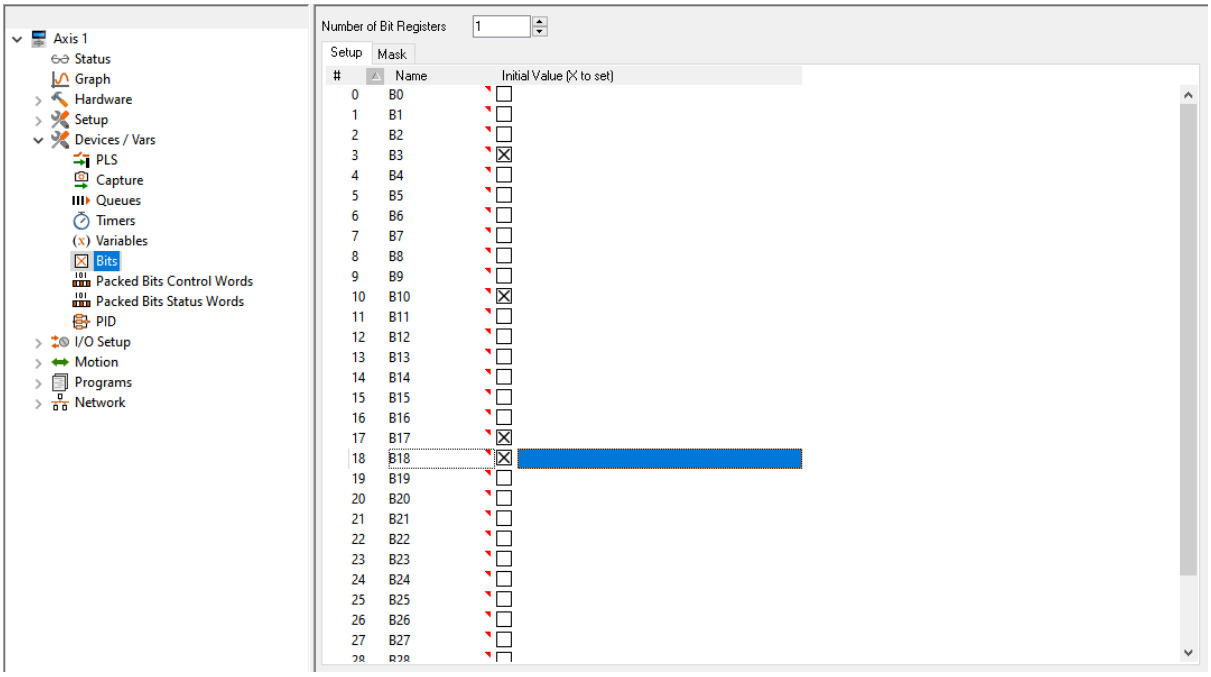


Figure 8-67: Bits View

The following parameters are part of the Bit definition:

Name
This is a twelve-character string that allows the user to assign a descriptive name to the Bit. Spaces are not allowed in the name of a Bit.

Initial Value
This is the initial value of the Bit that will be used on power up. If the Bit has been configured as a Save to NVM parameter, then the value in NVM will overwrite the initial value on power up.

Using Bits in a Program
Bits may be accessed in the User Program. Several examples of this are shown below.

```
Bit.Raise_Table = ON
Wait For Bit.Vacuum_ON = OFF
Wait For Bit.RunPart_A OR Bit.RunPart_B OR Bit.RunPart_C
If (Bit.RunPart_A = ON AND Bit.Vacuum_ON = OFF) Then
 Call Program.1
Endif
```

Bits are turned on by setting them equal to ON, TRUE, YES, SET, or ENABLE (not case sensitive), and can be deactivated by setting them equal to OFF, FALSE, NO, CLEAR, or DISABLE. Setting an individual bit equal to 1 or 0 in a user program will cause a red dot error. The Boolean values listed above must be used.

Adding and Deleting Bits
Bits can be added or deleted in groups of 32-bits called registers. The maximum number of registers is 8. Individual bits cannot be added or deleted. The default number of Bits available is 32 (1 register). To add an additional 32-bits, click the Up Arrow next to the Number of Bit Registers at the top of the Bits view (see Figure 8-67).

To decrease the number of Bits by 32, click the Down Arrow next to the Number of Bit Registers box. When decreasing the number of Bits, it is always the last 32-bits in the list that will be eliminated.

32-bit Bit Register and Bit Masking
When using different communications protocols (i.e. DeviceNet, Profibus, Modbus), it is often desirable to access groups of Bits in a single parameter, rather than having to access them individually. In the PTi210 module it is possible to access 32-bits in a single parameter. This parameter is named BitRegister.#.Value. Because some of the 32-bits may be used by the program, and should not be modified from the network communications, it is possible to "Mask Off" certain bits. Masking bits prevents them from being modified in the program when the 32-bit parameter is written to.

When a Bit Register (group of 32 Bits) is written to, the value is then logic-AND'ed with the mask to determine the resulting state of each of the 32 individual bits. If the individual bit value of the 32-bit mask is "1", then the corresponding bit from the written 32-bit parameter is passed through, and the resulting value stored in the specific bit will be the written bit value. If the bit value of the 32-bit mask is "0", then that particular bit is blocked (or masked), and the resulting bit value does not change, (Original Value AND NOT 32-Bit Mask) or (Value Written over Network AND 32-Bit Mask). An example of this is shown in Figure 8-68.

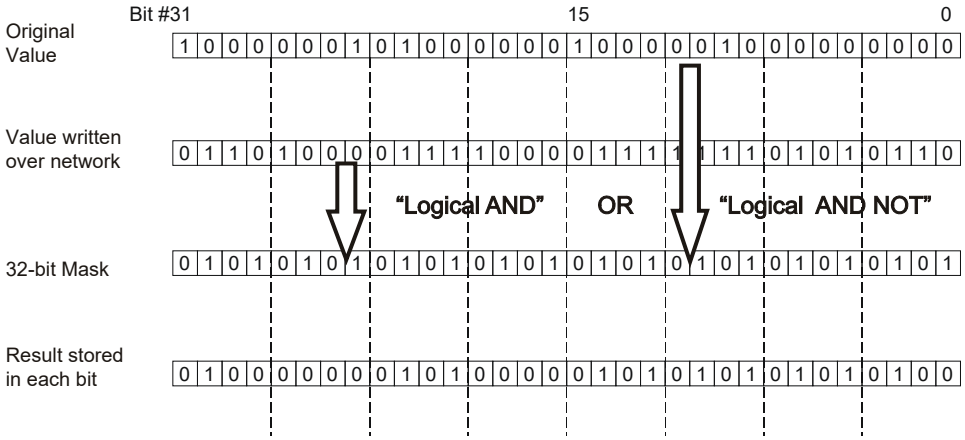


Figure 8-68: Writing to the Bit Register

The Mask is only used when WRITING to the 32-bit parameter, BitRegister.#.Value. When reading the 32-bit value, all bits are read regardless of the mask.

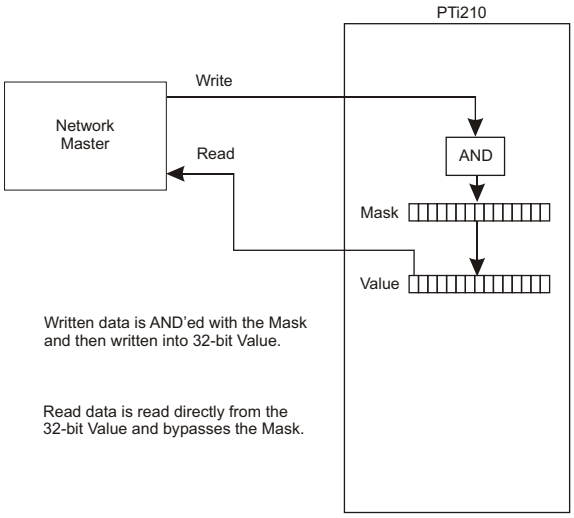


Figure 8-69: Bit Read/Write Process

Configuring the User Bit Mask Register

The Bit Mask is a 32-bit parameter that can be configured through Power Tools Studio, in the User Program, or over the communications network. The default value for the Mask register is 0xFFFFFFFF (HEX), or all bits ON. To change the Mask value using PowerTools Studio, navigate to the Mask tab on the Bits view, see Figure 8-70.

Figure 8-70: Bits View (Mask Tab)

In the Bits view - Mask Tab, each bit of the Mask can be set to 0 or 1 individually. ON (or 1) is indicated by a shaded square, and OFF (or 0) is indicated by an empty square. Bit 31 is the most significant bit in the word, and bit 0 is the least significant bit. If the bit is shaded, it means that particular bit will be passed through when written.

Each additional group of 32-bits that are added, a new Mask parameter will appear for that group. Mask 0 will control the mask for Bits 0 through 31. Mask 1 will control the mask for Bits 32 through 63. This sequence repeats for each additional 32 bits that is added.

To configure the mask in a user program, the parameter named BitRegisiter.#.ValueMask is written to. The mask can be written to using Hexadecimal based values or decimal based values. To write a hexadecimal value to the parameter, the hex value must be preceded with the characters "0x". To write a decimal value to the parameter, normal notation is used. For examples of writing the Mask to a value in a program, see below.

Example:

BitRegister.0.ValueMask = 0xFFFF0000

This example writes all bits of the upper sixteen bits, and 0 into each of the lower sixteen bits using hexadecimal notation.

Example:

BitRegister.0.ValueMask = 4294901760

This example writes all bits of the upper sixteen bits, and 0 into each of the lower sixteen bits using decimal notation.

8.7.9 Packed Bits

Packed Bits are 16-bit words configured by the user to read and write groupings of bit-level parameters (boolean and input/output events). Packed Bits are broken into Control Word(s) and Status Word(s). These words, once configured, can then be accessed via communication networks and/or within user programs to handle the bits as desired.

Control Words

A master device such as a PLC or HMI writes to the Control Word(s) as a 16-bit word. The data within that word is then broken into individual bits and written to the bit-level parameters that the user has mapped (see Figure 8-71). Note that if one of the bit-level parameters within the control word is modified from some other means (i.e. a user program or an Assignment), that the value of that parameter is not written into the Control Word value. For example, if a user initiates a Jog Plus within a user program, Bit0 of the ControlWord in Figure 8-71 would not automatically change to 1. Bit0 will only be 1 if the master device sets that bit when writing to the Control Word.

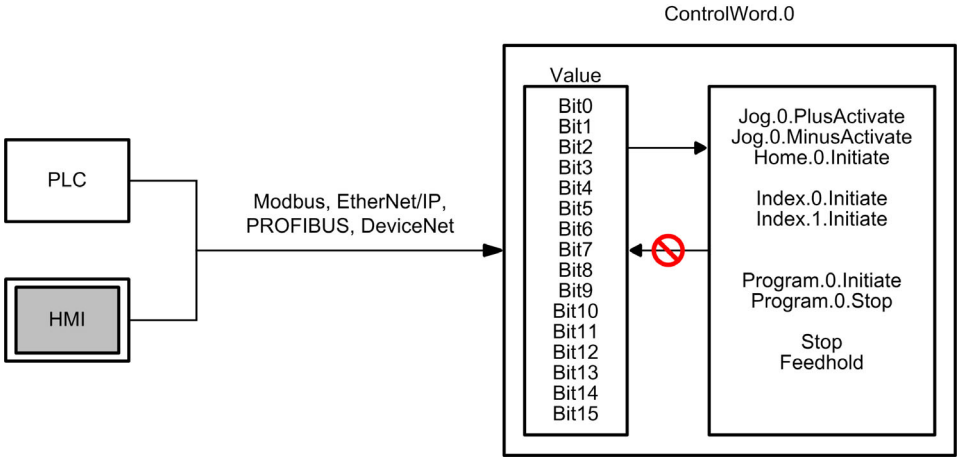


Figure 8-71: Control Words Communications Example

Status Words

Status Words are read by the master device as a 16-bit word. The drive assembles the bit-level parameters mapped by the user into 16-bit words for efficient transfer of status information to the master (see Figure 8-72).

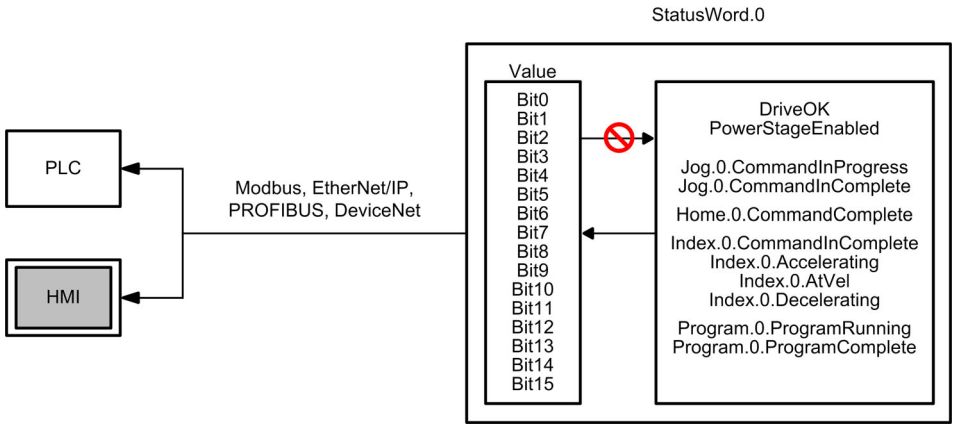


Figure 8-72: Status Word Communication Example

8.7.10 Packed Bits Control Words View

Control Words handle data being written to the user defined drive bit-level parameters. The user configures the Control Word(s) by dragging-and-dropping the desired bit-level parameters they wish to write into the Control Word mapping. Figure 8-73 shows the Packed Bits Control Words view onto which the user has dragged several motion initiate parameters.

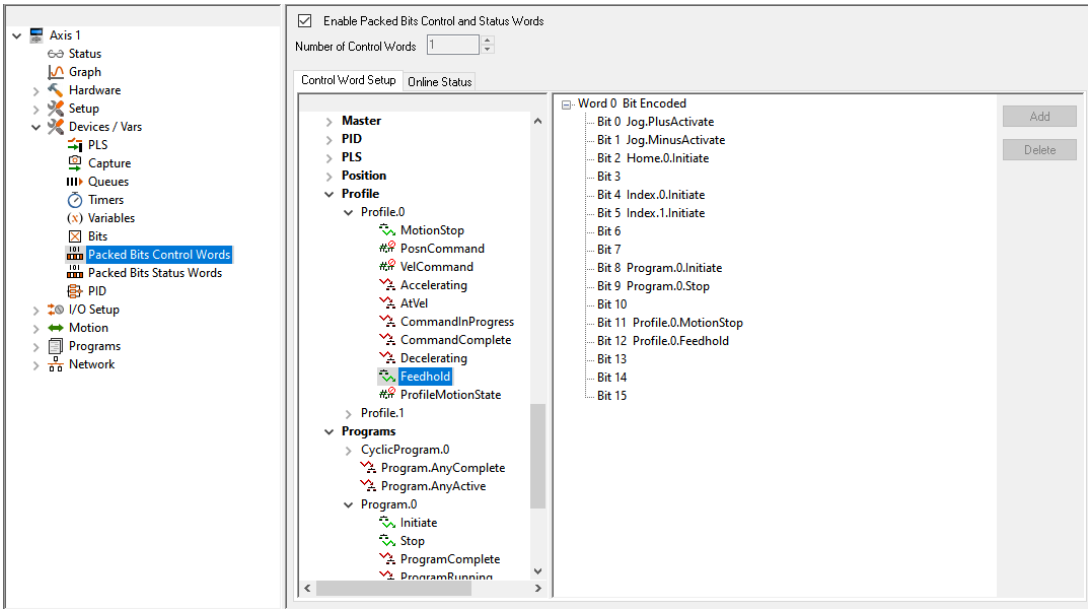


Figure 8-73: Packed Bits Control Words View

Enable Packed Bits Control and Status Words

In order to use the Pack Bits Control Word and Status Word mapping, the function must be enabled by activating the Enable Packed Bits Control and Status Words check box. If the check box is clear, the view remains blank (default). If the check box is selected, then the mapping tree appears which allows the user to configure the Control Word(s) as desired. This check box enables both Control Words and Status Words, and the two cannot be enabled/disabled independently.

Number of Control Words

The user can specify the number of desired 16-bit Control Words for the application. This parameter is adjustable from 1 to 4 words (1 default) using the arrows next to the box. The Number of Control Words cannot be adjusted while online with the device. If online, the user must disconnect communications and then adjust the number of control words.

Control Word Setup Tab

The Control Word Setup Tab is used to map the desired bit-level parameters onto the existing control word(s). The parameters can be mapped using one of two methods:

Drag and Drop: Using the Drag and Drop method, the user navigates the parameter tree, which has been pre-filtered to only show bit-level parameters, to find the desired parameter they wish to control (write to). Once the desired parameter is found, the user left-clicks and hold on the desired parameter. While still holding the left-button down, the user drags the parameter onto the desired bit of the desired Control Word. Then the left-button is released. Doing so will map the selected bit parameter to the selected Control Word bit.

Add Button: Using the Add Button method, the user navigates the parameter tree to find the desired parameter they wish to control. Once the desired parameter is found, the user left clicks on the parameter to highlight it. Then the user clicks on the desired bit of the Control Word to highlight it. Once both are highlighted, the ***Add*** button on the right side of the view (see Figure 8-73 above) should become available. Simply click the ***Add*** button and the highlighted parameter will be assigned to the highlighted bit on the Control Word.

To delete parameters from the Control Word, simply drag and drop the parameter off the control word (see explanation above), or highlight the desired parameter to delete, and click the ***Delete*** button on the right side of the view.

NOTE

Parameters can be mapped while online, but cannot be sent to the drive using the Update to RAM function. Therefore, any changes to Control Word mapping requires a full download.

Online Status Tab (Available Online Only)

While online with the drive, the user can monitor the Control Words value (see Figure 8-74). The data is displayed in binary form for each individual bit of the Control Word(s), as well as hexadecimal and decimal format for each Control Word.

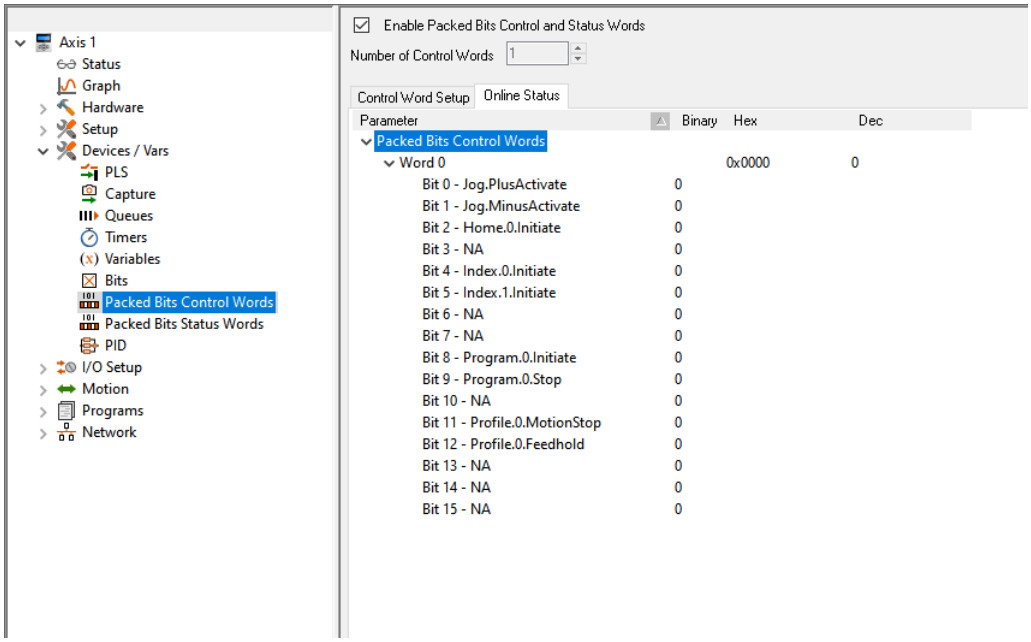


Figure 8-74: Control Word Online Status View

8.7.11 Pack Bits Status Words View

Status Words work to pass bit-level status information from the drive back to the master device as a 16-bit word(s). The user configures the Status Word(s) by dragging-and-dropping the desired bit-level parameters they wish to read onto the Status Word mapping. Figure 8-75 below shows the Packed Bits Status Words view onto which the user has dragged several status parameters.

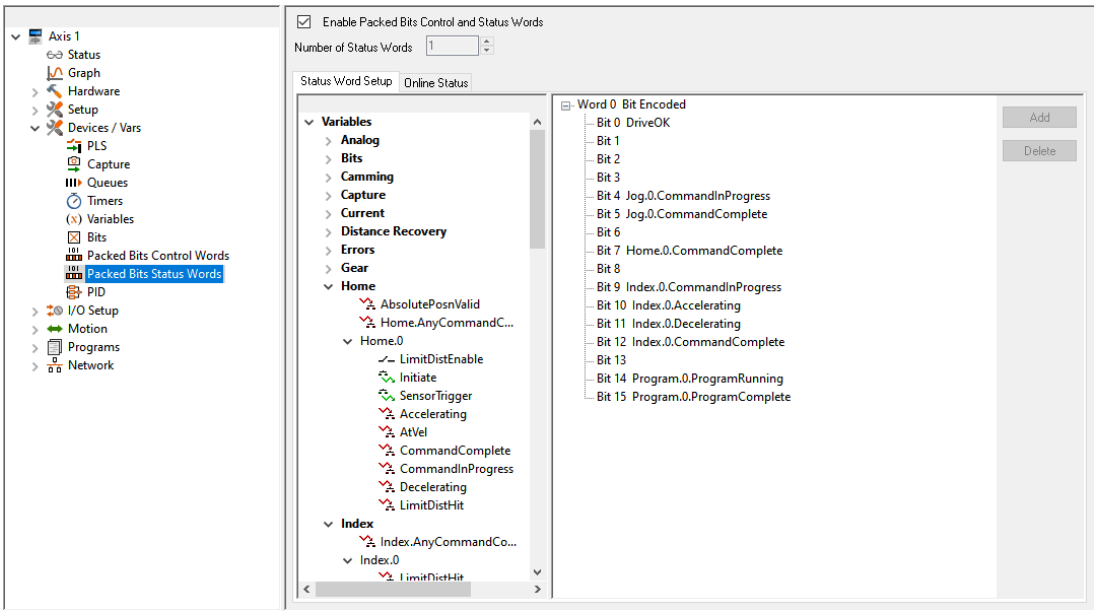


Figure 8-75: Pack Bits Status Words View

Enable Packed Bits Control and Status Words

In order to use the Pack Bits Control Word and Status Word mapping, the function must be enabled by activating the Enable Packed Bits Control and Status Words check box. If the check box is unchecked, the view remains blank (default). If the check box is checked, then the mapping tree appears which allows the user to configure the Status Word(s) as desired. This check box enables both Control Words AND Status Words, and the two cannot be enabled/disabled independently.

Number of Status Words

The user can specify the number of desired 16-bit Status Words for the application. This parameter is adjustable using the scroll buttons from 1 to 4 words (1 default). The Number of Status Words cannot be adjusted while online with the device. If online, the user must disconnect communications and then adjust the number of status words.

Status Word Setup Tab

The Status Word Setup Tab is used to map the desired bit-level parameters onto the existing status word(s). The method for mapping parameters to the Status Word(s) is the same as for the Control Words. See the Drag and Drop and Add Button methods described in the Control Words section above.

To delete parameters from the Status Word, simply drag and drop the parameter off the status word (see explanation above), or highlight the desired parameter to delete, and click the Delete button on the right side of the view.

NOTE

Parameters can be mapped while online, but cannot be sent to the drive using the Update to RAM function. Therefore, any changes to Status Word mapping requires a full download.

Online Status Tab (Available Online Only)

While online with the device, the user can monitor the Status Word value (see Figure 8-76). The data is displayed in binary form for each individual bit of the Status Word(s), as well as hexadecimal and decimal format for each Status Word.

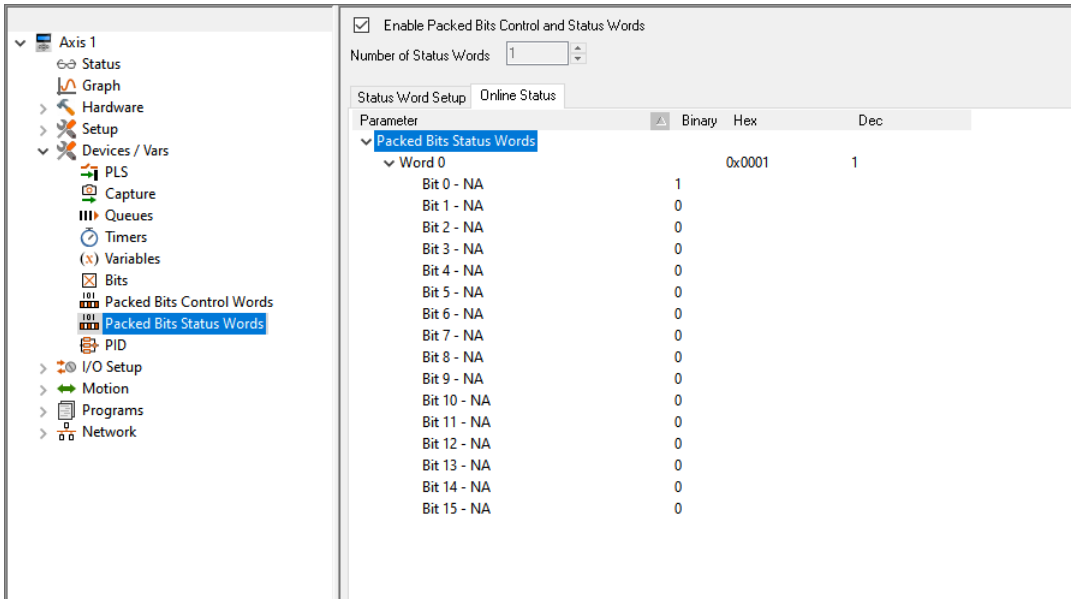


Figure 8-76: Pack Bits Online Status Tab

Using Pack Bits Status Words

Once the Status Word(s) have been configured by the user, they can be utilized by fieldbus communications (i.e. Modbus, EtherNet I/P, Profibus, DeviceNet, etc.) or within a user program.

8.7.12 PID View

The PID View provides access to the drive's PID menu setup parameters. To access the PID parameters, the PID Used checkbox must be enabled (checked).

Figure 8-77 below shows the PID View.

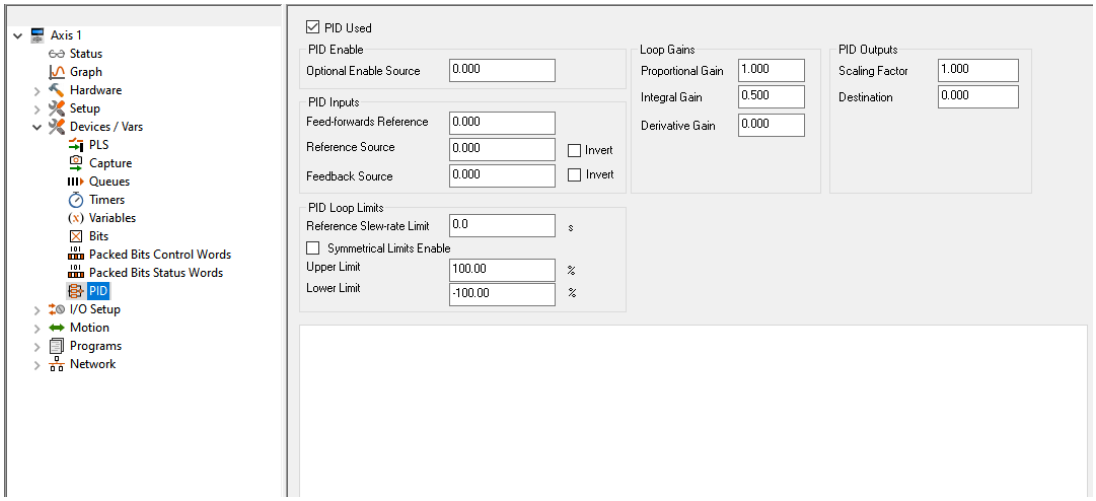


Figure 8-77: PID View

PID Used

When this checkbox is enabled (checked), the PID menu setup parameters will be shown and accessible for configuration within PowerTools Studio. When this checkbox is disabled (clear), the PID menu setup parameters will not be shown and therefore can't be modified.

For more information on the drive PID control system please refer to the Unidrive M/Digitax HD Control User Guide.

PID Enable – Optional Enable Source

This assignment specifies a drive menu parameter, the value of this menu parameter will be used in conjunction with Drive Healthy (Pr **10.001**) and PID1 Enable (Pr **14.008**) to enable the PID control.

If this assignment is set to 0, then the PID control is controlled by the drive's PID1 Enable (Pr **14.008**) and Drive Healthy (Pr **10.001**).

If this assignment is set to a menu parameter and the value of the menu parameter is 0, then the PID control will not enable, if, however, the value of the menu parameter is non-zero then the PID control will be controlled by PID1 Enable (Pr **14.008**) and Drive Healthy (Pr **10.001**).

This setting is only writable from the PowerTools Studio PID View and requires a reset to take effect.

PID Inputs Group

Main Reference Source

This assignment specifies a drive menu parameter which will be used as the source parameter for the PID1 main reference – PID1 Feed-fwds Ref Source (Pr **14.002**).

This setting is only writable from the PowerTools Studio PID View and requires a reset to take effect.

Reference Source

This assignment specifies a drive menu parameter which will be used as the source parameter for the PID1 reference – PID1 Reference Source (Pr **14.003**).

This setting is only writable from the PowerTools Studio PID View and requires a reset to take effect.

Reference Source - Invert

This checkbox allows the user to invert the Reference Source signal.

If this checkbox is enabled (checked) then the Reference Source signal is inverted.

If this checkbox is disabled (clear) then the Reference Source signal is unchanged.

Feedback Source

This assignment specifies a drive menu parameter which will be used as the source parameter for the PID1 feedback reference – PID1 Feedback Source (Pr **14.004**).

This setting is only writable from the PowerTools Studio PID View and requires a reset to take effect.

Feedback Source - Invert

This checkbox allows the user to invert the Feedback Source signal.

If this checkbox is enabled (checked) then the Feedback Source signal is inverted.

If this checkbox is disabled (clear) then the Feedback Source signal is unchanged.

PID Loop Limits Group

Reference Slew-rate Limit
This assignment specifies the time taken (in seconds) for the PID1 output to ramp from 0 % to 100 % following a step change from 0 % to 100 % in the PID1 input.

Symmetrical Limits Enable
The PID1 output may be limited to a specific range as defined by the Upper Limit and Lower Limit values, this allows the user to configure independent upper and lower limits.

This checkbox allows the user to apply symmetrical limits to the PID output.

If this checkbox is enabled (checked) then PowerTools Studio will apply a Lower Limit value which is equal in size but opposite in polarity to the Upper Limit value. For example, if the Upper Limit was set to 75.00 % then the Lower Limit value applied to the PID output would be -75.00 %. The Lower Limit box will also be dimmed and disabled.

If this checkbox is disabled (clear) then the Lower Limit value specified would be applied to the PID output.

Upper Limit
The PID1 output may be limited to a specific range as defined by the Upper Limit and Lower Limit values, this allows the user to configure independent upper and lower limits.

This assignment specifies the upper limit to apply for the PID1 output. The value is specified in % of output value and ranges from 0.00 % to 100.00 %.

Lower Limit
The PID1 output may be limited to a specific range as defined by the Upper Limit and Lower Limit values, this allows the user to configure independent upper and lower limits.

This assignment specifies the lower limit to apply for the PID1 output. The value is specified in % of output value and ranges from -100.00 % to 100.00 %.

If the Lower Limit value is greater than the Upper Limit value then the PID1 output is held at the Upper Limit value.

Loop Gains Group

The Loop Gains group provides the user with the ability to modify the PID1 controller Proportional, Integral and Derivative (Differential) gains.

The PID1 output value (Pr **14.001**) is defined as follows:

PID1 Output = PID1 Error (Pr **14.022**) x [Kp + Ki/s + Kds/(0.064 s + 1)]

Proportional Gain (Kp)
This assignment specifies the proportional gain (Kp) value for the PID1 controller and ranges from 0.000 to 4.000.

Integral Gain (Ki)
This assignment specifies the integral gain (Ki) value for the PID1 controller and ranges from 0.000 to 4.000.

Derivative Gain (Kd)
This assignment specifies the derivative gain (Kd) value for the PID1 controller and ranges from 0.000 to 4.000. This term may also be referred to as the differential gain.

PID Outputs Group

The PID Outputs group provides the user with the ability to apply a scaling factor to the PID1 controller output and define a destination parameter for the PID1 Output.

Scaling Factor
This assignment specifies the scaling factor to be applied to the PID1 controller output value and ranges from 0.000 to 4.000.

Destination
This assignment specifies the drive menu parameter which the PID1 controller output value will be written to and ranges from 0.000 to 59.999.

8.8 I/O Setup

External control capability is provided through the use of assignments to the sources (Unidrive M/Digitax HD Inputs, PTi210 module Inputs, and any SI-I/O module Inputs) or the destinations (Unidrive M/Digitax HD Outputs, PTi210 module Outputs, and any SI-I/O module Outputs).

Assignments provide a mechanism for the user to define the internal and external dynamic control structure to separate complex motion profiles. These functions directly correspond to any input or output line on the drive or option modules. External controllers, such as a PLC or other motion controllers, may be connected to affect or monitor the device's operation.

Depending on the model number, the Unidrive M is equipped with up to six input lines, and up to four output lines.

Unidrive M700/M701
The Unidrive M700/M701 is equipped with six input lines, three of which can be configured as either input or output and a relay output.

Unidrive M702
The Unidrive M702 is equipped with two input lines, two output lines and a relay output.

Digitax HD

The Digitax HD is equipped with two input lines and two output lines.

The PTi210 module has an additional three input and two output lines.

The Unidrive M input/output lines and relay output can be accessed through the two removable 11-pin control connectors and 2-pin relay output connector respectively.

The Digitax HD input/output lines can be accessed through the single 2 x 8-pin removable control connector. The PTi210 module input and output lines are located on the front face of the PTi210 module.

All inputs and outputs are configured as sourcing and are designed to operate from a +10 Vdc to +30 Vdc power source. The user is responsible for limiting the output current to less than 10 mA or less for each digital output.

8.8.1 Assignments View

The Assignments view is used to tie a Source to a Destination. Destinations are found on the right side of the Assignments view, and are functions that need to be triggered, such as Index Initiates, Program Initiates, Jog Initiates and so on. Sources are located on the left side of the Assignment view and reflect the status of events that occur in the drive. These events are based on drive activity. Figure 8-78 shows an example of the Assignments view.

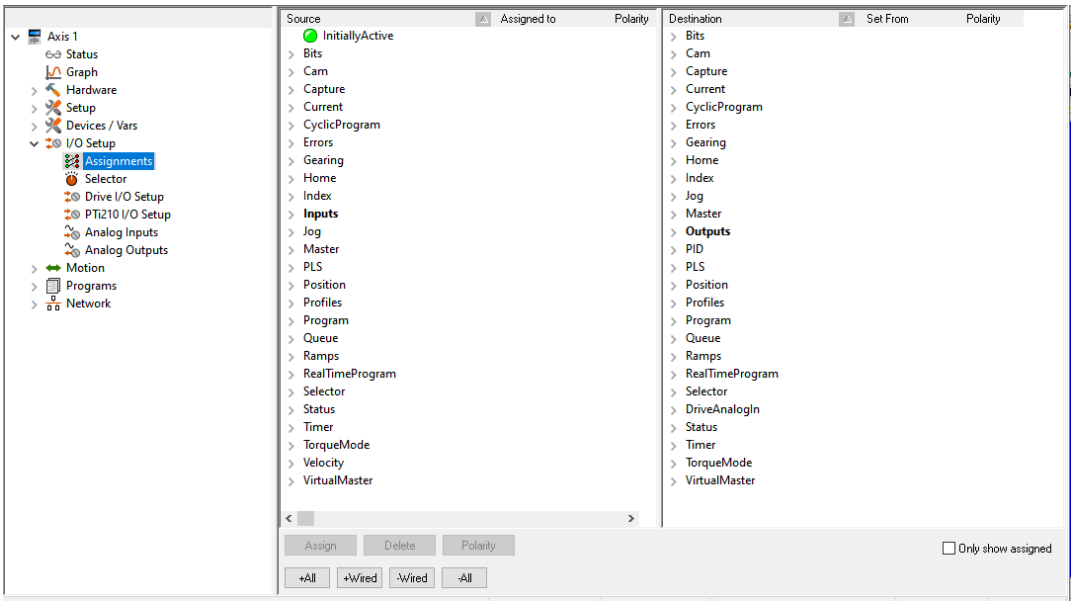


Figure 8-78: Assignments View

Sources and Destinations are put into groups based on functionality. By expanding individual groups, you will see the individual Sources or Destinations.

For example, if you expand the Inputs source group, you will see DriveIO, DriveInput, ModuleInput, and SlotX.DIO if a module is used. You can use these Source events to trigger certain actions (or destinations) on the right side of the view.

To make an assignment in the PTi210 module, you must assign a Source to a Destination. Any Source can be tied to any Destination to create the desired system operation.

Creating An Assignment

Various methods can be used to assign a Source (such as DriveInput.4) to a Destination (such as Index.0.Initiate).

Drag-and-Drop Method

Point to the Source, on the left, that is to be assigned to a Destination, on the right. Press and hold the left mouse button down, drag the Source until the mouse pointer is positioned over the desired Destination and release the left mouse button.

Once the mouse button is release, the assignment is created. The view will update to show that the assignment was created. In the Source side, the "Assigned To" column will show which Destination was assigned to the Source. In the Destination side, the "Set From" column will show which Source has been assigned to it.

Destinations can also be dragged from right to left, over to Sources.

Assign Button Method

Select both the Source and Destination that are to be assigned to each other. Once both are selected, the ***Assign*** button in the lower left corner of the view will become available. Click ***Assign*** to complete the assignment.

The view will update to show that the assignment was created. In the Source side, the "Assigned To" column will show which Destination was assigned to the Source. In the Destination side, the "Set From" column will show which Source has been assigned to it.

Any source can be assigned to up to ten different destinations maximum. Any destination can have as many sources as desired assigned to it.

Deleting An Assignment

Delete Button Method

Select both the Source and Destination for the assignment to be deleted. Once both are selected, the ***Delete*** button in the lower left corner will become available. Click ***Delete*** to remove the assignment.

Right-Click Method

Point to the specific assignment to be deleted and right-click. A shortcut menu appears, click **Delete**.

After either of these methods, the assignment will disappear. The "Assigned To" and "Set From" columns will no longer have any data for the specific Source and Destinations assignment.

Assignment Polarity

The active state of an assignment can be programmed to be Active Off, Active On, or Custom using PowerTools Studio. Making an assignment "Active On" means that the Destination will become active when the Source it is assigned to becomes active, and is inactive when the Source is inactive. Making an assignment "Active Off" means that the Destination will be active when the Source it is assigned to becomes inactive, and will be inactive when the Source is active.

The polarity of the assignment can also be changed to Custom when required. Custom polarity allows you to make a Destination activate and inactivate based on two different Sources.

NOTE

Destination functions that initiate motion (Jog.PlusInitiate, Jog.MinusInitiate, Index.#.Initiate, Home.#.Initiate, and Gear.Activate) should not be set to "Active Off". This could cause motion to initiate on loss of I/O Power.

Default polarity for a new assignment is "Active On". There are two methods that can be used to change the polarity of an assignment.

Polarity Button Method

Select both the Source and the Destination to be changed. Once selected, the ***Polarity*** button will become available in the lower left corner of the view. Click ***Polarity*** and change the settings as desired in the Polarity dialog box. Click ***OK*** to apply the changes.

Right Click Method

Point to the specific assignment you wish to change polarity on and right-click. A shortcut menu box will appear, choose **Polarity**. The Polarity dialog box will appear. Change the Polarity settings as desired and click ***OK*** to apply the changes.

Only Show Assigned

This check box removes the unassigned Sources and Destinations from the view. It allows the user to quickly see how many Sources and Destinations have been assigned.

User Level

The User Level filters the available assignments. The User Level is changed on the **Options** menu at the top of PowerTools Studio Toolbar. From the **Options** menu select **Preferences** then **User Levels**.

Easy mode filters out all but the most commonly used Sources and Destinations. **Detailed mode** filters out less, expanding the list of Sources and Destinations for more complex configurations. **Too Much mode** does not filter at all and provides all Sources and Destinations.

Expanding/contracting the Assignments View tree

The Assignments View tree can be expanded or contracted as required using the four buttons +All, +Wired, -Wired, -All.

+All

This button expands all elements in the Assignments View whether they are assigned (wired) or not.

+Wired

This button expands all assigned (wired) elements in the current Assignments View.

-Wired

This button contracts all unassigned (un-wired) elements in the Assignments View and expands the assigned (wired) elements, allowing the user to quickly view only the assigned elements.

-All

This button contacts all elements in the Assignments View.

8.8.2 Selector View

The Selector view allows the user to configure the Selector object. The selector uses a binary to decimal conversion, which requires fewer I/O points than direct assignments. Figure 8-79 shows an example of the Selector view.

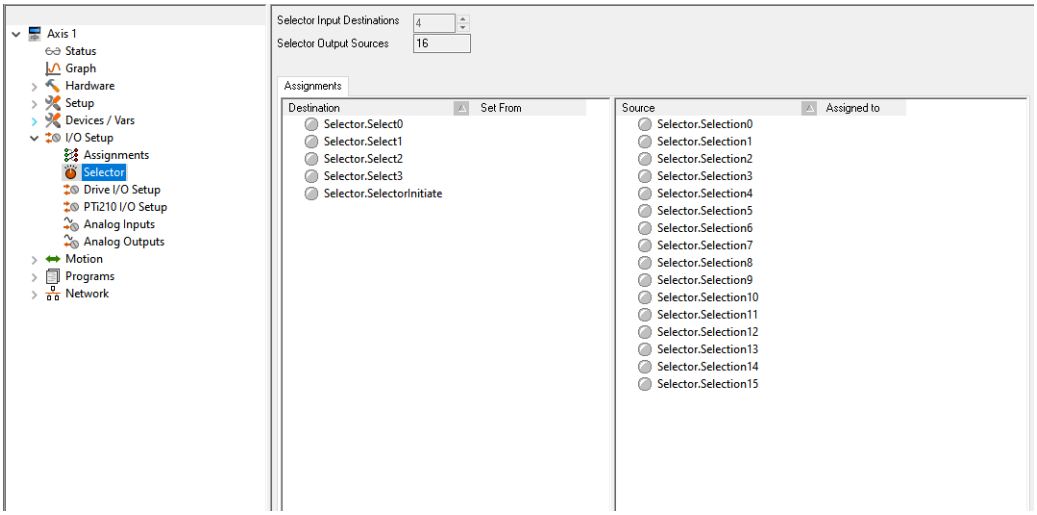


Figure 8-79: Selector View

The selector helps to minimize the required number of inputs and outputs to initiate different actions. The selector limits the I/O by using a conversion from binary to decimal. Figure 8-80 shows a block diagram of the Selector Object.

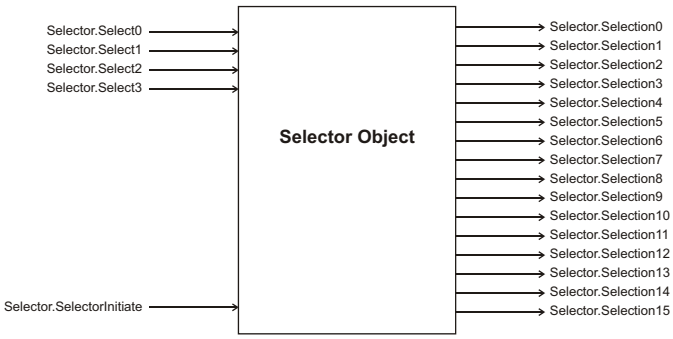


Figure 8-80: Selector Block Diagram

The Selector is configured by using a series of Sources and Destinations on the Assignments view. The inputs to the selector are **Selector.Select#** and **Selector.SelectorInitiate**. These inputs to the Selector can be found under the Selector group of Destinations on the Assignments view. The outputs from the selector are called Selector.Selection#, and can be found under the Selector group of Sources on the Assignments view.

In most cases, hardware inputs are assigned to the Selector.Select functions.

Based on the status of the binary select lines, a selector.selection source will be active when the Selector.SelectorInitiate destination is activated.

At the top of the Selector view (see Figure 8-79), the Selector Input Destinations field defines how many Select lines will be used. The number of Selector.Selection outputs is a direct result of the number of Select lines. The formula is as follows:

# of Selection outputs = 2^n (where n is the number of Select inputs)

The maximum number of Select lines is eight.

Once you have determined how many select lines you want, the assignments to these Selector.Select lines is then made in the Assignments view.

Example:

If the user enters 2 for the number of Selector Input Destinations, we would have 4 Selection lines (Selector.Selection0 through Selector.Selection3). The Selector.Selection number that activates is determined by the status of the Selector.Select lines when the Selector.SelectorInitiate bit is activated. Each select line has a specific binary value.

The binary value is determined as follows:

Sn x 2^n where Sn = Status of Selector.Select line n

Sn = 0 if Selector.Select line n is inactive, and

Sn = 1 if Selector.Select line n is active

The sum of all the binary values determines which Selector.Selection line will be active.

The following examples demonstrate how to determine which Selector.Selection will activate based on the Selector.Select lines.

Example:

If Selector.Select2 is active, Selector.Select1 is inactive, and Selector.Select0 is active, then the total binary value is as follows:

S2 = 1, S1 = 0, and S0 = 1. Therefore,

Total Binary Value = $(1 \times 2^2) + (0 \times 2^1) + (1 \times 2^0)$

Total Binary Value = 4 + 0 + 1

Total Binary Value = 5

Therefore, when Selector.SelectorInitiate activates, Selector.Selection5 will activate.

Example:

If Selector.Select2 is inactive, Selector.Select1 is active, and Selector.Select0 is active, then the total binary value would be as follows:

S2 = 0, S1 = 1, and S0 = 1. Therefore,

Total Binary Value = $(0 \times 2^2) + (1 \times 2^1) + (1 \times 2^0)$

Total Binary Value = 0 + 2 + 1

Total Binary Value = 3

Therefore, when Selector.SelectorInitiate activates, Selector.Selection3 will activate.

The Selector.Select lines can change without any action until the Selector.SelectorInitiate destination is activated.

Selector.Selection sources can be tied to any destination in the Assignments view. Figure 8-81 shows the four Selection lines being assigned to Index 0 through Index 3 Initiates, and the two Select lines being assigned to digital inputs. By doing this, we could initiate up to four indexes with only two Select lines and a Selector Initiate. This helps to minimize the number of inputs required to initiate a large number of indexes or programs.

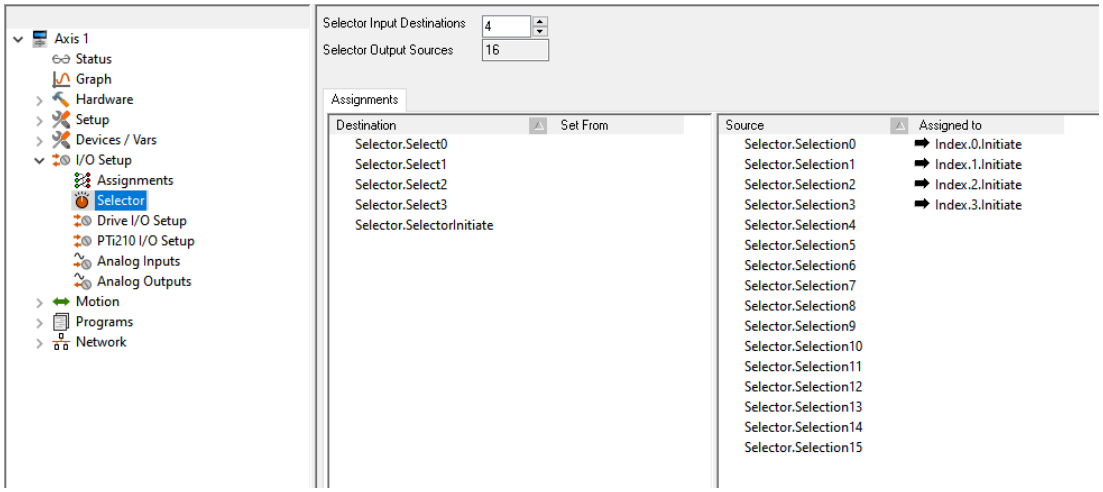


Figure 8-81: Selector Assignments Example

8.8.3 Drive I/O Setup View

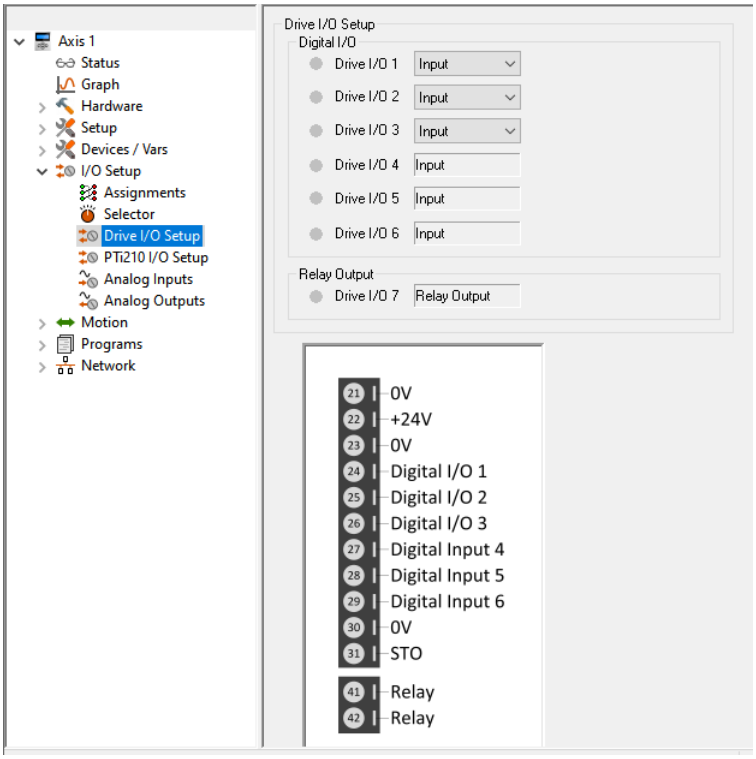


Figure 8-82: Unidrive M700/M701 Drive I/O Setup View

The Drive I/O Setup view allows the user to configure the digital I/O on the relevant drive. The contents of the Drive I/O Setup View will depend on the particular drive selected, Figure 8-82 shows the Drive I/O Setup for a Unidrive M700/M701. Use the list boxes to configure as desired. While online with PowerTools Studio, this view shows “Virtual LEDs” that are used to show the current status of the drive’s digital I/O.

8.8.4 PTi210 I/O Setup View

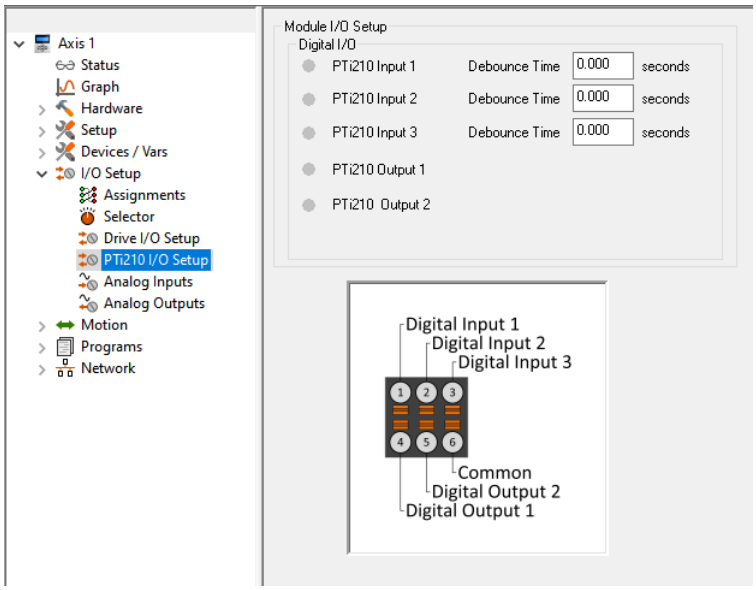


Figure 8-83: PTi210 I/O Setup View

The PTi210 I/O Setup view allows the user to configure the debounce time for each of the digital inputs on the PTi210 module. Debounce time is the minimum time that the given input must be active before it is accepted as a valid active signal. This can be used to reject momentary noise spikes in a electrically noisy environments. While online with PowerTools Studio, this view shows “Virtual LEDs” that are used to show the current status of the PTi210 module digital I/O.

8.8.5 Analog Inputs View

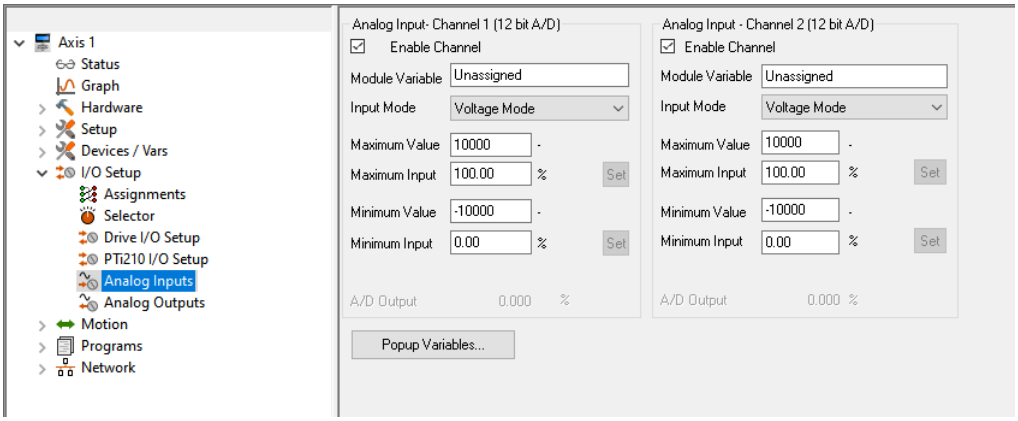


Figure 8-84: Analog Inputs View

Depending on the drive model, up to three Analog Input channels that can be used for a wide variety of applications. These Analog Inputs can be referred to as Analog Input 1, Analog Input 2, and Analog Input 3. By default, all three Analog Inputs are configured for Voltage Mode.

The Unidrive M702 has one terminal (terminal 8) which, by default, is configured as Digital Input 5, but from date code 1710 can be configured as a single-ended Analog Input 3 for use with a thermistor.

The Digitax HD has one differential analog input channel referred to as Analog Input 1.

For more information on the Analog Inputs not covered in this manual, please refer to the relevant drive *Control User Guide* or *Parameter Reference Guide*.

Following are descriptions of each of the setup parameters for Analog Input 1 and 2.

Analog Input - Channel 1

The first analog input on the Unidrive M700/M701 and Digitax HD is a medium-resolution (11-bit + sign) input.

Input Mode

Analog Input 1 on the Unidrive M700/M701 is capable of operating in multiple different modes so that it can work with different Analog circuits. The different types of analog circuits supported are as follows:

1. 4-20 mA Low - Drive Analog Input accepts an input value between 4 and 20 mA.
   4 mA corresponds to minimum scale (0.00 %), and 20 mA corresponds to full scale (100.00 %).
   For input values below 3 mA, the Analog Input level will be 0.00 %.
   No Current Loss trip is initiated.

2. 20-4 mA Low - Drive Analog Input accepts an input value between 4 and 20 mA.
   20 mA corresponds to minimum scale (0.00 %), and 4 mA corresponds to full scale (100.00 %).
   For input values below 3 mA, the Analog Input level will be 0.00 %.
   No Current Loss trip is initiated.

3. 4-20 mA Hold - Drive Analog Input accepts an input value between 4 and 20 mA.
   4 mA corresponds to minimum scale (0.00 %), and 20 mA corresponds to full scale (100.00 %).
   For input values below 3 mA, the Analog Input level will be held at the last value.
   No Current Loss trip is initiated.

4. 20-4 mA Hold - Drive Analog Input accepts an input value between 4 and 20 mA.
   20 mA corresponds to minimum scale (0.00 %), and 4 mA corresponds to full scale (100.00 %).
   For input values below 3 mA, the Analog Input level will be held at the last value.
   No Current Loss trip is initiated.

5. 0-20 mA - Drive Analog Input accepts an input value between 0 and 20 mA.
   0 mA corresponds to minimum scale (0.00 %), and 20 mA corresponds to full scale (100.00 %).
   No Current Loss trip is initiated.

6. 20-0 mA - Drive Analog Input accepts an input value between 0 and 20 mA.
   20 mA corresponds to minimum scale (0.00 %), and 0 mA corresponds to full scale (100.00 %).
   No Current Loss trip is initiated.

7. 4-20 mA Trip - Drive Analog Input accepts an input value between 4 and 20 mA.
   4 mA corresponds to minimum scale (0.00 %), and 20 mA corresponds to full scale (100.00 %).
   For input values below 3 mA, the Analog Input level will be 0.00 % and a 'An Input 1 Loss' trip is initiated.

8. 20-4 mA Trip - Drive Analog Input accepts an input value between 4 and 20 mA.
   20 mA corresponds to minimum scale (0.00 %), and 4 mA corresponds to full scale (100.00 %).
   For input values below 3 mA, the Analog Input level will be 0.00 % and a ‘An Input 1 Loss’ trip is initiated.

9. 4-20 mA - Drive Analog Input accepts an input value between 4 and 20 mA.
   4 mA corresponds to minimum scale (0.00 %), and 20 mA corresponds to full scale (100.00 %).
   For input values below 3 mA, the Analog Input level will be 0.00 %.
   No Current Loss trip is initiated.

10. 20-4 mA - Drive Analog Input accepts an input value between 4 and 20 mA.
    20 mA corresponds to minimum scale (0.00 %), and 4 mA corresponds to full scale (100.00 %).
    For input values below 3 mA, the Analog Input level will be 0.00 %.
    No Current Loss trip is initiated.

11. Volt - Drive Analog Input accepts an input value between 0 and 10 V.
    0 V corresponds to minimum scale (0.00 %), and 10 V corresponds to full scale (100.00 %).

The Unidrive M702 is not equipped with an Analog Input 1.

Analog Input 1 on the Digitax HD is only capable of operating in voltage mode as follows:

1. Volt - Drive Analog Input accepts an input value between 0 and 10 V.
   0 V corresponds to minimum scale (0.00 %), and 10 V corresponds to full scale (100.00 %).

Enable Channel Check Box

By default, the analog input channel is not enabled meaning that the PTi210 module will not read the A/D value read by the analog circuit. If the channel is not selected (disabled) the configuration parameters for the analog input are dim and unavailable. To enable the input, simply select the Enable Channel check box, and the configuration parameters will become available to edit.

If the user wishes to control the Analog Input through other means within the Unidrive M/Digitax HD, it is necessary to clear the Enable Channel check box.

Module Variable

The Module Variable parameter defines what PTi210 parameter will be controlled by Analog Input Channel 1. This means that the selected module parameter will automatically be populated with a value based on the Max and Min scaling values entered on this view.

NOTE

If assigning the Analog Input to the Jog.#.Vel parameter, it is recommended to change the global ramp type to "Linear". If the ramps are not set to linear, then the accel decel profiles can be infinite in length regardless of the value they are set to on the Jog view. The ramp type (or AccelType) can be found on the Ramps view.

Maximum Value

In order to scale the raw analog input value into the proper scale and units for the selected Module Variable, the user must define the maximum scale value and the minimum scale value. The PTi210 module then uses linear interpolation to scale the analog input and populate the specified Module Variable.

The Maximum Value is the value that the Module Variable will be set to when the Analog Input reaches the specified Maximum Input value.

Maximum Input

This defines the Maximum Input value in % that will yield the corresponding Maximum Value for the selected Module Variable. The Analog Input is not limited to this value, but this is used as a point on the “curve” for the linear interpolation equation.

Example:

If the Maximum Value is set to 5000, and the Maximum Input is set to 100 %, this means that when the Analog Input reaches 100 % of scale, the value of the selected Module Variable will be set to 5000.

Set Maximum Input Button

By pressing this button, the current value of the analog input will be read and stored in the Maximum Input text box. This is a simple tool that allows the user to set the analog input to full scale, and then easily read the value without requiring additional calibration. The button appears dim except when online with the PTi210 module.

Minimum Value

In order to scale the raw analog input value into the proper scale and units for the selected Module Variable, the user must define the maximum scale value and the minimum scale value. The PTi210 module then uses linear interpolation to scale the analog input and populate the specified Module Variable.

The Minimum Value is the value that the Module Variable will be set to when the Analog Input reaches the specified Minimum Input value.

Minimum Input

This defines the Minimum Input value in % that will yield the corresponding Minimum Value for the selected Module Variable. The Analog Input is not limited to this value, but this is used as a point on the “curve” for the linear interpolation equation.

Set Minimum Input Button

The ***Set*** button when pressed, reads the current value of the analog input and stores that value in the Minimum Input text box. This is a simple tool that allows the user to set the analog input to minimum scale, and then easily read the value without requiring additional calibration. The button appears dim except when online with the PTi210 module.

Example:

If the Minimum Value is set to -5000, and the Minimum Input is set to 0 %. This means that when the Analog Input reaches 0 % of scale, the value of the selected Module Variable will be equal to -5000.

A/D Output (Raw Value)

The Raw Value is a read-only parameter that while online with the PTi210 module will show the user the current value of the analog input in units of % of maximum scale. The range of this parameter is always 0 to 100 % regardless of the Maximum Input and Minimum Input values previously configured.

8.8.6 Analog Input - Channel 2

The second analog input on the Unidrive M700/M701 is a medium-resolution (11-bit + sign) input. All setup parameters for Analog Input 2 are the same as Analog Input 1.

For information on the parameters, please see Analog Input – Channel 1 above.

Input Mode

Analog Input 2 on the Unidrive M700/M701 is similar in operation and functionality to Analog Input 1, the only difference being that if the Input Mode is set to either 4-20 mA Trip or 20-4 mA Trip and the analog input value falls below 3 mA, this will result in the trip 'An Input 2 Loss'.

The Unidrive M702 and the Digitax HD are not equipped with an Analog Input 2.

Popup Variables Button

Once either of the Analog Input channels have been enabled the ***Popup Variables*** button becomes available. When the ***Popup Variables*** button is pressed the Select Variables From Tree window will open containing the list of variables that can be dragged into the Module Variable text box.

8.8.7 Analog Outputs View

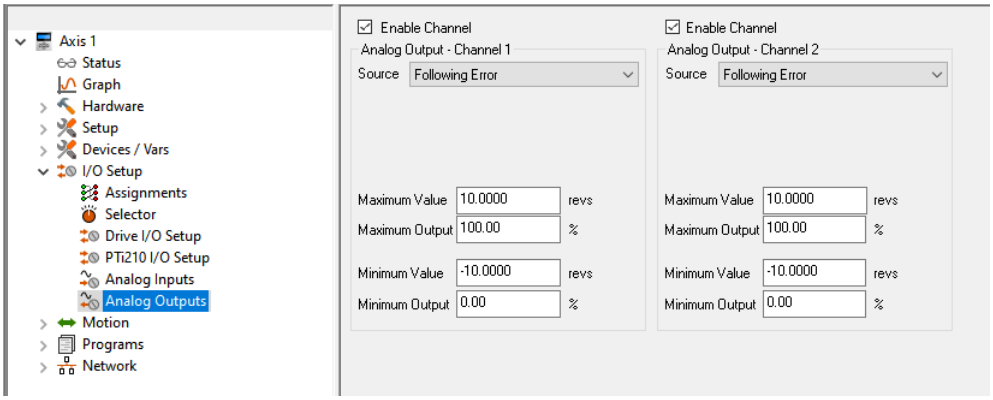


Figure 8-85: Analog Outputs View

The Unidrive M700/M701 has two Analog Output channels that can be used for a wide variety of applications. The Analog Outputs can be referred to as Analog Output 1 and Analog Output 2. Following are descriptions of each of the setup parameters for each Analog Output.

The Unidrive M702 and Digitax HD drives are not equipped with Analog Outputs.

For more information on the Analog Outputs that is not covered in this manual, please refer to the Unidrive M700/M701 *Control User Guide* or *Parameter Reference Guide*.

8.8.8 Analog Output – Channel 1

Enable Channel

By default, the analog output channel is not enabled meaning that the PTi210 module is not sending a value to the analog circuit. If the Enable Channel check box is clear (disabled), the configuration parameters for the analog output appear dim and are unavailable. To enable the output, simply select the Enable Channel check box, and the configuration parameters will become available for editing.

If the user wishes to control the Analog Output through other means within the Unidrive M700/M701, it is necessary to clear the Enable Channel check box.

Source

The user can create a direct connection from a Source parameter to the Analog Output. This means that the current value of the parameter selected as the Source will directly determine the value of the Analog Output signal. The Source list box contains a list of predefined parameters to select from. The list is as follows:

Following Error

If the user selects Following Error from the list of predefined parameters, the PTi210 module will constantly write the present value of the parameter FollowingError to the Analog Output. Following Error is defined as the difference between position command and the position feedback. Selecting Following Error from the list will automatically configure the scaling parameters to the correct units and default values. The units for FollowingError are User Distance Units.

User Defined Module Variable

If the user selects User Defined Module Variable from the list, then a text box titled Module Variable will appear where the user can enter any desired parameter from within the PTi210 module to control the Analog Output. For a list of available parameters, click ***Popup Variables*** button and the Select Variables From Tree window will open containing the list of variables that can be dragged into the Module Variable text box. Once the user enters a Module Variable, the units and resolution will change accordingly.

User Defined Drive Menu

If the user selects User Defined Drive Menu from the list, then text boxes titled Menu and Scale will appear. The user can enter any desired parameter from within the Unidrive M700/M701 to control the Analog Output. For a list of available parameters, refer to the Unidrive M700/M701 *User Guide* or *Parameter Reference Guide*.

Selecting User Defined Drive Menu from the list will also cause the Max and Min Value/Output parameters to disappear because the scaling capabilities of the drive parameters are minimized.

Velocity Command

If the user selects Velocity Command from the list of predefined parameters, the PTi210 module will constantly write the present value of the parameter VelCommand to the Analog Output. Velocity Command is defined as the velocity commanded by the PTi210 module to the drive, and is completely independent of feedback. Selecting Velocity Command from the list will automatically configure the scaling parameters to the correct units and default values. The units for VelCommand are User Distance Units/Unit Time.

Velocity Feedback

If the user selects Velocity Feedback from the list of predefined parameters, the PTi210 module will constantly write the present value of the parameter VelFeedback to the Analog Output. Velocity Feedback is defined as the velocity measured by the feedback device (i.e. encoder). Selecting Velocity Feedback from the list will automatically configure the scaling parameters to the correct units and default values. The units for VelFeedback are User Distance Units/Unit Time.

[Menu 3.02] Speed

Selecting this Source will write the Drive Speed Feedback (Pr **03.002**) to the Analog Output directly.

The units for this selection are always rpm regardless of the User Unit configuration.

This Source value is updated every 4 ms.

[Menu 4.02] Active Current (Iq)

Selecting this Source will write the Drive Active Current (Pr **04.002**) to the Analog Output directly.

The Active Curent is the torque producing current being created by the drive.

The units for this selection are Amps.

This Source value is updated every 250 µs.

[Menu 4.17] Reactive Current (Id)

Selecting this Source will write the Drive Reactive Current (Pr **04.017**) to the Analog Output directly.

The Reactive Current is also sometimes referred to as the Magnetizing Current, and is the current produced to generate the magnetic field or flux for an induction motor.

The units for this selection are Amps.

This Source value is updated every 250 µs.

[Menu 5.03] Output Power

Selecting this Source will write the Drive Output Power (Pr **05.003**) to the Analog Output directly.

The Output Power is the product of the Output Voltage and the Active Current.

The units for this selection are kW.

This Source value is updated every 4 ms.

Output Update Rate

The Analog Output will be updated every 250 microseconds or at the source parameter update rate if this is slower.

Module Variable

The Module Variable parameter is only available once the user has selected User Defined Module Variable from the Source list box. The field is used to define what PTi210 parameter will control the Analog Output. This means that the selected module parameter will directly determine the value of the Analog Output based on the Max and Min scaling values entered on this view.

Popup Variables Button

The ***Popup Variables*** button is only available once the user has selected User Defined Module Variable from the Source list box. Click ***Popup Variables*** and the Select Variables From Tree window will open containing the list of variables that can be dragged into the Module Variable text box.

Menu

The Menu parameter is only available once the user has selected User Defined Drive Menu from the Source list box. In this text box, the user enters the desired Unidrive M700/M701 menu parameter that will control the Analog Output. The parameter is entered in the format of **mm.ppp** where **mm** is the Menu number, and **ppp** is the Parameter number. The range for this parameter is 0.000 to 59.999. For more information on the Unidrive M parameter set, please refer to the relevant drive *Control User Guide* or *Parameter Reference Guide*.

Scale

The Scale parameter is only available once the user has selected User Defined Drive Menu from the Source list box. This parameter allows the user to enter a scaling factor that is multiplied with the value of the Menu parameter to determine the output value.
The formula to determine the actual output is as follows:

$$\left(\frac{\text{Parameter Value}}{\text{Parameter Max}}\right)^2 \text{Scaling Factor} = \text{Analog Output}$$

Maximum Value

This defines the Maximum Value of the Source parameter that will yield the corresponding Maximum Output for the Analog Output. The source parameter may be greater than this maximum value, however the Max Value and Max Output are used together to define a single point on the curve used for linear interpolation.

Maximum Output

In order to scale the Analog Output value into the proper scale and units for the selected Module Variable, the user must define the maximum scale value and the minimum scale value. The PTi210 module then uses linear interpolation to determine the value to be sent to the analog output.

The Maximum Output is the value that will be sent to the Analog Output when the Module Variable is equal to the Maximum Value.

Example:

If the Maximum Value is set to 5000, and the Maximum Output is set to 100 %, the Analog Output will be at full scale when the selected Module Parameter is equal to 5000.

Minimum Value

This defines the Minimum Value of the Source parameter that will yield the corresponding Minimum Output for the Analog Output. The source parameter may be less than this minimum value, however the Min Value and Min Output are used together to define a single point on the curve used for linear interpolation.

Minimum Output

In order to scale the Analog Output value into the proper scale and units for the selected Module Variable, the user must define the maximum scale value and the minimum scale value. The PTi210 module then uses linear interpolation to determine the value to be sent to the analog output.The Minimum Output is the value that will be sent to the Analog Output when the Module Variable is equal to the Minimum Value.

Example:

If the Minimum Value is set to -5000, and the Minimum Output is set to 15 %, the Analog Output will be at 15 % of full scale when the selected Module Parameter is equal to -5000.

Output Mode

The Analog Outputs on the Unidrive M700/M701 only support the Voltage Mode of operation.

1. Voltage Mode – In this mode, the Analog Output channel will send out a voltage signal ranging from 0 Volts (minimum scale) to 10.0 Volts (full scale).

8.8.9 Analog Output – Channel 2

See Analog Output – Channel 1.

8.9 Define Motion Profiles

8.9.1 Jog View

Figure 8-86: Jog View

Jog Number

This box allows you to select between Jog0 and Jog1 setup views.

Jog Name

This is a descriptive character string which can be assigned to the specific jog. Giving a name to a jog can make the motion setup easier to follow.

Time Base

This list box allows the user to select the time base for the individual jog. The options are Realtime and Synchronized.

Jog Velocity

This parameter specifies the target jog velocity for the individual Jog. The motor will run at this velocity when jogging with an assignment or through a program. This value is a signed number. The direction of the jog is determined by the sign of the jog velocity as well as using the Jog.PlusInitiate or the Jog.MinusInitiate.

Jog Acceleration

This is the acceleration ramp used when initiating this individual Jog. If S-Curve ramps are used, then this is the average acceleration rate for the entire ramp. The units for the acceleration are setup in the Setup - User Units view in PowerTools Studio.

Jog Deceleration

This is the deceleration ramp used when stopping this individual Jog. If S-Curve ramps are used, then this is the average deceleration rate for the entire ramp. The units for the deceleration are setup in the Setup - User Units view in PowerTools Studio.

Jog Sources and Destinations

Sources

Jog.AnyCommandComplete - The Jog.AnyCommandComplete source will activate when either Jog0 or Jog1 completes its deceleration ramp, and reaches zero commanded velocity. It will deactivate when any Jog is initiated again. If the Stop destination is used during a Jog, then the Jog.AnyCommandComplete will not activate.

Jog.#.Accelerating - This source is active while a jog is accelerating to its target velocity. Once the Jog reaches the target velocity, the Jog.#.Accelerating source will deactivate.

Jog.#.AtVel - This source activates when the individual jog reaches its target velocity. It deactivates when a jog deceleration ramp begins.

Jog.#.CommandInProgress - The Jog.#.CommandInProgress source is active throughout an entire jog profile. The source activates at the beginning of a jog acceleration ramp, and deactivates at the end of a jog deceleration ramp.

Jog.#.CommandComplete - The Jog.#.CommandComplete source will activate when the specific jog completes its deceleration ramp. It will remain active until the specific jog is initiated again. If the Stop destination is used during a Jog, then the Jog.#.CommandComplete will not activate.

Jog.#.Decelerating - This source is active while a jog is decelerating from its target velocity. Once the Jog reaches zero velocity (or its new target velocity), the Jog.#.Decelerating source will deactivate.

Destinations

The following destination functions can be found in the Assignments view under the I/O setup group:

Jog.PlusActivate - When this destination is activated, jogging motion will begin in the positive direction. The jog velocity is determined by which jog (Jog0 or Jog1) is active or not. A jog stops when this destination is deactivated. If the jog velocity is negative, Jog.PlusActivate will cause the motor to jog in the negative direction.

Jog.MinusActivate - When this destination is activated, jogging motion will begin in the negative direction. The jog velocity is determined by which jog (Jog0 or Jog1) is active or not. A jog stops when this destination is deactivated. If the jog velocity is negative, Jog.MinusActivate will cause the motor to jog in the positive direction.

Jog.Select0 - This destination is used to select between Jog0 and Jog1. When the Jog.Select0 destination is not active, the target velocity for the jog is the Jog.0.Velocity. If the Jog.Select0 destination is active, the target velocity of the jog is the Jog.1.Velocity. Jog.Select0 can be toggled "On" or "Off" while jogging. Jog acceleration and deceleration ramps are used to ramp between jog velocities.

Below is a description of jog operation using these destinations.

NOTE

In the table below Jog.0.Velocity = 100 rpm and Jog.1.Velocity = -500 rpm.

Jog.PlusActivate	Jog.MinusActivate	Jog.Select0	Motion
Off	Off	Off	0 rpm
On	Off	Off	+100 rpm
Off	On	Off	-100 rpm
On	Off	On	-500 rpm
Off	On	On	+500 rpm
On	On	Off	0 rpm
On	On	On	0 rpm

All Jog destinations are level sensitive.

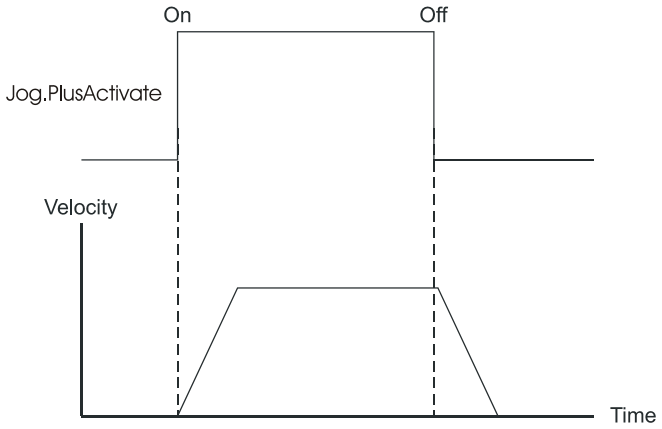


Figure 8-87: Jog Activate

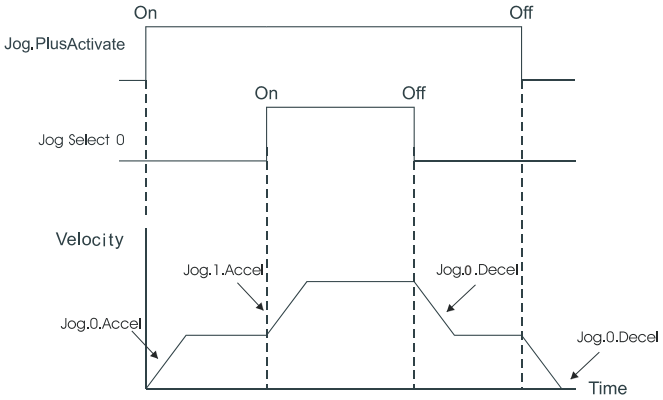


Figure 8-88: Jog Select Usage

If the Jog direction is reversed, the Jog.#.Decel value will be used to decelerate the motor to zero speed and then the Jog.#.Accel will be used to accelerate to the new (opposite sign) velocity.

NOTE

The Jog destinations cannot be initiated when any other motion type (homing, indexing, or programs) is in progress.

If both jog input functions are "On" there is no motion after a jog deceleration (they effectively cancel each other). The drive's display will show "Ready".

If the device is jogging with the Jog.PlusActivate destination active and the Jog.MinusActivate destination activates, the motor will behave the same as if it would if Jog.PlusActivate just deactivated.

The Stop destination (found under the Ramps group in the Assignments view) will override the Jog operation and decelerate the motor to zero speed at the stop deceleration rate.

Jog Motion can be used to jog off a Travel Limit Switch. Use the Jog Initiate for the opposite direction of the Limit Switch that is active.

8.9.2 Home View

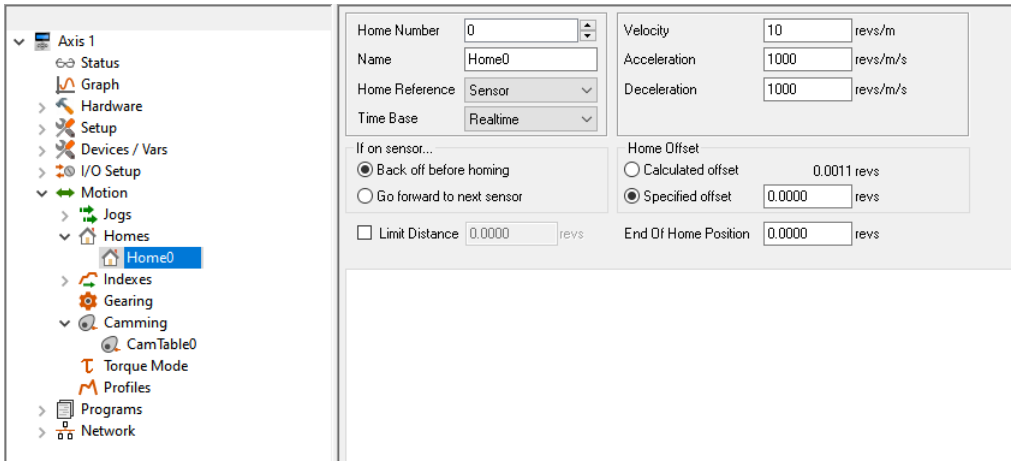


Figure 8-89: Home View

Home Number

The Home Number parameter displays which home sequence you are editing and allows you to scroll through multiple home sequences using the up and down arrows. The first release only allows for one home sequence.

Name

Allows you to assign a descriptive name to the home sequence up to 12 characters in length.

Home Reference

This parameter determines the signal used as the reference. The parameter can have one of three different values: Sensor, Marker, or Sensor then Marker. When the home reference is Sensor the rising edge of the Home.#.SensorTrigger destination is used to establish the home position. When the home reference is Marker, the rising edge of the motor encoder's marker channel is used to establish the home position. When the home reference is Sensor then Marker, the home position is established using the first marker rising edge after the Home.#.SensorTrigger destination activates.

Time Base

Selects the Time Base for the home move velocity and acceleration/deceleration. Realtime and Synchronized are the allowed selections.

Velocity

Sets the target velocity for the home. The polarity determines the home direction. Positive numbers cause motion in the positive direction and negative numbers cause motion in the negative direction in search of the home reference.

Acceleration

Average Acceleration rate used during the home. Units are specified on the User Units view.

Deceleration

This is the average Deceleration rate used at the end of the Home move in user units.

If on sensor... Group

These radio buttons determine how the system reacts if the Home.#.SensorTrigger is already active when the home is initiated.

Back off before homing Radio Button

If this radio button is selected, the drive will back off the sensor before beginning the home. It does this by moving the direction opposite to that specified by the sign of the home velocity. It continues moving in this direction at the target home velocity until the sensor deactivates. The motor then decelerates to a stop and performs a standard home.

Go forward to next sensor Radio Button

If this radio button is selected, then the system will ignore the sensor that is active when the home is initiated, and move in the proper direction until the first low to high transition of the Home Reference signal.

Home Offset Group

The Home Offset group has two buttons, the Calculated Offset Radio Button and the Specified Offset radio button.

Calculated offset Radio Button

The calculated offset is defined as the distance traveled during deceleration ramp from the home velocity to a stop plus the distance traveled at the home velocity for 1600 µs. This extra distance is used to guarantee that the motor will not need to backup after the deceleration ramp.

Specified offset Radio Button

The specified offset allows the user to choose an exact offset from the Home Reference point.

The commanded motion will stop at exactly the offset distance away from the reference point as specified. If the specified offset is smaller than the calculated offset, the motor will decelerate to a stop and then back up to its final offset position.

Limit Distance Check Box

This check box when selected enables the Home Limit Distance.

Limit Distance

The Limit Distance parameter places an upper limit on the incremental distance traveled during a home. If no home reference is found in this distance, the motor will decelerate to a stop at the limit distance and activate the Home.#.LimitDistHit source.

End of Home Position

This parameter defines the position at the completion of the home. This defaults to 0.0 such that at the end of a home, the Feedback Position and the Commanded Position are set to 0.0.

If you wish your Feedback Position to be something other than 0.0 at the end of a home, then enter the exact desired position here.

For more details on how homing works see *Home* on page 31

Home Sources and Destinations

Sources

Home.AbsolutePosnValid - This source is activated when a Home is successfully completed. It indicates that the device has been homed properly. It is will be deactivated by the Home.#.Initiate destination, an encoder fault, a reboot, or when the device is powered down, unless using Auxiliary Logic Supply (ALP).

Home.AnyCommandComplete - This source is activated when any home motion command is completed. If a drive stop destination is activated before the home has completed, this source will not activate. It will be deactivated when another home is initiated.

Home.#.Accelerating - This source is active while a home is accelerating to its target velocity. Once the home reaches the target velocity, the Home.#.Accelerating source will deactivate. This source will also activate during the "back off sensor" motion before the actual home.

Home.#.AtVel - This source activates when the home reaches its target velocity. It deactivates when a home deceleration ramp begins. Home.#.AtVel will not be activated during the "back off sensor" portion of the home.

Home.#.CommandComplete - The Home.#.CommandComplete source will activate when the specific home completes its deceleration ramp. It will remain active until the specific home is initiated again. If the drive stop destination is used during a home, then the Home.#.CommandComplete will not activate.

Home.#.CommandInProgress - Activated when the Home is initiated and remains active until all motion related to the Home has completed.

Home.#.Decelerating - This source is active while a home is decelerating from its target velocity. Once the home reaches zero velocity (or its new target velocity), the Home.#.Decelerating source will deactivate. This source will also activate during the "back off sensor" motion before the actual home.

Home.#.LimitDistHit - This source is activated when the home reference is not found before the Home Limit Distance is traveled. It will remain active until the home is initiated again.

Destinations

Home.#.Initiate - The Home.#.Initiate destination is used to initiate the home function. The Home is initiated on the rising edge of this function. The device will not initiate a Home if there is an Index, Jog, or Program in progress, or if the Stop destination is active or if a travel limit is active.

Home.#.SensorTrigger - This destination is required to be used if you are homing to a sensor. This destination is edge sensitive. The home position is determined when the Home Sensor destination is activated.

If the device receives a Home.#.Initiate input while the Home.#.SensorTrigger is active, you can choose to have the motor "back-off" of the home sensor before it initiates the home function, or move forward to the next sensor.

If debounce is used on the hardware input that the Home.#.SensorTrigger is assigned to, the debounce determines the length of time the input must be active to be considered a valid input.

The rising edge of the sensor is still used for the reference position. This maintains accuracy while providing the ability to ignore false inputs.

8.9.3 Index View

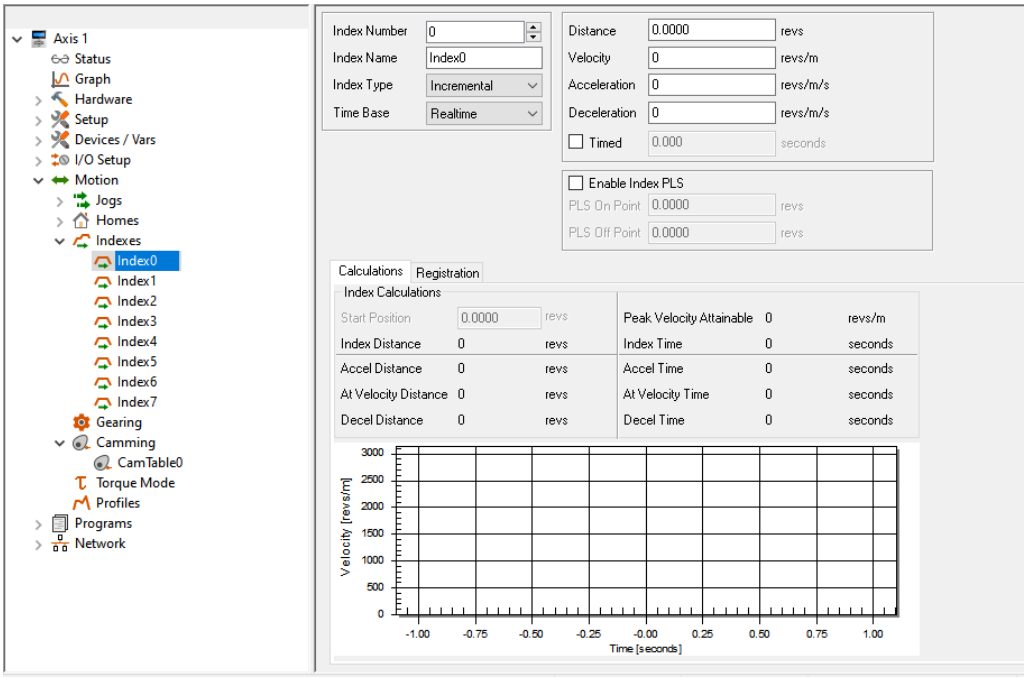


Figure 8-90: Index View

Following is a list of parameters used to configure each index.

Index Number

The Index Number parameter selects the index number with a scroll box.

Index Name

The User can specify an Index name of up to 12 alphanumeric characters. This allows assigning a descriptive name to each index indicating different machine operations.

IndexType

Select the index type from Incremental, Absolute, Correction, Posn Tracker Cont., Posn Tracker Once, Registration, Rotary Plus, or Rotary Minus.

Click the down arrow on the parameter list box to select the desired type of Index Profiles, as follows:

Incremental Indexes run a specified distance from the current position.

Absolute Indexes move to an exact position with respect to the home reference point. The absolute index could run in either a clockwise (CW) or counterclockwise (CCW) direction dependent on the current position when it is initiated.

Correction Indexes are used to follow a dynamic fieldbus or analog value that changes the index distance of the index prior to and during the index motion. Correction indexes use incremental distance values. The index 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. See Analog Input view.

Posn 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.

A Registration Index runs at the specified velocity until a registration sensor is seen or until it reaches the Registration Limit Distance. If a Registration Sensor is seen, then the index runs an additional Registration Offset distance.

Rotary Plus and Rotary Minus type indexes are typically used in applications which use rotary rollover. These absolute indexes are forced to run in a specific direction regardless of the starting point.

TimeBase

This list box selects the Time Base for the index velocity and acceleration/deceleration. Realtime and Synchronized are the available selections.

Distance/Position

The Distance/Position parameter is a signed value that specifies the distance the index will travel (incremental index) or the absolute position the index will move to (absolute index). In the case of an incremental index, this parameter also determines the direction the index will travel. If an index type of Registration is selected, then this is a limit distance, or the maximum distance the index will travel if a registration sensor is not seen.

Velocity

This sets the target velocity for the index profile. The velocity parameter is unsigned and must be greater than zero. Direction of the index is not determined by the velocity, but by the Distance/Position parameter.

Acceleration

Average Acceleration rate used during the index. Units are specified on the User Units view.

Deceleration

The Deceleration parameter specifies the deceleration value to be used during the index in user units.

Timed Indexes

A Timed Index allows the user to specify the amount of time in which to perform an index rather than specifying the Velocity, Acceleration, and Deceleration. The processor in the PTi210 module will automatically calculate the necessary velocity, accel, and decel in order to achieve the programmed distance in the specified time.

NOTE

A user program cannot compound into a Timed Index, or compound out of a Timed Index.

All index types can be specified as a Timed Index, except for Registration type indexes. This is because a registration index does not have a specified distance or absolute position. During a registration type index, the registration sensor could activate at any time, and therefore it is impossible to calculate the necessary velocity, accel, and decel. If Registration type is selected, then the Timed check box will become disabled.

Based on the Distance entered (or Position for Absolute indexes) and the Timed value specified, the calculations could result in extremely high Velocities, Accels, and Decels. To avoid damage to mechanical parts, or potentially dangerous situations, the user is allowed to enter the Maximum Velocity, Acceleration, and Deceleration used for the calculations. The results of the firmware calculations will never exceed the maximum values specified.

Figure 8-91 shows a screen capture in which the Timed check box has been enabled. Notice how the parameters that normally say Velocity, Acceleration, and Deceleration have changed to say Max. Velocity, Max. Accel, and Max. Decel. When the Timed check box is selected, these parameters automatically become maximums for use in the calculations.

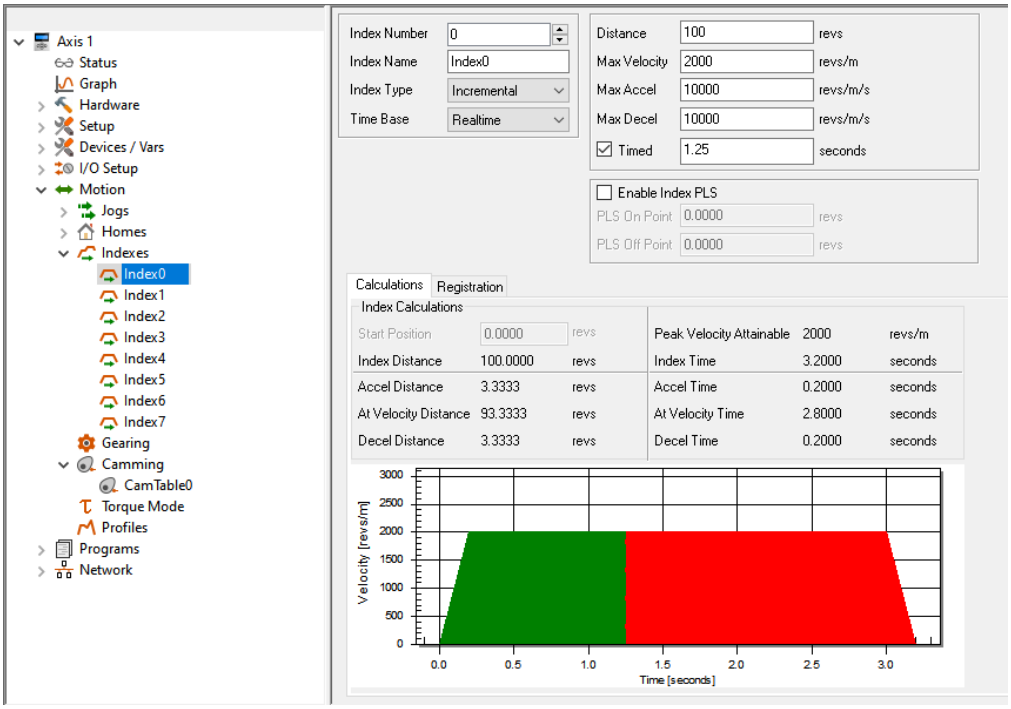


Figure 8-91: Index View with Timed Index Selected

If the values for Max.Velocity, Max.Acceleration, and Max.Deceleration are such that the distance cannot be covered in the specified time, the Index.ProfileLimited flag will activate when the index is initiated, indicating the index cannot be performed as desired. In this case the Calculations graph will highlight in red the area of the profile which cannot be performed in the specified time. The internal calculation are performed only when the index is initiated, and therefore is the only time the flag will activate. The Index.ProfileLimited flag will remain active until cleared using the Index.ResetProfileLimited assignment or program instruction. In this situation, the index will still operate, but the time will be extended. In other words, the profile will be performed using the maximums values and still cover the specified distance, but not in the specified time.

The units for the Timed parameter depend on the current setting of the Time Base parameter.

If Time Base is set to "Realtime" (default), then the units for the Timed parameter are Seconds. The user can program the index time with resolution of 0.001 Seconds (or milliseconds). If Time Base is set to "Synchronized", the units for the Timed parameter are defined by the Master Distance Units found on the Master Setup screen.

Doing a synchronized Timed Index means that the user can specify the master distance in which the index should be performed. This can be very useful in many synchronized motion applications.

The internal calculations are designed to calculate a triangular profile (all accel and decel). The ratio of acceleration to deceleration will be the same ratio as Max. Acceleration to Max. Deceleration parameters. For example, if the deceleration is desired to be twice the acceleration, a number twice the value of max acceleration would be entered for maximum deceleration. Even in trapezoidal moves, the same ratio of acceleration and deceleration is maintained.

The calculations are based on the assumption that Feedrate Override is set to 100 %. If set to greater that 100 %, the motor could run in excess of the specified Max. Velocity.

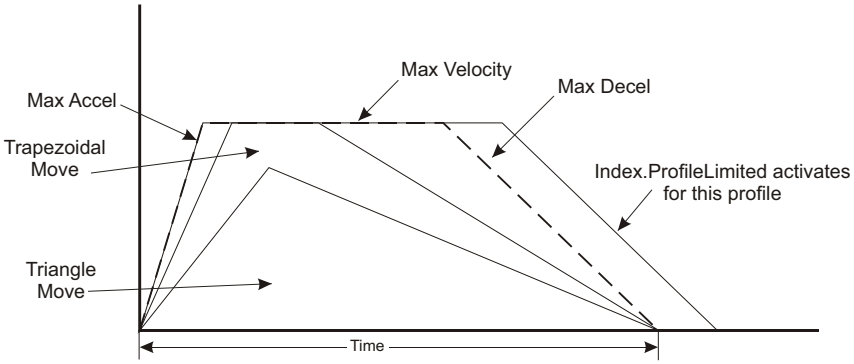


Figure 8-92: Timed Index Profile Diagram

Enable Index PLS Check Box

This check box enables (when selected) or disables (when clear) the Index PLS function.

An Index PLS is similar to a global PLS (explained in the PLS View section), but is incremental in nature. The Index PLS has On and Off points just like a global PLS, but the On and Off points are specified as an incremental distance from the start of the index, instead of absolute positions. Each index has its own On and Off points, and the Index.#.PLSStatus is only updated when Index# is run. The direction of the PLS does not matter, the Index.#.PLSStatus will activate and deactivate the same incremental distance from the start of the index.

PLS On Point

This parameter is an incremental distance from the start position of the index, at which the PLS.#.Status will become active. It is an unsigned value in user units. The On Point must always be less than the Off Point.

PLS Off Point

This parameter is an incremental distance from the start position of the index, at which the PLS.#.Status will deactivate. It is an unsigned value in user units. The Off Point must always be greater than the On Point. If the Off Point is larger than the Distance parameter in an Incremental type of index, the PLS Status will never deactivate until the index is run again.

Examples:

Example 1:

Index 0 is an Incremental index with a distance of 5 Revs. The PLS On Point is set to 1 Rev, and the PLS Off Point is set to 4 Revs. A home is completed, and Position Feedback is equal to 0.0 Revs.

If Index 0 is run, the Index.0.PLSStatus will activate when the feedback position reaches 1 Rev and remain active until feedback position reaches 4 Revs, and deactivate. At the end of Index 0, position feedback is equal to 5 Revs. If we initiate Index 0 again, Index.0.PLSStatus will activate 1 Rev into the index, or at 6 Revs. It will remain active until position feedback reaches 9 Revs, and deactivate. This index could be run over and over again, and Index.0.PLSStatus will activate 1 Rev from the starting position and deactivate 4 Revs from the starting position every time.

Example 2:

Index 1 is an Incremental index with a distance of -10 revs. The PLS On Point is set to 4 Revs, and the PLS Off Point is set to 6 Revs. A home is completed, and Position Feedback is equal to 0.0 Revs.

If Index 1 is run, the Index.1.PLSStatus will activate when the position feedback reaches -4 Revs (or 4 Revs from the start of the index). Index.1.PLSStatus will then deactivate when position feedback reaches -6 Revs (or 6 Revs from the start of the index). If Index 1 is run again, Index.1.PLSStatus will activate and deactivate at -14 Revs and -16 Revs respectively.

Index PLSs can be used on any type of an index.

If an index is so short (possible in the case of an absolute index) that it reaches the On Point, or incremental distance, into the index, but never reaches the Off Point, the Index.#.PLSStatus will remain active until the index is run again.

Similarly, if the index is so short that it never reaches the On Point, the Index.#.PLSStatus will never activate.

Registration Tab Parameters

The following parameters are only used if Registration is selected as the Index Type.

Analog or Sensor Radio Buttons

Select one of these radio buttons to determine what signal will be used as your registration trigger.

If Sensor is selected, a source must be assigned to the Index.#.SensorTrigger Typically a proximity sensor is wired to a hardware input, and therefore a module or drive input source is assigned to the Index.#.SensorTrigger, but any source can be used.

If Analog is selected, one of the analog signals must be selected in the analog list box. Available selections are Analog In Ch1, Analog In Ch2, or Current Feedback. Then a comparison operator must be selected from the operator list box. Available selections are > (greater than) and < (less than). Last, an analog value must be entered for comparison.

Registration to Analog Input Value

If Analog is selected, the value of the drive Analog Input is used as the registration signal.

When the value of the analog input reaches a value that satisfies the comparison operator, the sensor trigger will activate. Units for the registration value will match the units configured on the Analog Inputs screen when Analog In is selected.

Registration Offset

The incremental distance the motor will travel after a valid registration sensor or analog limit value has been detected. This is a signed parameter; so if an index is travelling in the negative direction, the offset needs to be negative and continue in the same direction. If the registration offset is zero or less than the decel distance shown on the calculations tab, the motor will decelerate at the programmed rate and then back up to the specified offset distance from the trigger position.

Enable Registration Window Check Box

This check box enables (if selected) the Registration Sensor Valid Window. When active, only registration marks that occur inside the registration window are seen as valid.

Window Start

This parameter defines the start of the Registration Sensor Valid Window relative to start position of this index. This is an unsigned value and is relative only to starting position of this index. Index direction does not affect this parameter. The Registration Window Start position (or distance) should be less than the Registration Window End position. If a registration sensor is seen outside of this window (not between the WindowStart and WindowEnd positions) then it will be ignored.

Window End

This parameter defines the end of the Registration Sensor Valid Window relative to start position of this index. This is an unsigned value and is relative only to starting position of this index. Index direction does not affect this parameter. The Registration Window End position (or distance) should be greater than the Registration Window Start position. If a registration sensor is seen outside of this window (not between the WindowStart and WindowEnd positions) then it will be ignored.

Example:

Index 0 is defined as a Registration type of index. The user wants the index to run at velocity for 10 Revs, or until the Torque Feedback reaches 50 % continuous torque and then continue for another 0.5 Revs.

In the Limit Distance parameter, enter 10.0.

On the registration tab, select the Analog radio button.

In the analog list box, select Torque Command

In the comparison operator list box, select ">"

In the analog value parameter, enter 50 (Units are established on the User Units view)

In the Registration Offset parameter, enter 1.5

This index would accelerate up to its' target velocity, and run at speed until one of the following:

The Limit Distance is approaching, and the index decels down to zero velocity, completing the move at the Limit Distance. At this point, the Index.#.LimitDistHit source would activate.

Or,

The Torque Command reaches or exceeds 50 % continuous, and the index continues at speed before decelerating to zero velocity at the registration point plus the Registration Offset distance.

If the Registration Offset distance is in the opposite direction from the move, or is so short that the motor cannot stop in the specified distance at the programmed deceleration rate, the motor will decelerate with the programmed ramp, and then back-up to the specified position (registration point + the Registration Offset).

Index Sources and Destinations

Sources

Index.AnyCommandComplete - Active when any index motion command is completed. If a stop is activated before the index has completed, this destination will not activate. Deactivated when any new index command is initiated.

Index.#.Accelerating - This source is active while an index is accelerating to its target velocity. Once the index reaches the target velocity, or begins to decelerate, the Index.#.Accelerating source will deactivate.

Index.#.AtVel - This source activates when the target index velocity is reached. If Feedrate override is changed or FeedHold is activated AtVelocity shall remain active. Index.#.AtVel will deactivate at the start of any deceleration or acceleration. During a synchronized index, this source could be active even without any motor motion if the master axis stops.

Index.#.Command Complete - The Index.#.CommandComplete source will activate when the specific index completes its deceleration ramp. It will remain active until the specific index is initiated again. If the drive stop destination is used during an Index, then the Index.#.CommandComplete will not activate.

Index.#.Command In Progress - The Index.#.CommandInProgress source is active throughout an entire index profile. The source activates at the beginning of the index acceleration ramp, and deactivates at the end of the index deceleration ramp. During a synchronized index, this source could be active even without any motor motion if the master axis stops.

Index.#.Decelerating - This source is active while an index is decelerating from its target velocity. Once the index reaches zero velocity, or its next target velocity, the Index.#.Decelerating source will deactivate.

Index.#.LimitDistHit - Activated when the registration sensor is not found before the Limit Distance is traveled. If the Registration Window is enabled the sensor must be activated inside the window to be recognized.

Index.#.PLSStatus - Controlled by the PLSOn and PLSOff Points which are relative to the distance commanded since the start of the index. Activated when index distance command is in between the PLSOn point and PLSOff points.

Index.ProfileLimited - For timed indexes, if the values for Max. Velocity, Max. Acceleration, and Max. Deceleration are such that the distance cannot be covered in the specified time, the Index.ProfileLimited flag will activate when the index is initiated, indicating that the index cannot be performed as desired. The Index.ProfileLimited flag will remain active until cleared using the Index.ResetProfileLimited assignment or program instruction. In this situation, the index will still operate, but the time will be extended. In other words, the profile will be performed using the maximums values and still cover the specified distance, but not in the specified time.

Destinations

Index.ResetProfileLimited - If a timed index was not able to complete in the specified time, the Index.ProfileLimited source will activate. Index.ResetProfileLimited is used to clear the ProfileLimited flag and acknowledge that the index did not complete in the specified time. This can be activated through an assignment, or through a user program. This function is edge-sensitive, so holding it active will not prevent ProfileLimited from activating.

Index.#.Initiate - The Index.#.Initiate destination is used to initiate the specific index. The Index is initiated on the rising edge of this function. An Index cannot be initiated if there is an Home, Jog, or Program in progress, or if the Stop destination or if a travel limit is active. It can be activated from an assignment or from a program.

Index.#.Sensor Trigger - If registration to Sensor is selected, when this destination activates, motor position is captured and is used as the registration point for registration type indexes.

Adding and Deleting Indexes

Adding or removing indexes from the user configuration can be done in three ways. Indexes may only be added or deleted while offline.

Adding an Index

Toolbar Button Method

The ***Add Index*** button will add a new index to the user configuration. Indexes are added in sequential order. Clicking on the button will add an index and the Index view will be displayed allowing the user to enter the index parameters.

PowerTools Studio Menu Bar Method

From the PowerTools Studio menu bar, select **Edit** > **New** > **Index**. An index will be added in sequential order and the Index view will be displayed allowing the user to enter the index parameters.

Right Click Method

Navigate to the Indexes View. Position the mouse pointer in the right side of the view and right-click the mouse. A selection menu will appear allowing the user to add a New Index or Delete an Index. Click New Index and an index will be added in sequential order and the Index view will be displayed allowing the user to enter the index parameters.

Deleting an Index

Toolbar Button Method

The ***Delete Index*** button will delete an index from the user configuration. The highest numbered index will automatically be deleted unless a different index is selected on the Indexes heading screen. To delete a specific index, click on the Motion > Indexes branch in the hierarchy tree. From this view, select the specific Index to be deleted, and then click on the ***Delete Index*** button.

PowerTools Studio Menu Bar Method

Navigate to the Indexes View, and select the Index to be deleted. From the PowerTools Studio menu bar, select **Edit** > **Delete** > **Index**. The selected Index will be deleted from the configuration.

Right Click Method

Navigate to the Indexes View. Select the Index to be deleted, and then right-click the mouse. A selection menu will appear allowing the user to add a New Index or Delete an Index. Click on Delete Index and the selected index will be deleted from the configuration.

8.9.4 Gearing View

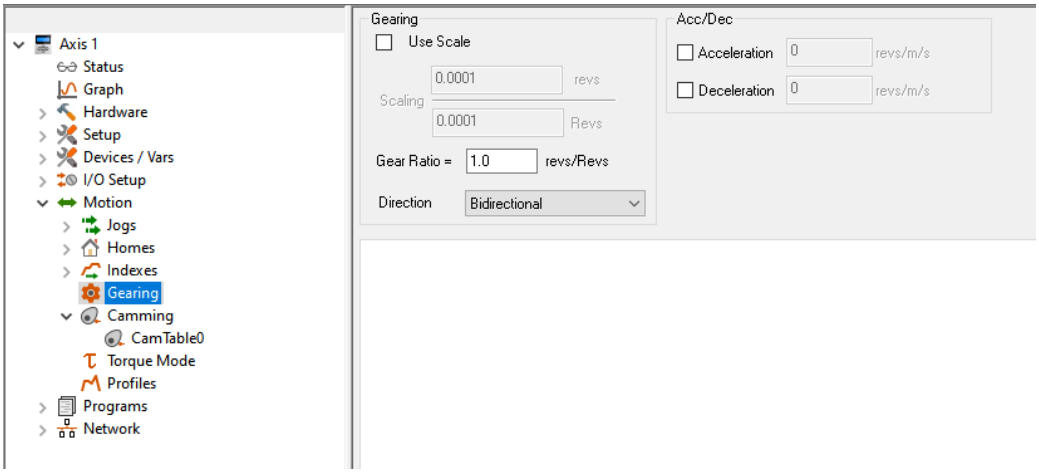


Figure 8-93: Gearing View

The following parameters are used to configure Gearing motion:

Use Scale Check Box

Select this check box (Gear.UseScaleEnable) to enable the Scaling feature and disable Gear Ratio. Scaling allows the user to use irrational ratios such as 1.0/7.0 that were not possible with the single parameter ratio (Gear.Ratio). Scaling is defined as follows:

$$\text{Scaling} = \frac{\text{Gear.ScaleNumerator (User Units)}}{\text{Gear.ScaleDenominator (Master Units)}}$$

Gear Ratio

Gearing is used to fix the motion of the motor to the motion of the master axis signal at a specified ratio. This is commonly called “electronic line shafting” or “electronic gearing”. To gear the motor to the master axis, a ratio must be specified as a relationship between follower distance units and master distance units. The ratio is as follows:

$$\text{Gear Ratio} = \frac{\text{\# of Follower Distance Units}}{\text{1 Master Distance Unit}}$$

The ratio (Gear.Ratio) is defined as the number of follower distance units to move the motor per distance unit of travel. The gear ratio can be positive or negative and is a signed 32-bit parameter. The resolution of the parameter is determined by the number of decimal places configured for the Master Velocity Units on the Master Units Setup view.

By default, gearing does not use acceleration or deceleration ramps with respect to the master encoder. This means that once gearing is activated, peak torque is available to try to achieve the specified gear ratio. Therefore, if the master axis is already in motion when gearing is activated, the control loop will attempt to accelerate the motor to the programmed ratio within one update (800 µsec $\leq$ update rate $\leq$ 1600 µsec). Analogously, when gearing is deactivated, the motor will use peak torque to bring the motor to a stop without a deceleration ramp.

Direction

Bidirectional

The follower will follow both the plus and minus master axis command at the specified ratio.

ComMinus

The follower will follow only the minus master axis command.

ComPlus

The follower will follow only the plus master axis command.

Acceleration Enable

When this check box is selected it allows a gear to run a specified accel ramp after the gearing command is turned On.

Acceleration

This parameter sets the acceleration of the realtime gearing ramp. Gear.Accel units are in Follower Units/Velocity Time Base/ Acceleration Time Base. The Gear.Accel functions only when the follower is ramping its speed up to meet the Master’s at the specified Gear.Ratio.

Deceleration Enable

When this check box is selected it allows a gear to run a specified decel ramp after the gearing command is turned Off.

Deceleration

This parameter sets the deceleration of the realtime gearing ramp. Gear.Decel units are Follower Units/Velocity Time Base/Acceleration Time Vase. The Gear.Decel functions only when the follower is ramping its speed down meet the Master’s at the specified Gear.Ratio.

8.9.5 Camming View

Electronic cams provide a non-linear motion function for a single axis. The basic motion can best be illustrated in Figure 8-94 of a mechanical cam and cam follower (or slave).

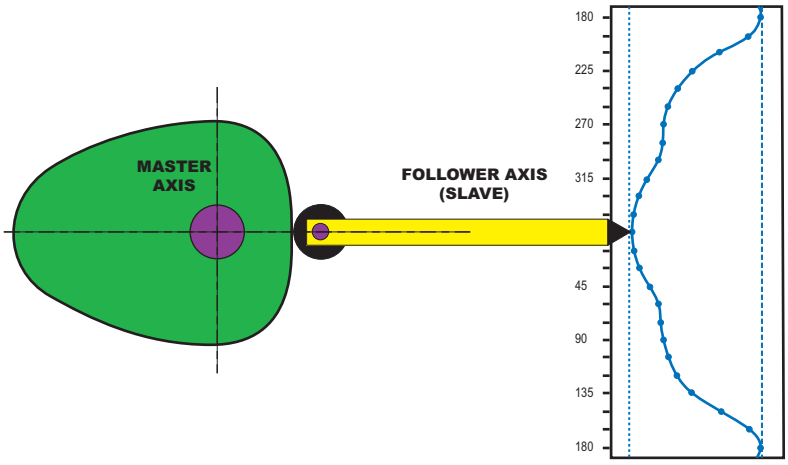


Figure 8-94: Mechanical Master and Slave Follower

As the master axis (the cam lobe) rotates, the follower axis produces a non-linear motion profile. This same profile can then be produced with a single motor driving a linear axis programmed with an electronic cam.

The cam motion object uses a master/follower principle in a synchronized mode and also has a follower with Realtime mode that allows the follower to travel through its cam table without a physical master axis moving.

Control Techniques provides a Cam as a collection of cam table(s) that can be used individually of chained together to form a full sequence of motion. Each cam table is a user specific sequence of movements whereby the user can specify the master and follower movement along with the interpolation type. Coupled with a user program to monitor the flow, the motion can dynamically be altered by changing the cam table chains selecting a different sequence of tables. You can further adjust the flow by dynamically changing the cam tables themselves or using a cam table time base index to adjust time or distance.

As an alternative, the cam is initiated in the same manner as jogs, home and indexes.

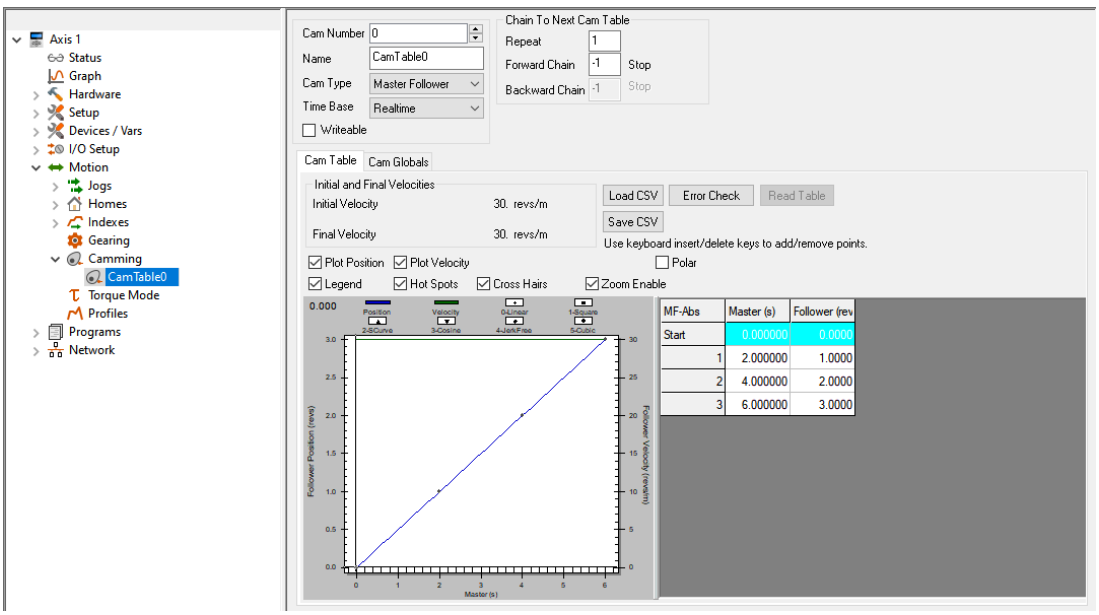


Figure 8-95: Camming View

Cam Number

Use the arrows of the scroll box to increment or decrement the Cam Number and view the setup information for that Cam number. A maximum of 32 different cam tables can be created.

Name

The user can specify a cam name (Cam.#.Name) of up to 12 alphanumeric characters. This allows assigning a descriptive name to each cam table indicating different machine operations.

Cam Type

Cam type (Cam.#.CamType) allow the user to choose from; Master Follower, Absolute MFI, Incremental MFI, Cubic Spline, or Time Based Index cams. Most data entries are “Absolute” which means each point is an absolute distance from the start of the cam table which is an implied zero, although the starting value does not have to be zero. The Incremental MFI (Master/Follower/Interpolation) has the entries as distance deltas from point to point which means the point is relative to the previous point.

Master Follower

Master Follower is an Absolute Master Follower with a fixed interpolation type of Linear for each point. This is convenient when there are many points entered or are importing data from a CAD system. A large number of data points are required for smooth motion.

The maximum number of data points in Master Follower mode is 8154.

Absolute MFI

Absolute MFI allows a different master, follower, and interpolation for each point in the cam. The values are in reference to the beginning of the cam table (position zero). The master positions must increase from point to point since change in master position must always be increasing.

The position interpolation types that are valid in this mode are:

Linear – Constant velocity across the complete point.

Square – Velocity increases or decreases linearly across the point. This means that the position changes quadratic across the point.

S-curve - The velocity and position change along a sinusoidal shape across the point

Cosine – The velocity starts and ends at the same velocity, but increases or decreases along a sinusoidal shape in order that the proper final position is achieved

Jerk Free – The jerk starts and ends at zero. Jerk increases or decreases in smooth transition.

The maximum number of data points in Absolute MFI mode is 6523.

Incremental MFI

Incremental MFI is different from Absolute MFI in that each point is a delta position. This allows values in the middle to be modified.

The maximum number of data points in Incremental MFI mode is 6523.

Cubic Spline

Cubic Spline uses the third order polynomial splining to find smooth velocity and position curves from point to point. This table allows a different master, follower, and interpolation for each point in the cam.

The maximum number of data points in Cubic Spline mode is 6523.

The valid interpolation types in this mode are:

Linear - Linear velocity across the complete point.

Cubic - A calculated third degree polynomial across the point such that acceleration is continuous moving from point to point.

Time Based Index

This is very similar to an index in time mode. By building it into the cam system, it is able to take advantage of such things as bidirectional movement, chaining, initial and final velocity definition.

The valid interpolation types allowed for this cam table are:

Square, Jerk Free, and S-curve.

Cam Table Plot Error

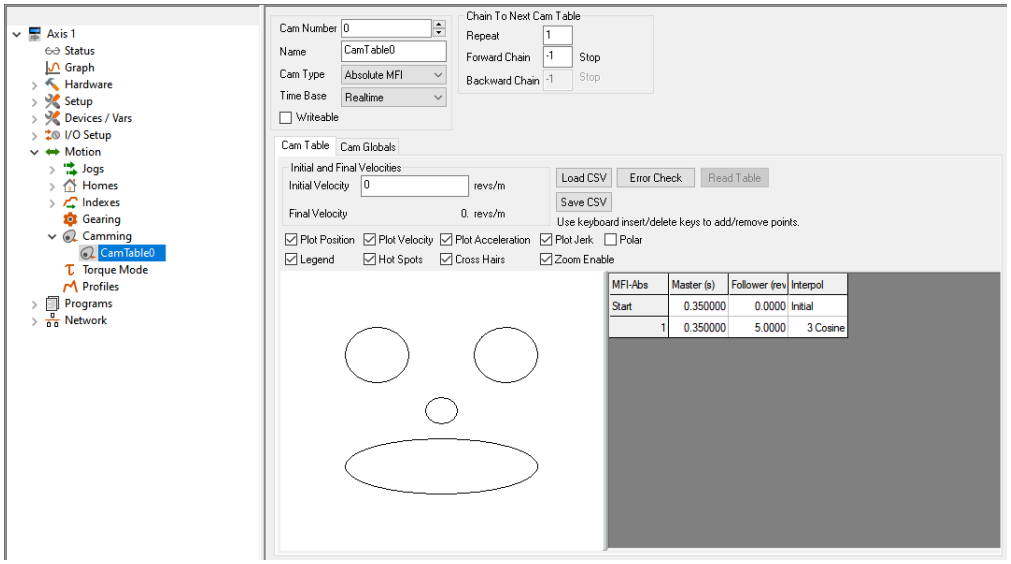


Figure 8-96: Cam Table View With a Smiley Face

When the Master and Follower parameter values are entered into the cam table the graph is updated to reflect these values. If PowerTools Studio is unable to plot the graph due to these values a smiley face will appear in the graph area of the view. To correct this error just change parameter values until the smiley face is cleared and the plot appears.

8.9.6 Torque Mode View

Torque Mode is a mode where the drive is controlled by torque command rather than a position command. The drive can switch between position and torque mode. Note that when in torque mode, there is no following error.

Torque Mode can be velocity limited. If there is not enough back force to keep the motor velocity below the velocity limit, the torque mode will switch into and out of a velocity limiting mode as needed.

The Velocity Limiting has values and enables for acceleration and deceleration. This allows for position control integration with compound and blended Indexes. We recommend using the graphical monitor to debug your motion integration.

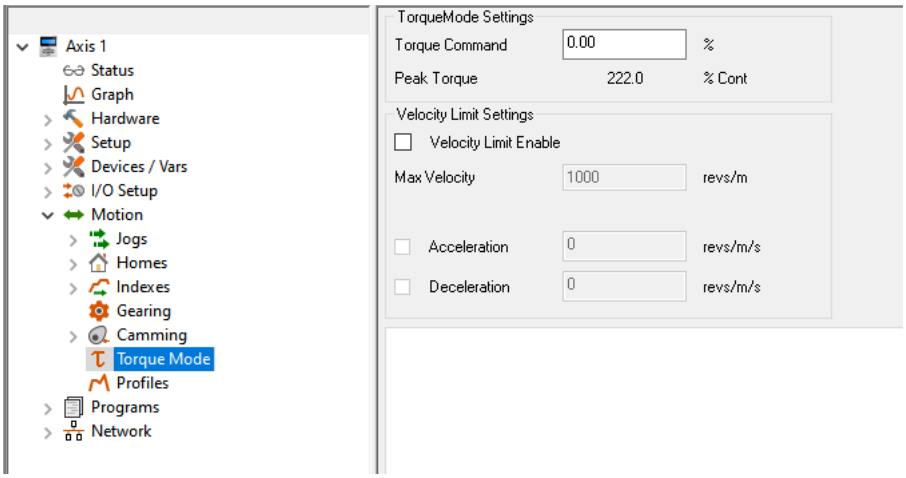


Figure 8-97: Torque Mode

Torque Mode Settings Group

Torque Command

This parameter (TorqueMode.TorqueCommand) value is the torque level that will be applied when Torque mode is activated.

Peak Torque

This is a read only parameter displays the peak torque available from the selected drive and motor combination.

Velocity Limiting Settings

Velocity Limit Enable Check Box

This check box (TorqueMode.VelocityLimitEnable) when selected will enable the velocity limiting feature in Torque mode.

Max Velocity
This parameter (TorqueMode.VelocityLimit) value is the maximum limit of the velocity when velocity limit enable is enabled.

Acceleration Check Box
Select this check box (TorqueMode.AccelEnable) when an acceleration ramp is desired.

Acceleration
This parameter (TorqueMode.Accel) is the acceleration ramp for the velocity limit feature, when enabled.

Deceleration Check Box
Select this check box (TorqueMode.DecelEnable) when an deceleration ramp is desired.

Deceleration
This parameter (TorqueMode.Decel) is the deceleration ramp for the velocity limit feature, when enabled.

8.10 Multiple Profiles
Motor motion or "Axis" motion may be generated from either of two Profiles: Profile.0 and Profile.1. Each of these Profiles can run any type of motion (i.e. Index, Jog, Gear, etc.) at any time. Both of the Profiles can generate motion simultaneously. For example while Gearing, an incremental index can be initiated "on top" of the Gear velocity. The sum of both Profiles provides the motors commanded position and this parameter is called PosnCommand.

In order to run motion on both Profiles, a program must be used. To specify which profile a motion object runs on, the On Profile instruction is used. The default Profile is Profile.0 and therefore it is unnecessary to specify On Profile.0 in user programs. If no Profile is specified, the default profile is used. For example, a user program that initiates an index on Profile.0. The following two program lines will generate the same result.

```
Index.0.Initiate
```

and

```
Index.0.Initiate On Profile.0
```

Both of these lines of code will initiate Index 0 on Profile 0. The first one uses Profile 0 because it is the default profile, and the second one uses Profile 0 because it is specified. The On Profile.0 command is completely optional, but may be used for clarity.

To run a motion object on the other profile (Profile 1), we must specify the use of Profile 1. The following program line will perform Index 0 on Profile 1.

```
Index.0.Initiate On Profile.1
```

Any motion may be run on either Profile, but running the same motion object on both profiles simultaneously is prohibited. For example, it is illegal to run Index 0 on Profile 0 and on Profile 1 at the same time.

Illegal:

```
Index.0.Initate
Index.0.Initiate On Profile.1
```

Legal:

```
Index.0.Initiate
Wait For Index.0.CommandComplete
Index.0.Initiate On Profile.1
```

Any two motion objects can be run on both profiles at the same time. For example, it is legal to run Index 0 on Profile 0 and Index 1 on Profile 1 at the same time.

Legal:

```
Index.0.Initiate
Index.1.Initiate On Profile.1
```

The distance and velocity of the two indexes is summed to generate the overall position command and velocity command for the motor.

All motion run from the Assignments view is automatically run on Profile 0. It is not possible to change the Profile on which motion run from the Assignments view operates. Therefore in order to run motion from both the Assignments view and from a program simultaneously, motion initiated by the program must be run on Profile 1.

The Profile view allows the user to view the Position Command and Velocity Command for each profile individually. An example of this view is shown below.

8.11 Create User Programs

8.11.1 Programs View

The Programs View allows the user to create application specific code to perform all of the necessary motion and I/O related function of a machine/system. Figure 8-98 shows an example of the Program View.

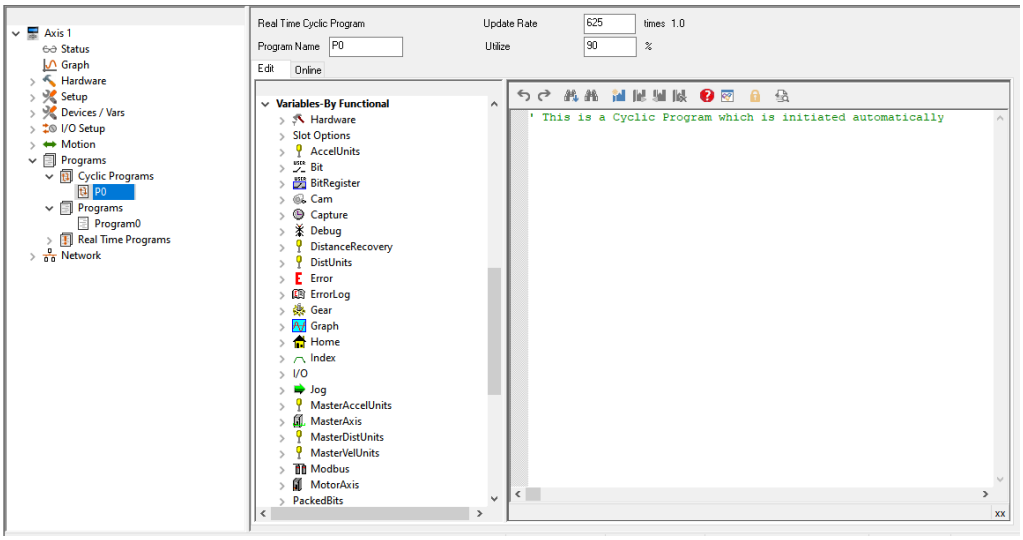


Figure 8-98: Program View

For details on the Cyclic, Real Time, and Programs views or how to create a program, see *Programming* on page 154 in this manual.

8.12 Graph View

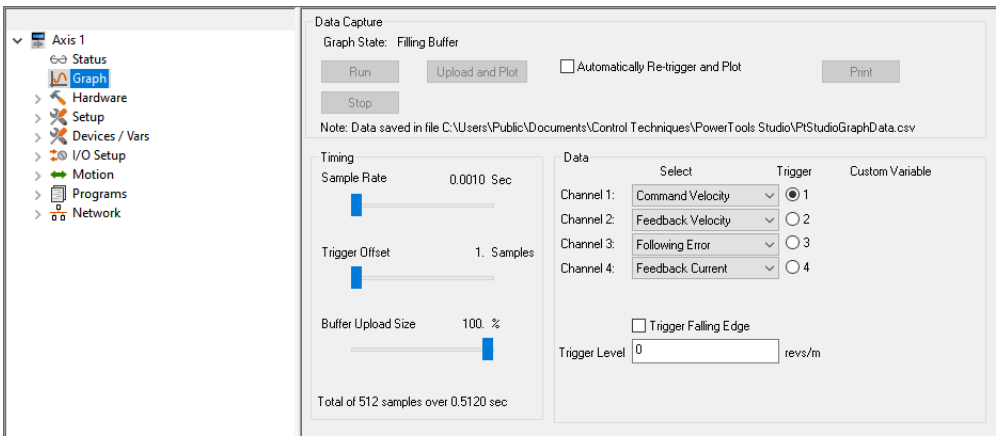


Figure 8-99: Graph View - Online

8.12.1 Data Capture Group

Graph State

There are four graph state conditions in the following order: Filling Buffer, Filled, Waiting for Trigger, and Filled and Triggered.

Run

The ***Run*** button commands the drive to begin a high speed data capture of the parameters as selected in each of the four data channels. After the ***Run*** button is activated the buffer will fill up to the trigger offset while the words “Filling Buffer” appear indicating this graph state. Once the trigger offset level is reached the words “Waiting Trigger” will appear next to the Graph State indicating that graphical monitor is now ready to be triggered based on the trigger level selected. The ***Run*** button may be activated by the letter “R” on the keyboard.

Upload and Plot

The ***Upload and Plot*** button will upload captured data from the drive and display this data in the Graph window. The user should wait for the Graph State to read “Filled and Triggered” before the data is uploaded.

Stop

The ***Stop*** button stops the data capture with the data captured at that point. You can upload and plot that data. If the buffer is only partially filled you will get a combination of good and bad data. ***Stop*** works well as a manual trigger, in place of the configured trigger.

Automatically Re-trigger and Plot Check Box

Select this check box and PowerTools Studio will monitor the graph state for the triggered condition. When this condition occurs, it automatically initiates the UploadPlot command, waits for a brief time then initiates the ***Run*** button to repeat the cycle. Initial the user must press the ***Run*** button to start the auto cycle.

This mechanism is only active when the graph view is displayed, If the user enters a different PowerTools Studio view the auto update will stop and it will restart when returning to the Graph view.

Print

The ***Print*** button is used to print the graph in the Graph window.

8.12.2 Timing Group

The sliders can be moved in several different ways.

1. With the mouse pointer over the slider, left click and hold while dragging the slider back or forth to the desired setting.

2. With the mouse pointer over the slider, left click on the slider and then the arrow keys on the PC keyboard can be used to move the slider in fine increments. The Page Up and Page Down keys move the slider in course increments. The Home key will move the slider all the way to the left and the End key will all the way to the right.

Sample Rate

The Sample Rate slider gives the user control of time spacing for the captured data. To give the user a better idea of what this number means, the total number of samples and total capture time is displayed on the bottom of the “Timing” group box.

Trigger Offset

The Trigger Offset slider allows the user to adjust the amount of data to be captured prior to and after the actual trigger point. It is often desirable to see what is happening to the system just before the trigger event activates. If the Trigger Offset slider is completely to the left (minimum # of samples), then the data captured and shown in the graph will all be AFTER the trigger point. If the slider is positioned completely to the right (maximum # of samples), then the data captured and shown in the graph will all be BEFORE the trigger point.

Buffer Upload Size

The Buffer Upload Size slider allows the user to adjust the amount of data to be uploaded when the Upload and Plot button is pressed. The system always captures all 512 samples regardless of this setting. By moving the slider to the left of the default 100 % position, the upload and plot will take less time because less data is being uploaded. If the slider is completely to the right (default), the full captured buffer will be uploaded and displayed. If the slider is completely to the left, only 1 % of the captured buffer will be uploaded and displayed.

8.12.3 Data Group

Data Channel 1 - 4 Select List Boxes

The Channel 1 through Channel 4 list boxes allow the user to select what data parameters are to be captured and displayed.
If parameters that have the same user units are selected on consecutive data channels, then those two channels will be shown on the same x/y axis when the data is graphed. If it is desirable to have two parameters having the same user units displayed on separate x/y axes, then it is necessary to put the parameters in non-consecutive channels. The data selection of None can be used on the channel separating the two desired channels.

Trigger Radio Buttons

Selecting the radio button will cause the graphical capture to trigger off the selected Channel. The Trigger Level text box on the bottom of the display will change units to the selected channel's parameter units. This trigger level may be changed at any time but the change must be sent to the drive via the ***Update to RAM*** or ***Download*** button. If a manual trigger is desired, set the channel data to None and select the corresponding trigger radio button. If no trigger is selected the capture will begin when the ***Run*** button is clicked and end at the end of the Total Sample Rate.

Module Parameter

A Module parameter text box is only available once the user has selected Module Parameter from the Data Channel Select list box. This field is used to define what PTi210 module parameter will be plotted on that channel. The module parameter can be entered two ways: by just typing any module parameter using the program format for the variable, or click the ***Popup Variables*** button and the variable window will open. Then select the variable and drag it over to the channel module parameter text box.

Trigger Mask List Box

This list box is only available when Drive I/O or Module I/O is selected in the channel select list box and the Trigger radio button is selected for that channel. The Trigger Mask list box will only list the inputs or outputs for the selected channel parameter.

Trigger Falling Edge Check Box

When the Trigger Falling Edge check box is selected, the trigger is detected when the data transitions below the trigger level.
When the Trigger Falling Edge check box is clear, the trigger is detected when the data transitions above the trigger level.

Trigger Level

This is the level at which the graph is triggered. The Trigger Level text box will change units to the selected channel's parameter unit. This trigger level may be changed at any time but the change must be sent to the drive via the ***Update to RAM*** or ***Download*** button.

8.13 Network

8.13.1 Parameter Access View

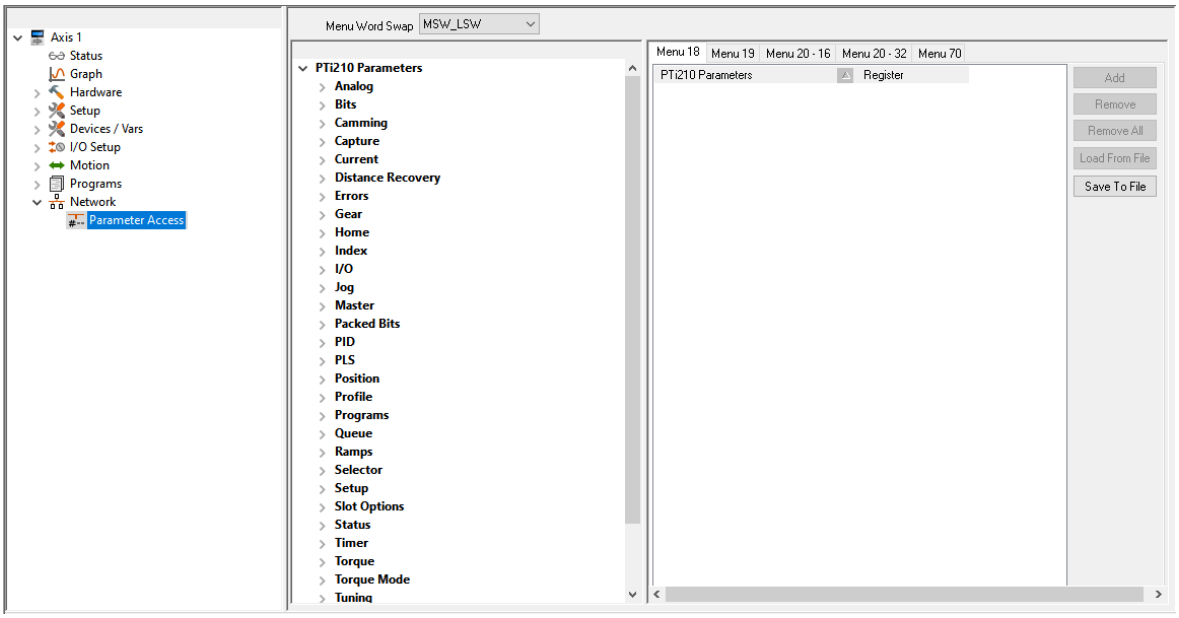


Figure 8-100: Parameter Access View - Menus 18, 19, 20 and 70

Menu Word Swap

When a 32-bit parameter is assigned to a 16-bit parameter, that parameter will take two parameters with the higher parameter holding the upper 16-bits. The default Menu Word Swap is MSW-LSW.

Example: MSW-LSW

If var.var0 = 10 and is assigned to Pr **18.011**, it will actually reside in parameters Pr **18.011** and Pr **18.012**, Pr **18.011** = 10 and Pr **18.012** = 0.

Word Swap is used in the PTi210 module to change the order of the menu parameters.

Example: LSW - MSW

If var.var0 = 10 and is assigned to Pr **18.011**, it will actually reside in parameters Pr **18.011** and Pr **18.012**, Pr **18.011** = 0 and Pr **18.012** = 10.

Menu 18 - Menu 20

The user may select parameters to be interfaced to the following drive menus:

18.002-18.010 - 16-bit, Read Only

18.011-18.030 - 16-bit

19.002-19.010 - 16-bit, Read only

19.011-19.030 - 16-bit

20.001-20.020 - 16-bit

20.021-20.040 - 32-bit

If a 32-bit parameter is assigned to a 16-bit parameter, that parameter will take two parameters with the higher parameter holding the upper 16-bits. For example, if var.var0 is assigned to Pr **18.011**, it will actually reside in parameters Pr **18.011** and Pr **18.012**.

The parameters do not have a decimal place. But the complete value is passed. Consider the example where PosnCommand is assigned to Pr **20.030** and Position has four decimal places. If PosnCommand is equal to 12.8434, Pr **20.030** will equal 128434.

For read/write parameters, the value can be changed by either the registry parameter being directly changed, or by the menu parameter being changed. If Index.0.Dist is assigned to Pr **20.021** and a program changes the value of Index.0.Dist to 12.8000, Pr **20.021** will equal 128000. If a user changes Pr **20.021** to 136434, Index.0.Dist will then equal 13.6434. If a registry parameter is read only, the menu parameter can read the value, but not change it.

The transfer between the registry and menu parameters is asynchronous and occurs a the same priority as a user program.

Menu 70

Menu 70 Setup is used to gain non-cyclic access to PTi210 parameters in the module. When the user drags a PTi210 parameter from the left to the right PowerTools Studio gives the user the flexibility to assign a drive parameter number to the PTi210 parameter. Using this newly assigned address the user may now access this PTi210 parameter using a PPO 4 word configuration (see the *SI-Profibus User Guide* for more information), DeviceNet explicit messages (see the *Drive User Guide*), or Modbus TCP/IP (see the *Drive User Guide*). PTi210 menu 70 parameters are NOT accessible using Modbus RTU from the front of the drive.

Menu 70 parameters are 32-bit signed integer parameters with an address range from 7017 to 7599 inclusive, these parameters are not accessible from user programs

Example:

It is possible to send PTi210 parameters to another drive using RTMoE cyclic links for use in Master/Follower or Digital Lock applications.

In this example the Source drive is an M700 (IP Address: 192.168.0.94) with the PTi210 module fitted in slot 3, and the destination drive is also an M700 (IP Address: 192.168.0.14), we will create an Easy Mode non-synchronous (synchronous links are not supported) cyclic link using the onboard Ethernet interface menu 10 running at 1 ms.

The PTi210 parameter "PosnCommand" will be assigned to PTi210 Menu 70 parameter 21 (**#70.21**) and sent on cyclic link#1 using the Tx1 Easy Mode link, and will be received on the destination drive using the Rx1 Easy Mode link and written to Menu 20 parameter 21 (**#20.021**).

In the Source drive:

1. From the "Network > Parameter Access" view, select "Position > PosnCommand" PTi210 parameter and add it to the Menu 70 tab screen using Register #70.21.
2. Download the application configuration.
3. Create the cyclic link to send the PTi210 parameter, assuming initial default values, set the following drive parameters:
   3.1 **4.10.011** = 1 (*Tx1 Link Number*)
   3.2 **4.10.012** = 3.70.021 (*Tx1 Source Parameter*)
   3.3 **4.10.013** = 1 (*Tx1 Parameter Count*)
   3.4 **4.10.015** = 192.168.0.14 (*Tx1 Destination Address*)
   3.5 **4.10.016** = 1 (*Tx1 Message Rate*)
   3.6 **4.10.002** = 1 (*Easy Mode Reset*)

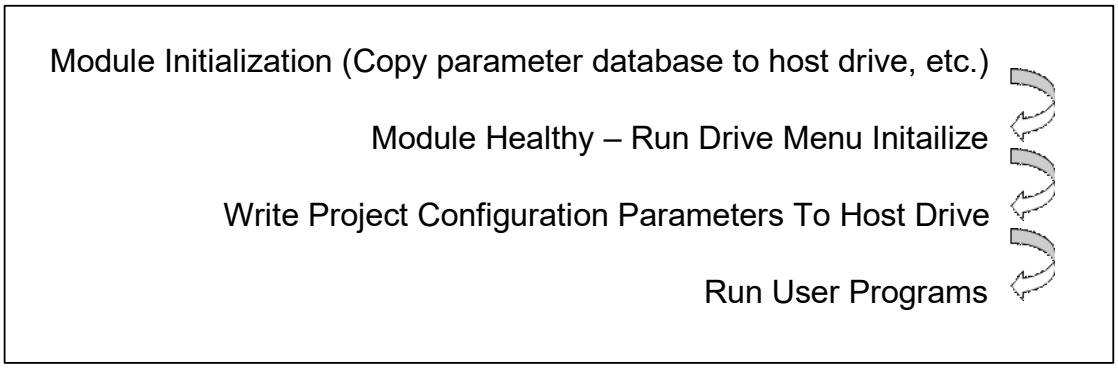
In the Destination drive:

1. Create the cyclic link to receive the PTi210 parameter, assuming initial default values, set the following drive parameters:
   1.1 **4.10.041** = 1 (*Rx1 Link Number*)
   1.2 **4.10.042** = 0.20.021 (*Rx1 Destination Parameter*)
   1.3 **4.10.043** = 1 (*Rx1 Parameter Count*)
   1.6 **4.10.002** = 1 (*Easy Mode Reset*)
2. After a few moments the cyclic link will be created and the encoder position command on the PTi210 drive will be sent to the other drive parameter (**#20.021**).

8.14 PTi210 Initialization Sequence

To aid diagnostics and to assist the user developing applications, it may be useful to understand the initialization sequence of the PTi210 option module.

There are four main initializing routines in the PTi210 module, which can be illustrated in the following diagram.



This shows that drive parameters written in the Drive Menu Initialize will be over-written by the project configuration, and the project configuration parameters will be over-written by the User Programs.

9 Programming

9.1 Programs

Expanding Programs in the Hierarchy Tree the user will notice there are three types of programs. Cyclic Programs, User Programs (Programs), and Real Time Programs.

A Cyclic Program is designed to execute over several update rate cycles, utilizing a specified amount of the update rate time and a specified number of update rate cycles. CyclicProgram.#.EnableProgram must be activated to initiate the Cyclic Program.

A Real Time Program is designed to execute to completion in a single update rate cycle. RealTimeProgram.#.EnableProgram must be activated to initiate the Real Time Program.

User Programs can be initiated either using the Program.#.Initiate destination or the program instruction. User Programs use as many update rate cycles as needed to complete the program.

9.2 Program Window Components

There are 5 major components to the Program Views or programming window. These components are the Program Parameters, Program Toolbar, Instruction List, Red Dot Error Bar, and the Code Window. Figure 9-1 points out the components listed above.

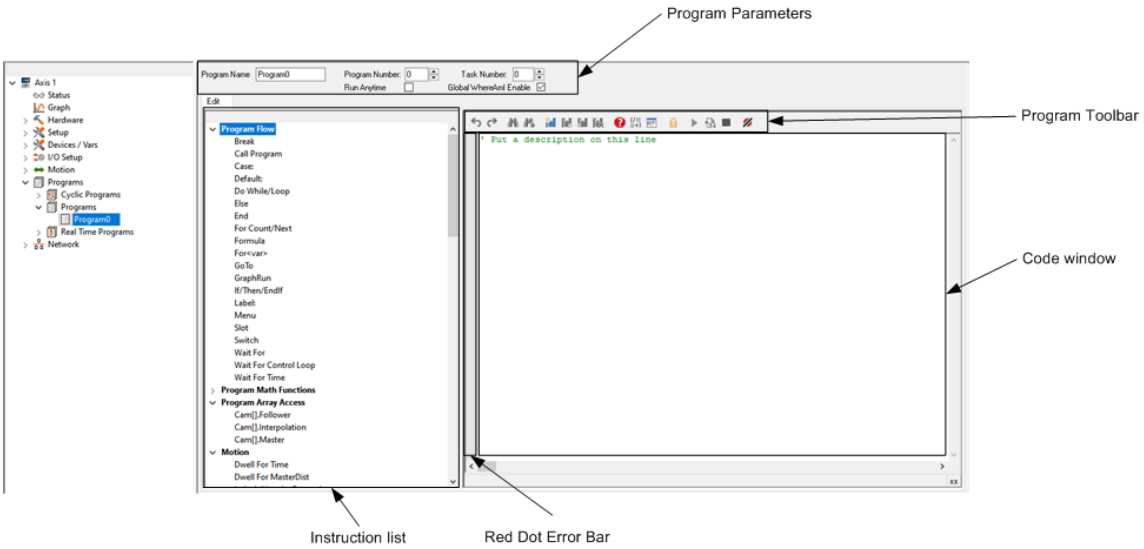


Figure 9-1: Program View Components

Each of these main components must be used to successfully create a program. Each of the components are described in detail below and in the individual program type views.

9.2.1 Program Toolbar

Following is a detailed description of each of the buttons found on the Program Toolbar. These buttons will help the user edit programs as well as debug errors and troubleshoot program functionality. Some of these buttons are only available when online with the drive and others are not available in a Real Time or Cyclic programs.

Figure 9-2: Program Toolbar

Undo Last Change 

This button will undo the last change made to the program. PowerTools Studio will save up to ten of the last changes performed in the program.

Redo Last Change 

This button will redo the last change that was undone. PowerTools Studio will save up to ten of the last changes that have been undone in the program.

Find 

This button allows the user to search for a given string inside the program. Modifying several parameters in the Find dialog box (i.e. Search Up, Search Down, Match Case, etc.) can customize the search.

Find Next

This button will find the next instance of the string last searched for. This allows you to quickly find all the matches to your search with out re-entering the selected word.

Bookmark

This button will insert a bookmark on the line of code on which the cursor is placed. Bookmarks allow the user to mark certain sections of the program for easy access to at a later time. The next Bookmark and Previous Bookmark buttons can be used to jump from one bookmark to the next very quickly. If this button is clicked when a bookmark already exists on the line of code, the bookmark will be removed.

Goto Next Bookmark

This button will position the cursor on the next available bookmark ahead of the cursor in the program.

Goto Previous Bookmark

This button will position the cursor on the previous bookmark behind the cursor in the program.

Delete All Bookmarks

This button will delete all of the bookmarks in the program. To delete only a single bookmark, place the cursor on the line for which you wish to delete the bookmark, and click on the Bookmark button.

Red Dot Help

If a user program contains an error, the realtime program parser will detect it, and place a red dot next to the line of code with the error. For help on what the particular error is, click on the Red Dot Help button, and then click on the line of code with the red dot next to it. PowerTools Studio will attempt to give a detailed description of the error.

Drag In Operands

This button will open the Drag In Operands pop-up window. From this window, the user can drag formula Operands (i.e. +, -, /, *) into the program formula.

Drag In Variables

This button will open the Drag In Variables pop-up window. From this window, the user can find any variable they wish to use in a program, and simply drag it into the program code.

This list will easily allow you to find any of the available pre-defined variables in the PTi210 module. The available parameters shown in the window depends on the selected Program User Level.

Lock Program

Toggling this button will lock and unlock the program for editing. When locked, the user is not able to modify the program code. After downloading, the program automatically locks to prevent the user from inadvertently changing program statements. To unlock the program, simply click the button.

Run This Program

Clicking on this button will automatically initiate the program that is currently being viewed. The drive must first be enabled in order to run a program. (Only available while online)

Program Where Am I?

Clicking on this button will show the line of the program that is currently being executed. A blue arrow will point to the line in the program that was executing when the button was clicked.

The arrow will not continue to follow program flow. If the program is not currently running, then the arrow will point to the top of the program, or to the last line of the program that was processed before it was stopped. (Only available while online)

Stop All

This button is the same as the Stop destination found in the assignments view. Clicking on this button will stop all programs and motion. If in motion, the motor will decelerate to a stop using the StopDeceleration ramp value. (Only available while online)

Disable/Enable Error Check

This button can be used to temporarily disable the program parser. The parser is what detects errors in a user program. When user programs are very large, the parser can take an appreciable amount of time to check the entire program for errors. To avoid this, the

user can disable the program parser, enter all of the changes, and then re-enable the parser to check for errors. (Only available while online)

9.3 Cyclic Program View

The Cyclic Program initiation is synchronized to a multiple of the update rate cycles. Therefore it provides a deterministic cycle time that is repeated automatically. The Cyclic Program executes over multiple update rates and therefore has a reduced program instruction set compared to the non real time programs.

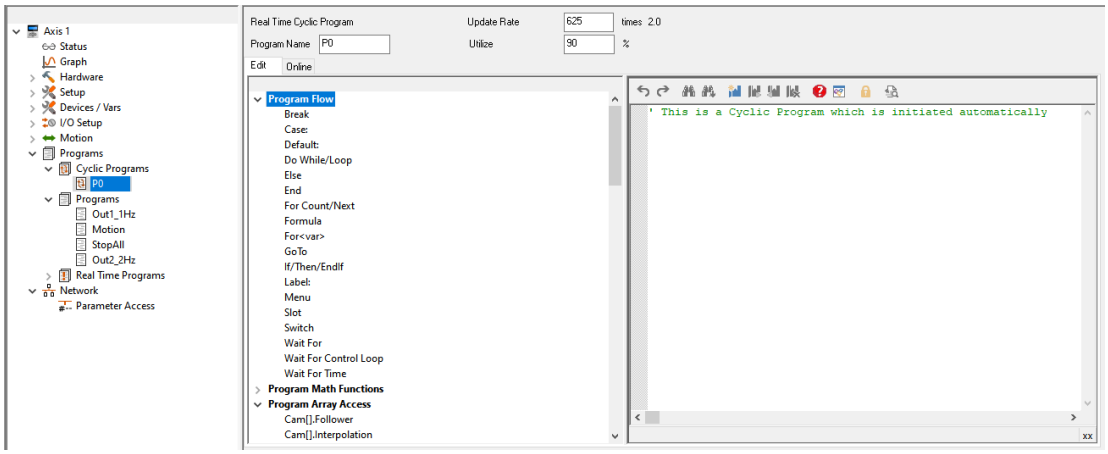


Figure 9-3: Cyclic Programs View

9.3.1 Program Parameters for a Cyclic Program

Program Name

This is a 12 character string that the user can assign to an individual program. It allows the user to give a descriptive name to the program for ease of use.

Update Rate

This parameter defines the number of Update rates the cyclic program has to finish. This parameter can only be setup on the Cyclic Program view.

Utilize

Utilize is the percentage of the Control Loop time utilized for control loop, Real Time Program and a portion of the Cyclic Program. When this limit is hit, the cyclic program is suspended until the next control loop. The percentage left over will process user programs, communications and other processes. To prevent complete starvation of user programs and communications, the execution of the Cyclic Program is spread out over several control loops based on the Utilize percentage.

The drive's CPU time is allocated between the background processes and the foreground control loop interrupt. The control loop interrupt executes event updates, motion calculations, the complete real time program and a portion of the cyclic program up to the Utilize percentage of the trajectory rate. Then the control loop interrupt is exited and the background processes continue to run.

No matter how low the Utilize percentage is set, the required control loop motion calculations will be completed, the Real Time Program will run to completion, and at least one cyclic instruction will be executed. If these do not complete, a real time program timeout fault is generated. A Cyclic Program Timeout Fault occurs if the cyclic program has not completed at the next Cyclic Program Initiate.

In actuality the Cyclic Programs are initiated from the control loop but are run outside the control loop interrupt. They run as processes along with user programs, but at a higher priority so cyclic programs are run before any user program. Other processes may run at lower or higher priorities. These include processing the communications. The cyclic programs are suspended and resumed based on the utilize percentage and trajectory rate.

9.4 User Programs View

Click on Programs\Program#, the program view will appear on the right (see Figure 9-4). The left side (pane) of this view contains the instruction list. The right side of the Programs view contains the Program Toolbar above the program code window.

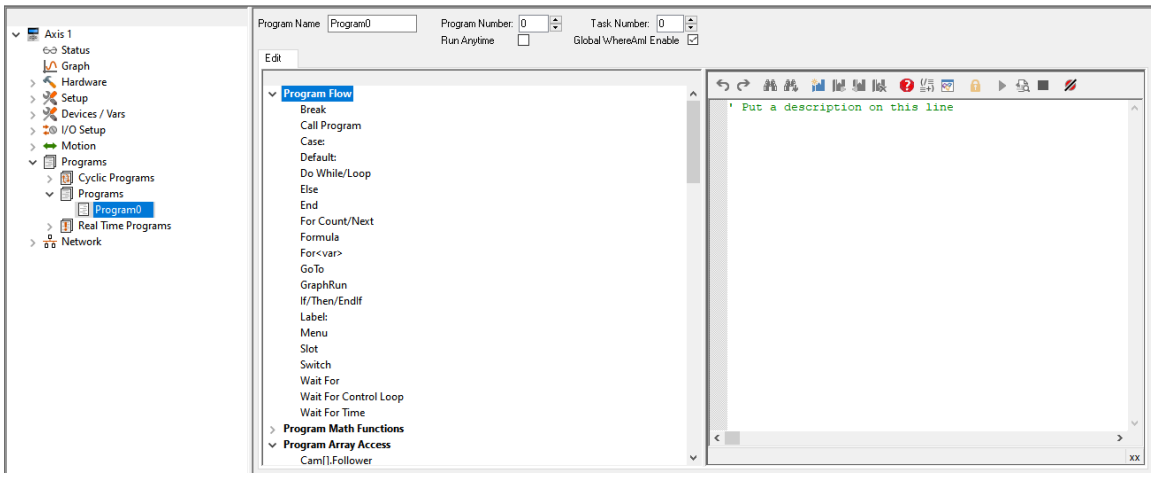


Figure 9-4: User Program View

9.4.1 Program Parameters for User Programs

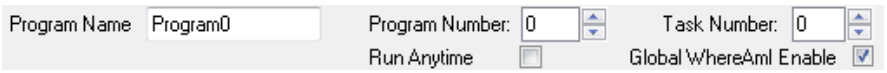


Figure 9-5: Program Parameters

Program Name

The user can enter a descriptive name up to 12 characters for the specific user program.

Program Number

Each program has an associated Program Number. The number helps to keep the programs organized.

Task Number

The Task Number parameter determines which task the program runs on. Default Task Number is 0. Users can select Task Numbers from 0 to 3. Programs assigned to different tasks can run simultaneously due to the multitasking capabilities of the PTi210 module. Programs assigned to the same task cannot run at the same time.

For more information on Multitasking, refer to *Program Multi-Tasking* on page 161.

Run Anytime Enable

The PTi210 module programming environment has been designed to automatically stop all user programs when an error occurs (regardless of what type of error), or when the drive is disabled. Some applications require the ability to run a program as soon as an error occurs, or continue running a program even through an error is active. In order to do this, the program must be classified as "Run Anytime". To configure the program to be able to run during an error or while the drive is disabled, the Run Anytime check box must be selected in the Program view.

When an error occurs, the drive will still be disabled, and no motion will be possible. For this reason, it may be necessary to reset the error in the Run Anytime program prior to initiating motion again. If a motion instruction is processed while the drive is disabled, the program will stall on that particular line of the program, but the program will not be stopped.

Certain conditions will still cause a program designated as Run Anytime to stop. These conditions are listed below:

- Stop Function is activated
- Run Anytime program has a Program error

Multiple programs may be configured as Run Anytime programs. Run Anytime programs can be called from a user program just the same as any other program. If a Run Anytime program calls another program which is not configured as Run Anytime, while the drive is faulted or disabled, the task will be stopped.

Resetting Errors in "Run Anytime" Programs

To reset an error from a Run Anytime program, use the Error.Reset = ON command in the user program. The Error.Reset command does not clear all types of errors. Some errors require power to be cycled in order to clear the error. For more information on the method used to clear individual errors, see *Diagnostics* on page 239.

After using the Error.Reset command in a user program, use a Wait For Time 0.100 'seconds command to give the drive time to clear the error and re-enable the drive before initiating motion. If this is not done, the motion will be initiated before the drive is enabled, and the instruction will be ignored.

9.4.2 Global Where Am I Enable

This is used to control the functionality of the Global Where Am I arrow in a program. If the user activates the Global Where Am I feature by clicking the Global Where Am I button on the PowerTools Studio toolbar, a blue arrow will follow the program flow on a given task by pointing to the line of the user program that is currently being processed. If the Global Where Am I is active, and one user program calls another user program (using the Call Program instruction), the PowerTools Studio view will automatically switch to the "called program". In some cases it may be desirable to stop the screen from automatically changing to the "called program", this can be done by disabling (clearing) the Global Where Am I Enable check box. By default, all Global Where Am I Enable check boxes are selected (active).

9.5 Real Time Program View

The Real Time program is designed to be executed to completion in every update rate cycle. The real time program instruction set is reduced to ensure all the operations are completed within the update rate.

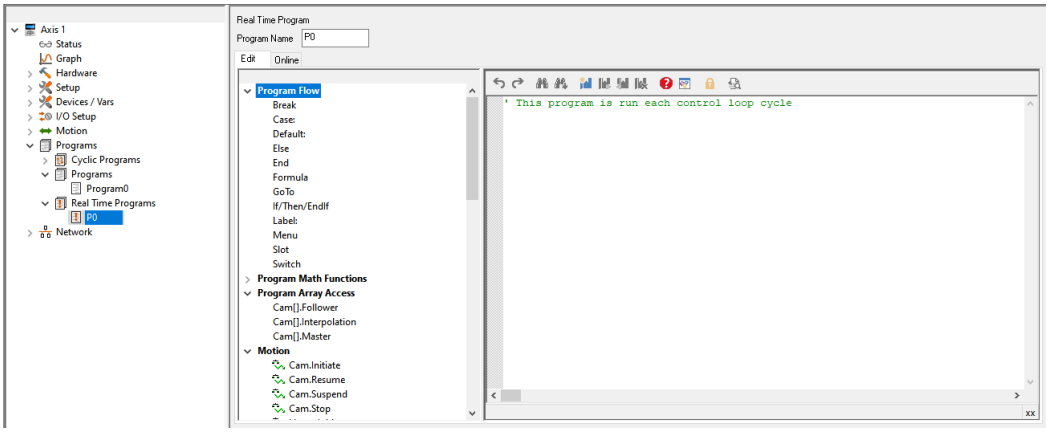


Figure 9-6: Real Time Programs View

9.5.1 Program Parameter for a Real Time Program

Program Name

This is a 12 character string that the user can assign to an individual program. It allows the user to give a descriptive name to the program for ease of use.

9.6 Program Multi-Tasking

Many applications require the operation of a background task that operates outside of the main program loop, but must be consistently processed. For instance, a background task that performs calculations for values sent to an operator interface or a background task that monitors parameters for error detection.

The PTi210 module processor has the ability to execute multiple tasks. Because only one task can be processed at a time, a process called time slicing must be used. Time Slicing is simply splitting the total processing time between multiple tasks. The PTi210 module processor generates an interrupt that causes any task to stop, and the control loop to update. The timing of the interrupt is dependent on the Trajectory Update Rate parameter configured by the user on the Setup view (default update rate is 1 msec). Within the control loop update, the PTi210 module updates the motion trajectory, captured data, the PTi210 module digital inputs and outputs, and other control parameters. Between each control loop update, the PTi210 module processes messages (i.e. Modbus, Errors, etc.) and then runs as much of the user program as possible until the next interrupt begins. Each update, a different task is processed. Therefore, the time it takes a given user program to complete depends how many tasks are being processed.

The task assignment is performed on the Program view. Figure 9-7 shows the Program view with the Task Number parameter. Use the up and down arrows next to the Task Number to change the task number. To create a new Task, simply click the up arrow until PowerTools Studio asks if you wish to create a new Task.

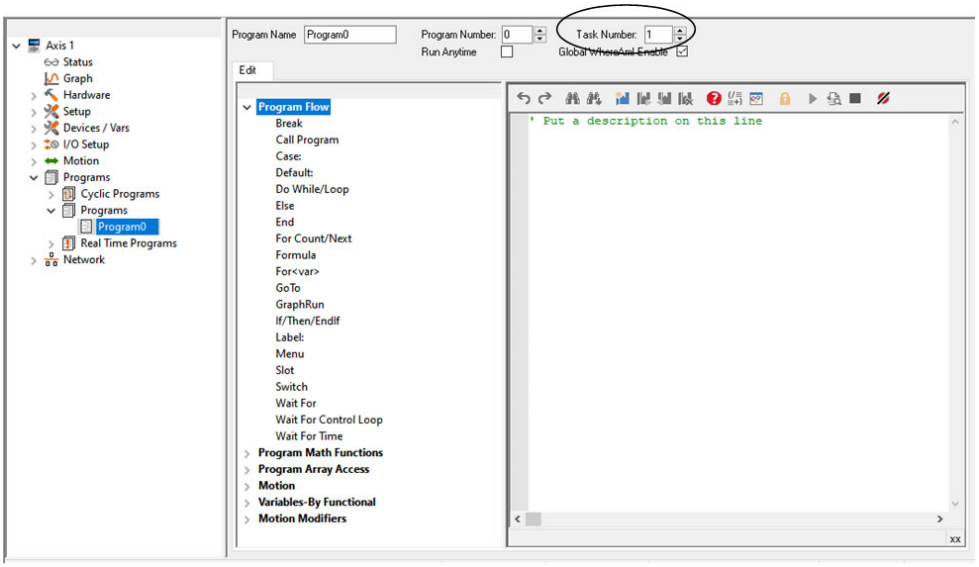


Figure 9-7: User Program View

The PTi210 module allows up to four different tasks in a single application.

If the user wishes to operate two programs simultaneously, the two programs must be assigned to two different tasks. Multiple programs can be assigned to the same task if desired, but that means that the two programs can not be run at the same time. If a given program calls another program, then calling and the called programs must be on the same task. All programs default to task zero and therefore will not run simultaneously unless specified to do so.

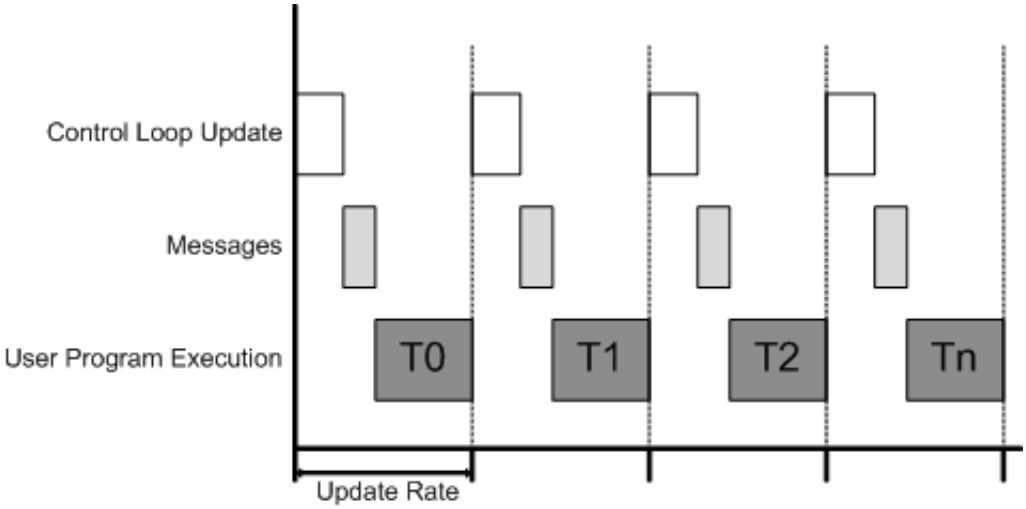


Figure 9-8: Time Slicing with Multiple Tasks Handling Diagram

9.7 Timing Diagram

In Figure 9-8, the Update Rate represents the Trajectory Update Rate that the user selects on the Setup view. The first routine to be processed in the update is the control loop update. Next, all messages will be handled. If no message has been sent from a Modbus master, and no errors are active, then this step is skipped. After all messages are processed, then execution switches to the user programs. The user programs are assigned to tasks, and the tasks are handled in ascending order starting with task 0. If a task has been assigned, but not initiated, then that task can be skipped. When the next interrupt occurs, the task is stopped, and the process is repeated using the next available task. Once each task has been processed (depends on how many have been assigned by the user), the whole process starts over at the first task. This process description is accurate as long as no program is blocked.
The following figure shows examples of user programs and task numbers and how the PTi210 module processes them.

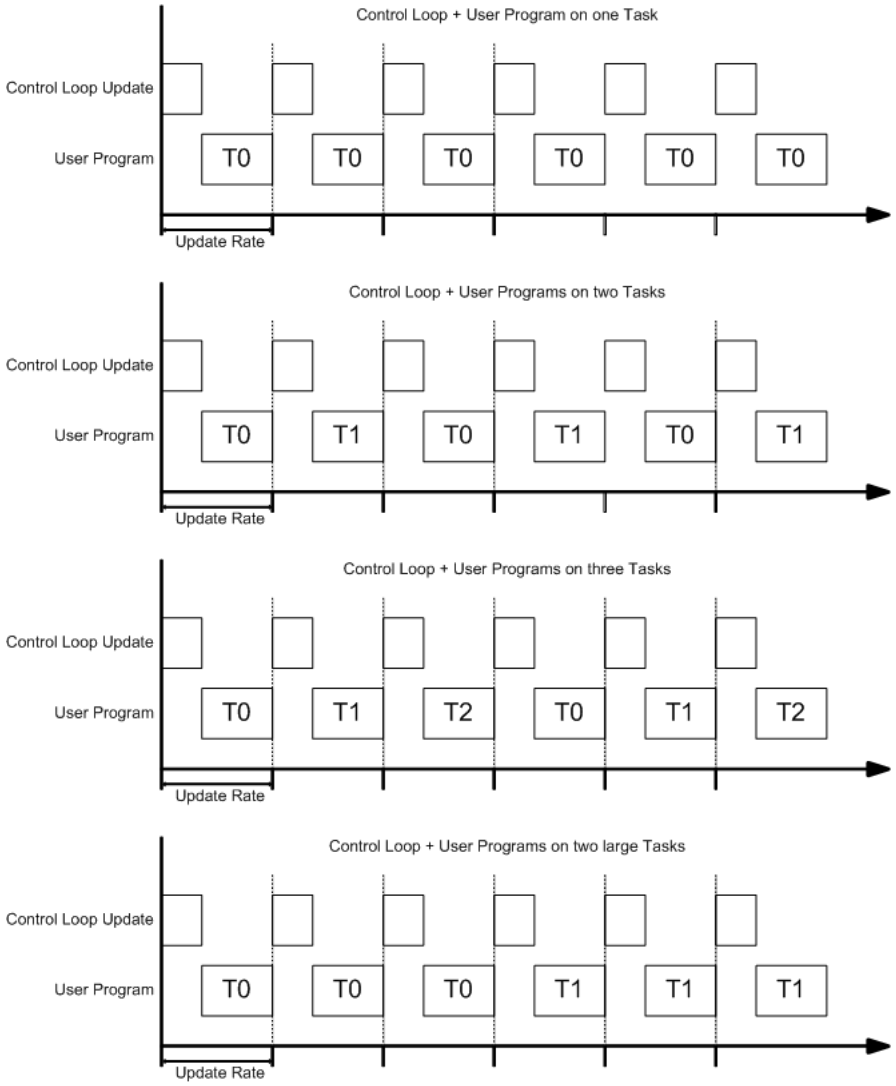


Figure 9-9: Diagram of User Programs with Multiple Tasks

The following three figures are timing diagrams of a cyclic program with different utilization values (showing the effect) and a user program initiated to run on a single task.

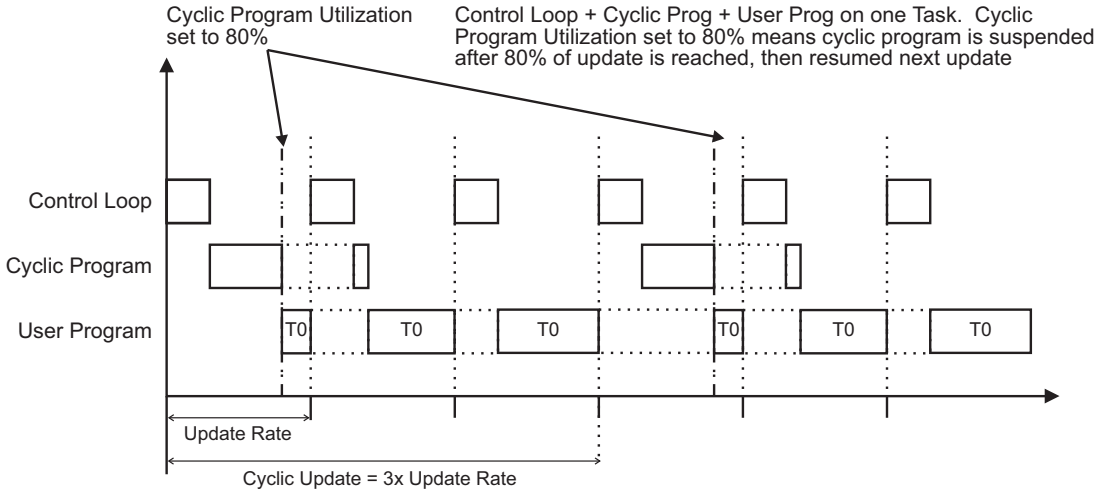


Figure 9-10: Diagram of User Program and a Cyclic Program

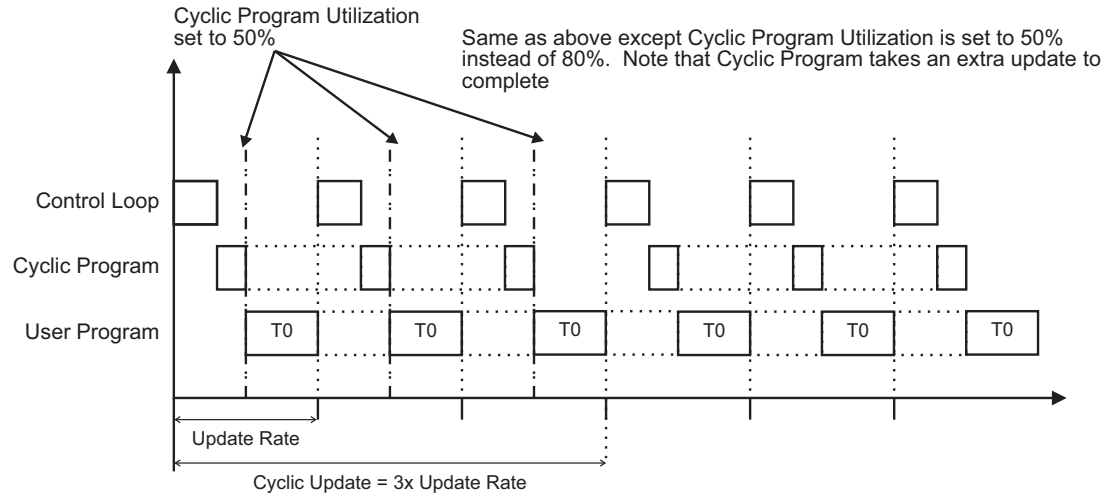


Figure 9-11: Diagram of User Program and a Cyclic Program

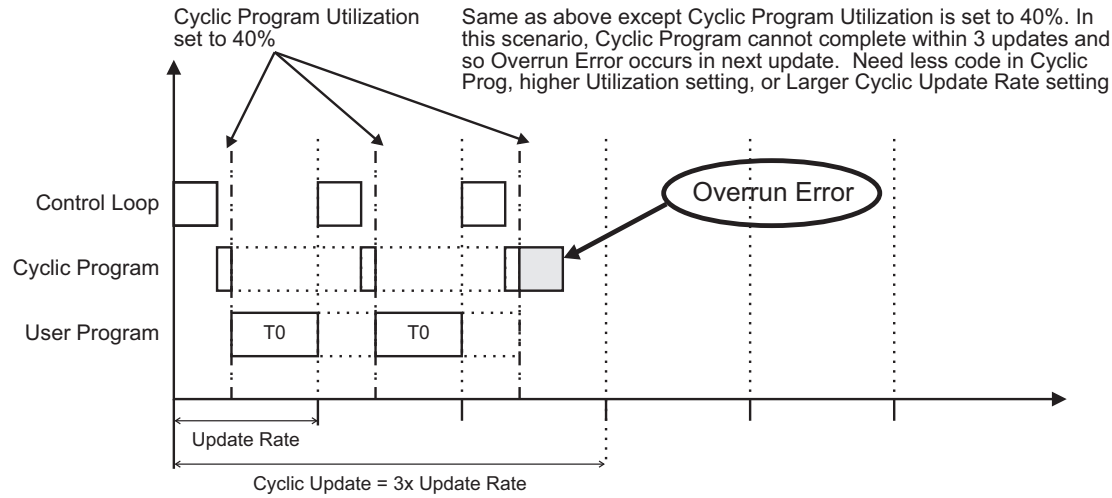


Figure 9-12: Diagram of User Program and a Cyclic Program

The following two figures are timing diagrams of a cyclic program with different utilization values (showing the effect) and a user program initiated to run on several tasks.

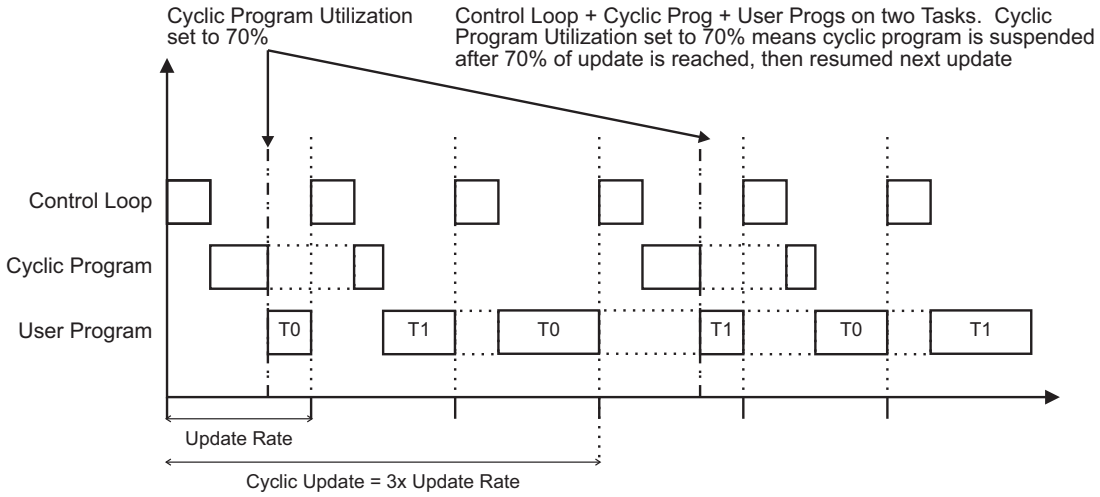


Figure 9-13: Diagram of User Programs on Two Tasks and a Cyclic Program

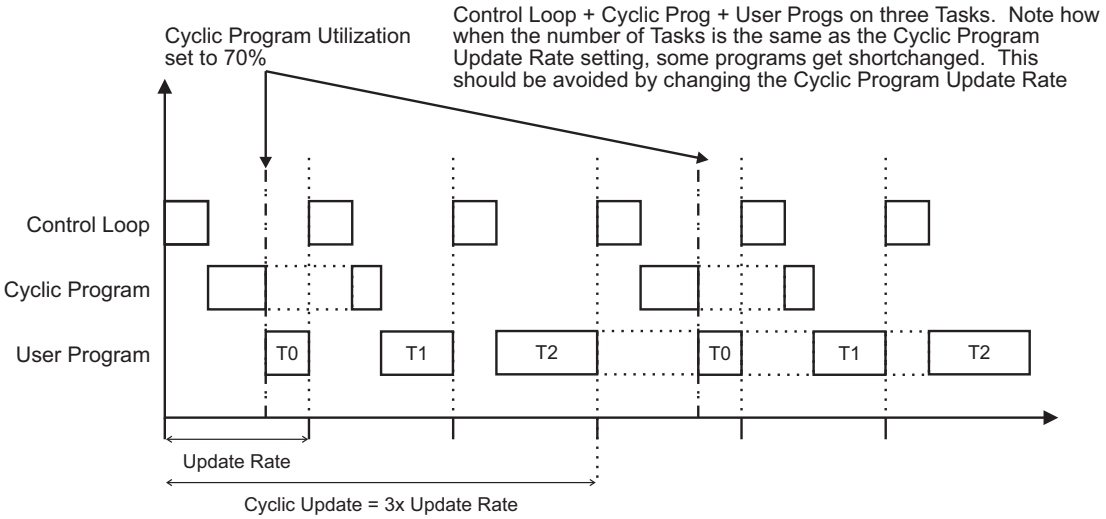


Figure 9-14: Diagram of User Programs on Three Tasks and a Cyclic Program

The next two figures show how the Real Time program is executed prior to the user program and how the size of the Real Time program effects the time left in the update rate for the user program to complete.

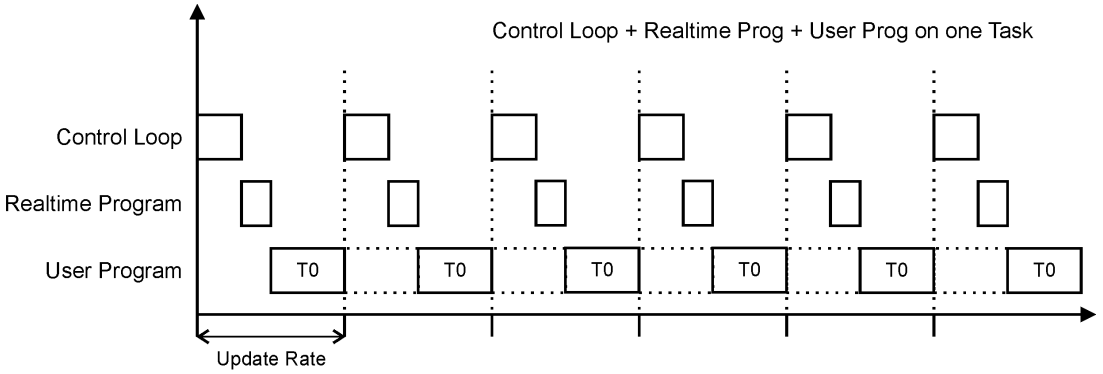


Figure 9-15: Diagram of User Program and a Real Time Program

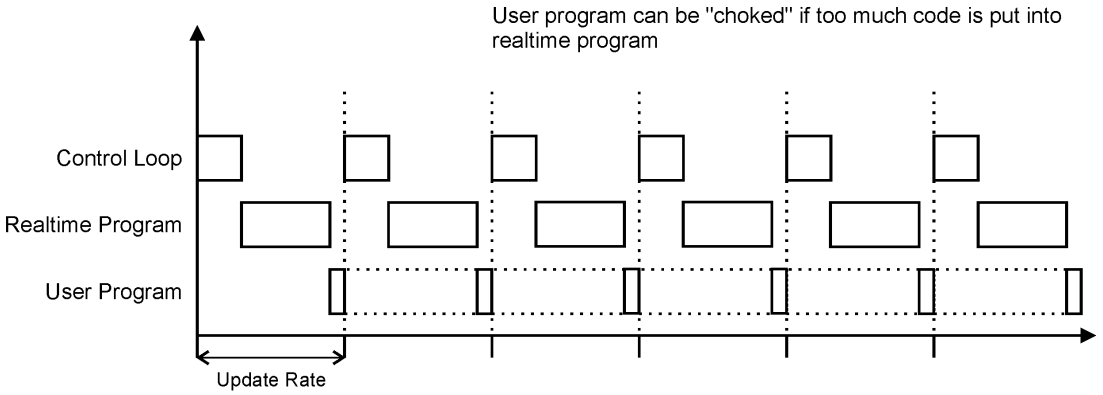


Figure 9-16: Diagram of User Program and a Real Time Program

When the Real Time program is to large and can not be completed in the update rate time the drive will error.

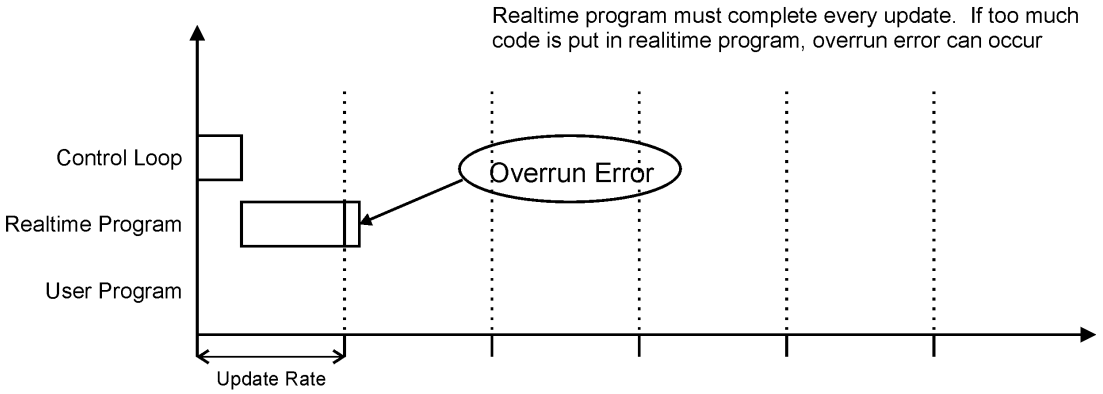


Figure 9-17: Diagram of User Program and a Real Time Program

Figure 9-18 shows how the Real Time program finishes and the two task User programs are processed.

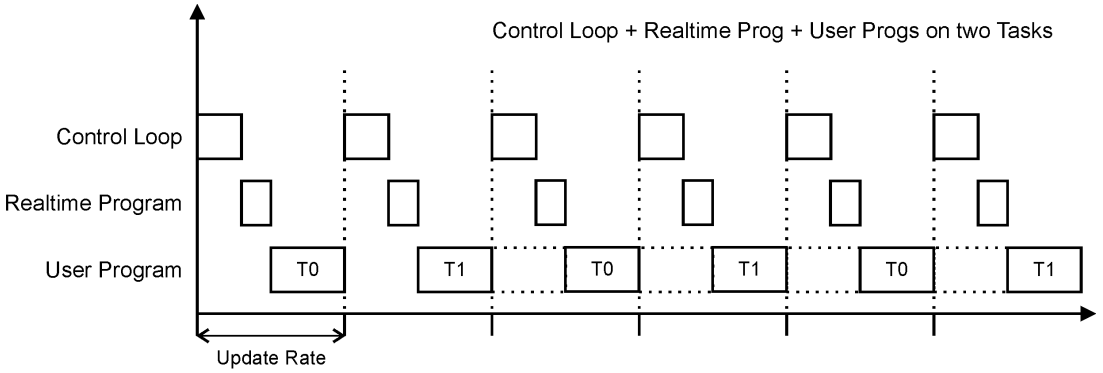


Figure 9-18: Diagram of User Programs on Two Tasks and a Real Time Program

The following three figures show how the module processes Real Time, Cyclic, and User Programs with different number of tasks.

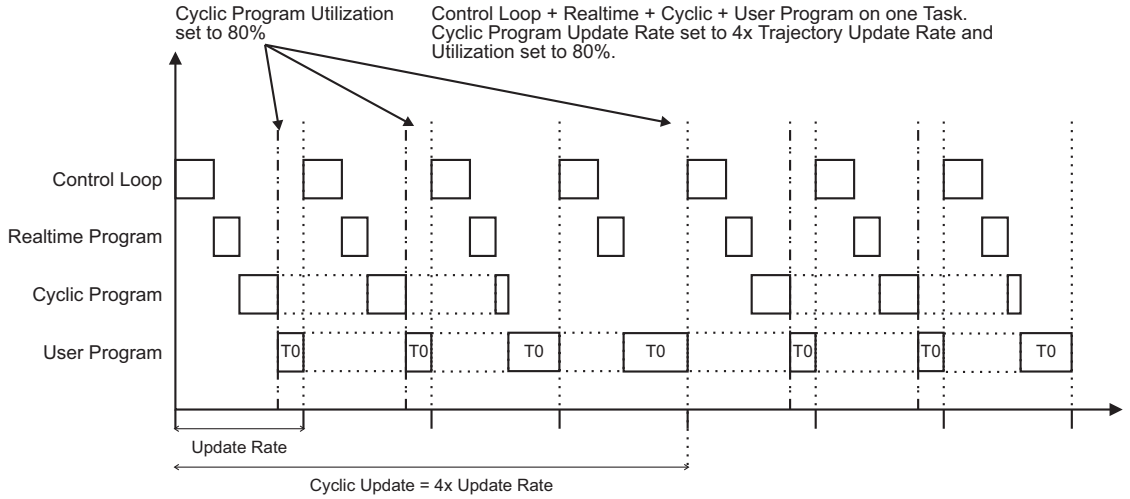


Figure 9-19: Diagram of User Program, a Real Time Program and Cyclic Program

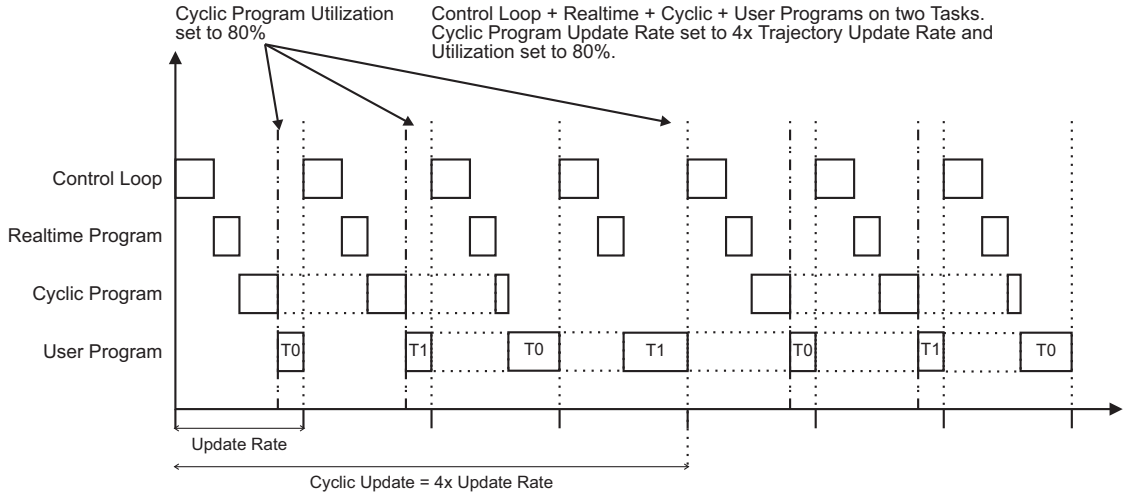


Figure 9-20: Diagram of User Programs on Two Tasks, a Real Time Program and a Cyclic Program

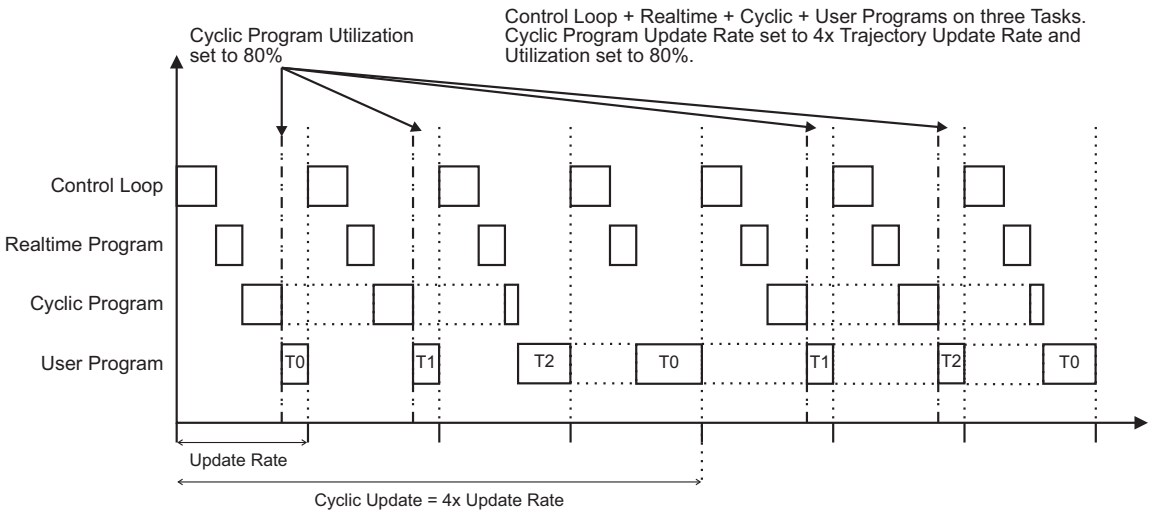


Figure 9-21: Diagram of User Programs on Three Tasks, a Real Time Program, and a Cyclic Program

9.8 Program Instruction List

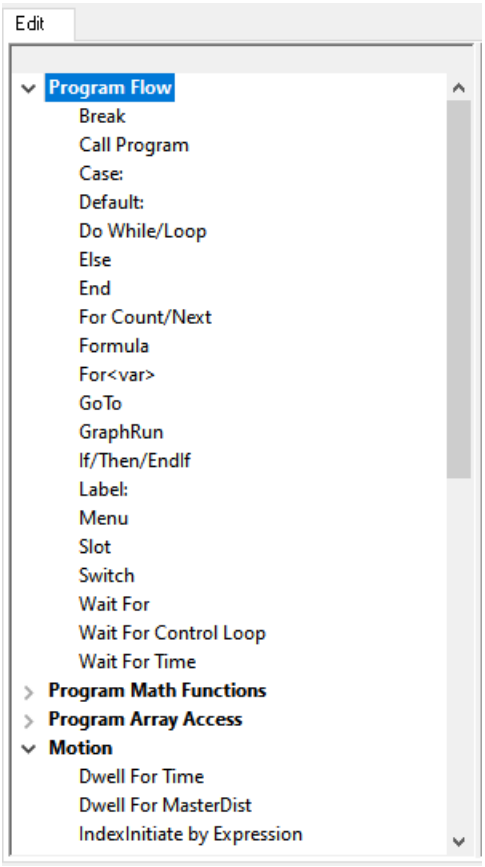


Figure 9-22: Program Instruction List

9.8.1 Program Flow Instructions

Break

For Switch instruction, Break stops the instruction execution flow and continues execution after the EndSwitch instruction. A Break is required to prevent the Case instruction flow continuing the executing through to the next case or default instructions.

For Do While instruction, Break stops the instruction execution flow and continues execution after the Loop instruction.

With the For Count instruction, Break stops the instruction execution flow and continues execution after the Next instruction.

Example:

```
Switch<var.var0>
case <1>
   Home.0.Initiate        'Home0,Sensor,SpecifiedOffset=0.0000 revs,Vel=200 revs/m
break
case <2>
   Dwell For Time 60 'seconds
break
Default:
   Index.0.Initiate       'Index0,Incremental,Distance-300 revs,Vel=2000 rev/m
Wait For Index.AnyCommandComplete
EndSwitch
```

In the above example if var.var0 = 2 then the program will execute from case <2> and process the instructions of case 2 and proceed through the program. If var.var0 = 0 the program starts to execute from Default and process continue from there.

Call Program

This program flow instruction is used to call another program. When the called program finishes the controller picks up where it left off in the program that called it. This is often used when a section of code is used in multiple places in a program. By moving the code to another program and then calling that program, the total number of program lines can be reduced. This can also make the program easier to understand. Care should be taken not to "nest" more than four program calls due to processor stack overflow. Therefore, no more than four different programs should be called without returning to the original program.

In the diagram below, => Represents a Call Program instruction.

Program0 => Program1 => Program2 => Program3 => Program4 ⛔ NO MORE THAN 4!

Example:

```
Call Program.10                 'Program 10 contains a complex home
                                'routine.
```

Example:

```
Call Program.100                'Program 100 contains a "teach position"
                                'routine.
```

Case

When the Switch Instruction value and the case number instruction match, all the instructions that follow the case instruction up to a Break or EndSwitch are executed. This includes instructions following the next Case instructions and the default instruction found. Duplicate "Case numbers" are not allowed.

Default:

The default is an optional statement within the switch instruction. It must follow all the case instruction within a Switch instruction. When none of the case instruction numbers match the switch expression value the program instructions following the Default instruction are executed. They are also executed if there is no break instruction in the previous case statements

Do While/Loop

This program instruction is used for repeating a sequence of code as long as an expression is true. To loop forever use "TRUE" as the test expression as shown in the third example below. The test expression is tested before the loop is entered. If the test expression is evaluated as False (0) the code in the loop will be skipped over.

Logical tests (AND, OR, NOT) can be used in the Do While/Loop instruction. Parenthesis "()" can be used to group the logical tests.

Example:

```
Do While ModuleInput.1=ON
   Index.1.Initiate
   Dwell For Time 1.000 'seconds
Loop
```

Example:

```
Do While (Slot2.DIO.4=ON AND Slot2.DIO.4=OFF)
   Index.1.Initiate
   Dwell For Time 1.000 'seconds
Loop
```

Example:

```
Do While (TRUE)
   Index.1.Initiate
   Dwell For Time 1.000 'seconds
Loop
```

Else

This program flow instruction is used in conjunction with the If/Then/Endif instruction. If the If/Then test condition evaluates to true the code after the If/Then and before the Else is executed. If the test evaluates to false the code between the Else and the Endif is executed.

Example:

```
If ModuleInput.1=ON Then
   ModuleOutput.1=ON
   ModuleOutput.2=OFF
Else
   ModuleOutput.1=OFF
   ModuleOutput.2=ON
Endif
```

Example:

```
If (DriveInput.5=ON) Then
   Jog.0.Vel = 1.0 'in/s
Else
   Jog.0.Vel = 0.1 'in/s
Endif
```

End

This program flow instruction is used to halt the execution of the program. It can be used in multiple places within the program. It is not required on the last line of every program. It is implied that after the controller executes the last line of code in a program the program will halt.

It is commonly used inside of If/Then/Endif constructs to end the program if a certain condition has been met.

Example:

```
If ModuleInput.1=OFF Then
   End
Endif
```

Example:

```
If ModuleInput.1=ON Then
   ModuleOutput.1=ON
   End
Endif
```

For Count/Next

This instruction is used to execute a section of code a specific number of times.

The user must enter the two numbers that control the repeat count. This instruction can either count up or count down. If the first number provided by the user is larger than the second number, then the instruction will count down. If the first number provided by the user is smaller than the second number, then the instruction will count up. The valid range for the numbers provided is 0 to 2147483647.

Example:

```
For Count = 1 to 5
   Index.1.Initiate
   Dwell For Time 1.000 'seconds
Next
```

Example:

```
For Count = 1 To 10
   Wait For ModuleInput.1 = ON
   Index.0.Initiate
   Wait For Index.AnyCommandComplete
   ModuleOutput.1=ON
   Wait For Time 1.000 'seconds
   ModuleOutput.1=OFF
Next
```

Formula

This program instruction can be used to enter a formula or assignment into a program. All PTi210 parameters are available for use in a formula. They may be dragged and dropped into a formula, but the program User Level will determine how many appear for dragging and dropping (see *User Levels* on page 14). Formulas can also be created by simply typing them into the program. This instruction was created to inform the user that formulas can be used in a program.

Examples:

```
Index.1.Vel = 20.0
Index.0.Dist = Index.2.Dist + 0.1
ModuleOutput.1 = ON
Index.0.Accel = (Index.0.Accel*1000)+5.00
```

For<var>

This program instruction is used to execute a section of code as long as the end number is not reached.

For <variable> = <start> to <end> step <increment> ... Next

- <variable> = A single user variable that will be set and incremented by this instruction.
- <start> = A value that the <variable> will be initialized to at the start of the instruction.
- <end> = A number that is used to exit the For<var> loop. For a positive increment, when <variable> is Great than or equal to <end> the loop is exited. For negative increment, when <variable> is Less than or equal to <end> the loop is exited.
- <increment> = A value that the <variable> will be incremented by at the end of the loop. Expression is a mathematical formula including variables, numbers, and math operators that represent a single value when executed.

GoTo

The GoTo instruction is used in conjunction with the Label: instruction to cause program flow to transfer to a specified location within a program. The destination label is allowed to be above or below the GoTo instruction within the same program. It is not possible to GoTo a label outside of the program containing the GoTo instruction, nor is it possible to use a GoTo/Label: to exit out of a For Count/Next loop. In either of these conditions, a Red Dot error will be generated.

The Label to which program flow transfers is a character string up to 50 characters in length and can be made up of any alphanumeric character. The label name must not start with a number, and must end with a colon character ":".

Labels are not case sensitive.

Example:

```
Do While (TRUE)
   If (ModuleInput.1 = ON) Then
   GoTo RunIndex1: 'Go to RunIndex1 label
Else
   GoTo RunIndex2: 'Go to RunIndex2 label
EndIf

RunIndex1:
Index.1.Initiate
GoTo EndLoop:

RunIndex2:
Index.2.Initiate

EndLoop:
Wait For Index.AnyCommandComplete
Loop
```

See the Label: instruction for additional examples.

GraphRun

This instruction allows you to enable the graph from a user program. It is similar to pressing the "Run" button on the Graph view. The instruction will wait until the full graph buffer is full, then it exits into the looking for trigger state.

If/Then/EndIf

This is a program flow control instruction used to selectively run a section of code only if a logical test condition is true. If the test evaluates to true the code between the If/Then and Endif lines is executed. If the test evaluates to false the code is not executed and the program skips to the next line of code after the Endif.

Logical tests (AND, OR, NOT) can be used in the If/Then/Endif instruction. Parenthesis "()" can be used to group the logical tests.

Example:

```
If ModuleInput.1=ON Then
   ModuleOutput.1=ON
   ModuleOutput.2=OFF
Endif
```

Example:

```
If (ModuleInput.1=ON AND ModuleInput.2=OFF) Then
   ModuleOutput.1=ON
   ModuleOutput.2=OFF
Endif
```

Example:

```
If (DriveInput.4=OFF) Then
   Jog.0.PlusInitiate 'Vel=20in/s
   Wait For DriveInput.4=OFF
   Jog.Stop
Endif
```

Example:

```
If (NOT DriveInput.5=ON) Then
   Jog.0.MinusInitiate 'Vel=20in/s
   Wait For DriveInput.5=OFF
   Jog.Stop
Endif
```

Label:

The Label: instruction is used in conjunction with the GoTo instruction to cause program flow to transfer to a specified location within a program. The destination label is allowed to be above or below the GoTo instruction within the same program. It is not possible to GoTo a label outside of the program containing the GoTo instruction, nor is it possible to use a GoTo/Label: to exit out of a For Count/ Next loop. In either of these conditions, a Red Dot error will be generated.

The Label to which program flow transfers is a string of up to 50 characters in length and can be made up of any alphanumeric character. The label name must not start with a number, and must end with a colon character ":". When using the Label: instruction, a ":" will be automatically inserted for the user.

Labels are not case sensitive.

Example:

```
Start:
Index.1.Initiate
Wait For Index.AnyCommandComplete
If (ModuleInput.2 = ON) Then
   GoTo Start:
EndIf
ModuleOutput.1 = ON
End
```

See GoTo instruction for additional examples.

Menu

The Menu instruction is used to read from or write to any host drive menu parameter. The format used is as follows:

Menu.<m>.<p>

Where <m> represents the host drive menu number (0 to 99) and <p> represents the parameter number (0 to 255) within that menu.

NOTE

Leading zeros may be omitted from both the menu and parameter values, e.g. Pr **20.021** can be expressed as either ‘20.21’ or ‘20.021’, similarly Pr **01.015** can be expressed as either ’01.015’, or ‘1.015’, or ’01.15’, or ‘1.15’.

Examples:

Menu.1.15 = 1 ‘Set Preset Selector (Pr **01.015**) to 1 (Preset Reference 1)

Var1 = Menu.1.021 ‘Read Preset Reference 1 (Pr **01.021**) into Var1

Slot

The Slot instruction is used to read from or write to any host drive menu parameter or option slot menu parameter. The format used is as follows:

Slot.<s>.<m>.<p>

Where <s> represents the target destination slot position number (see below), <m> represents the target destination menu number (0 to 99) and <p> represents the parameter number (0 to 255) within that menu.

NOTE

Leading zeros may be omitted from the values, e.g. Drive Pr **20.021** can be expressed as either '0.20.21' or '0.20.021', similarly Pr **01.015** can be expressed as either '0.01.015', or '0.1.015', or '0.01.15', or '0.1.15'.

Examples:

Slot.0.1.15 = 1 'Set Drive Preset Selector (Pr **01.015**) to 1 (Preset Reference 1)

Slot.3.10.11 = 1 'Set Digitax HD Easy Mode Tx1 Link Profile (Pr **10.011**) to 1 (Sync)

Var1 = Slot.0.1.021 'Read Preset Reference 1 and copy value into Var1

Var2 = Slot.3.10.4 'Read Digitax HD Easy Mode Cyclic Message Rate and copy into Var2

The possible values for the Slot position number are shown in the following table.

Slot Position Number	Destination
0	Host drive parameter
1	Slot 1 option module parameter
2	Slot 2 option module parameter
3	Unidrive M70X: Slot 3 option module parameter Digitax HD: Factory Fit Ethernet parameter
4	Unidrive M70X: Factory Fit Ethernet parameter Digitax HD: Not used
5 to 9	Not used

Switch

The Switch instruction is used in conjunction with the Case: instruction to cause program flow to transfer based on the switch expression value and execute the instructions associated with that case instruction.

Example:

```
Switch<var.var0>
case <1>
   Home.0.Initiate        'Home0,Sensor,SpecifiedOffset=0.0000 revs,Vel=200 revs/m
break
case <2>
   Dwell For Time 60 'seconds
break
Default:
   Index.0.Initiate'Index0,Incremental,Distance-300 revs,Vel=2000 rev/m
Wait For Index.AnyCommandComplete
EndSwitch
```

In the above example if var.var0 = 2 then the program will execute from "case 2" and process the instructions of case 2 and proceed through the program.

Wait For

This program flow instruction is used to halt program execution until an expression becomes true. Once the expression becomes true the program continues on with the next line of code.

Logical tests (AND, OR, NOT) can be used in the Wait For instruction. Output events (EZ=ON, AtVel, etc.) as well as comparisons (PosnFeedback > 1234, VelFeedback < 100, etc.) can be used in a Wait For instruction.

Example:

```
Wait For (ModuleInput.1=ON AND ModuleInput.2=OFF)
Index.0.Initiate
```

Example:

```
Wait For Index.AnyCommandComplete
If (ModuleInput.2=ON) Then
   Jog.0.PlusInitiate 'Vel=20in/s
   Wait For ModuleInput.2=OFF
   Jog.Stop
Endif

If (DriveInput.6=ON) Then
   Jog.0.MinusInitiate 'Vel=20in/s
   Wait For DriveInput.6=OFF
   Jog.Stop
Endif
```

Example:

```
Wait For (MasterAxis.PosnFeedback > 1000.00)
ModuleOutput.1 = ON
```

Example:

```
Wait For (VelFeedback > 50.00)
ModuleOutput.2 = ON
```

Wait For Control Loop

This program instruction is used to halt the program execution until the next control loop. The control loop processes the input and output events. So the “Wait For Control Loop” is very useful to allow event to be processed before using the results or clearing it’s activation request.

Examples:

```
DefineHome=true
Wait For Control Loop
DefineHome=false
```

Wait For Time

This program instruction is used to halt program execution for a specified period of time. This instruction is not a motion instruction and can be used while a motion instruction is executing.

Units: Seconds, Resolution: 0.001 seconds

A comment is automatically inserted after the "Wait For Time" instruction which notes that the time is in units of seconds.
The comment starts with the apostrophe ' character.

The accuracy of the Wait For Time instruction is dependant on the Trajectory Update Rate found on the Setup view. The actual accuracy for this instruction is +/- 1 Trajectory Update. Therefore, if the Trajectory Update Rate is set to 2msec, then the worst-case accuracy is 2msec.

Example:

```
Wait For Time 5.000 'seconds
Do While (TRUE)
   Index.1.Initiate
   Wait For AtVel
   ModuleOutput.1=ON
   Wait For Time 1.000 'seconds
   ModuleOutput.1=OFF
   Wait For Index.AnyCommandComplete
Loop
```

9.8.2 Program Math Functions

ArcSin

This trig function can be used in formulas from within a program. Example: `var.var0 = ArcSin(var.var1).`
Returns the trigonometric ArcSin in degrees. The ArcSin is the angle whose Sine is the given number.

ArcTan

This trig function can be used in formulas from within a program. Example: `var.var0 = ArcTan(var.var1).`
Returns the trigonometric ArcTan in degrees. The ArcTan is the angle whose Tan is the given number.

ArcCos

This trig function can be used in formulas from within a program. Example: `var.var0 = ArcCos(var.var1).`
Returns the trigonometric ArcCos in degrees. The ArcCosine is the angle whose cosine is the given number.

Cos

This trig function can be used in formulas from within a program. Example: `var.var0 = Cos(var.var1).`
Returns the trigonometric cosine in degrees. Cos(x) x is in degrees and accurate to 6 decimal places.

Modulus

Returns the remainder (Modulus) resulting when a numerator is divided by a denominator. The result has the same sign as the denominator. The floating-point operators are NOT rounded to integers as would be in the Mod operator.

The exact mathematical function for the Modulus function is as follows:

`Modulus(x,y) = x - [(FLOOR (x/y)) * y]`

Where FLOOR is defined as rounding the argument down to the next whole integer value towards negative infinity.

Example: FLOOR(-3.5715) = -4 or FLOOR(3.5715) = 3

The FLOOR function itself is not available to the user within a user program.

Example 1: `Modulus(5,1.4) Returns 0.8`

Example 2: `Modulus(5,-1.4) Returns -0.6`

Example 3: `Modulus(-5,1.4) Returns 0.6`

Example 4: `Modulus(-5,-1.4) Returns -0.8`

Sin

This trig function can be used in formulas from within a program. Example: `var.var0 = Sin(var.var1).`
Returns the trigonometric sine in degrees. Sin(x) x is in degrees and accurate to 6 decimal places.

Tan

This trig function can be used in formulas from within a program. Example: `var.var0 =Tan(var.var1).` Returns the trigonometric tangent in degrees. Tan(x) x is in degrees and accurate to 6 decimal places.

9.8.3 Program Array Access

Cam[ ].Follower

This instruction allows the user to write a value to a specified cam table follower element number. In the following example Cam table 1, follower element 2 will be changed to the value of 3.

Example:

```
Cam[1,2].Follower =3
```

Cam[ ].Interpolation

This instruction allows the user to write a value to a specified cam table follower element number. In the following example Cam table 1, Interpolation element 2 will be changed to the value of 3.

Example:

```
Cam[1,2].Interpolation =3
```

Cam[ ].Master

This instruction allows the user to write to a specified cam table master element number. In the following example: Cam table 0, master element 3 will be changed to 7.

Example:

```
Cam [0,3].Master=7
```

9.8.4 Motion Instructions

Dwell For Time

This motion instruction is used to pause program execution for a very precise amount of time.

It operates as a motion instruction - similar to an index, home or jog. Like all other motion instructions it will not start until the preceding motion instruction has completed. A "Wait for Index.AnyCommandComplete" is not required. Likewise, any subsequent motion commands will wait and start after the dwell has completed. The total time required to complete a sequence of indexes and "Dwell For Time" instructions is extremely repeatable.

The "Dwell For Time" instruction is in units of seconds with a resolution of milliseconds (0.000 seconds).

If you want to pause the program while an index is executing you should use a Wait for Time instruction described below.

A comment is automatically inserted after the Dwell For Time instruction that notes that the dwell time is in units of seconds. The comment starts with the apostrophe ' character.

Example:

```
Do While (TRUE)
   Index.0.Initiate'Incremetal,Dist=25.000in,Vel=25in/s
   Dwell For Time 1.000 'Seconds
Loop
```

Example:

```
Do While (TRUE)
   Index.0.Initiate
   Dwell For Time 1.000 'Seconds
   Index.1.Initiate
   Dwell For Time 0.500 'Seconds
Loop
```

Dwell For MasterDist

This motion instruction is used to pause program execution for a precise change in distance on the master encoder signal. This is typically used in synchronized motion applications. This dwell does not begin until all other motion has completed. When the dwell begins, program flow will wait until the specified master distance has passed. The units for the dwell value are specified in the Master Units View.

Example:

```
Do While (TRUE)
   Index.0.Initiate
   Dwell For MasterDist 12.00 'MstrInch
Loop
```

IndexInitiate by Expression

This motion instruction is used to initiate a single index. The index is preset to include an acceleration up to speed, a run at speed and a deceleration to a stop. IndexInitiate by Expression is used to initiate different indexes with the same line of code in a program.

One notable change from a standard Index.#.Initiate is that Wait for Index.AnyCommandComplete line of code normally inserted after the initiate will not be inserted after IndexInitiate by Expression. No comments will be added to this instruction as the index selected can change anytime before the initiate command is encountered.

The following example will initiate index.0, wait for complete, initiate index.1, wait for complete, index.2... etc.

Example:

```
var.var0 = 0
a:
IndexInitiate var.var0
Var.var0 = 1 + var.var0
Wait for Index.AnyCommandComplete
goto a:
```

CompoundIndexInitiate By Expression

This motion instruction is used to vary the index numbers making up a compound index. No comments will be added to this instruction as the index selected can change anytime before the initiate command is encountered.

The following code will continuously compound Initiate Index.0 and Index.1 in a loop.

Example:

```
a:
if var.var0 = 0 then
var.var0 = 1
else
var.var0 = 0
endif
CompoundIndexInitiate var.var0
goto a:
```

CamInitiateByExpression

This motion instruction is used to initiate a cam. The cam is preset to include cam type, time base, direction, any chaining requirements, and a deceleration to a stop, optional. CamInitiate by Expression is used to initiate different Cams with the same line of code in a program.

The following example will initiate Cam Table 0, wait for cam complete, initiate Cam Table.1, wait for cam complete, index.2... etc.

Example:

```
var.var0 = 0
a:
CamInitiate var.var0
Var.var0 = 1 + var.var0
Wait For Cam.CommandComplete
goto a:
```

Cam.Initiate

Cam.#.Initiate has two forms - Program Instruction and Assignment (as a Destination). Both are used to initiate a specific Cam. For the Destination, the Cam is initiated on the rising edge of this event. Using the Destination, a Cam cannot be initiated if there is an Index, Home, Jog, or Program in progress, or if the Cam.#.Initiate is an instruction. If any motion is active, the program will hold on this instruction until that motion is complete (unless it is run on a different profile).

Cam.Resume

Resumes the cam execution from a Cam.Suspend command (or SetCamMasterOffset(MasterPosn), or SetCamFollowerOffset (FollowerPosn). The cam points are all relative to the start of the Cam table. On Cam.Resume the current physical position becomes the cam start point and the cam aligns it's resume position to the current physical position without any physical movement.
This means if you suspend a cam and move the physical position; for example with a Jog, you will need to position the motor back to the desired position before resuming.

Cam.Suspend

When Cam.Suspend is activated the current cam will stop using the Cam.Decel ramp rate (if Cam.DecelEnable is enabled). The cam will accept a Cam.Resume after a suspend. The suspend records the master and follower positions at the point of the suspend for future Cam.Resume execution.

Cam.Stop

When Cam.Stop is activated the cam will stop using the Cam.Decel ramp rate (if Cam.DecelEnable is enabled).

Gear.Stop

Gear Stop will stop gearing motion that has been initiated from a program.

Example:

```
Gear.Initiate
Wait for ModuleInput.2=ON
Gear.Stop
```

Gear.Initiate

Gear Initiate will initiate gearing from a program. Gearing will remain active until the Gear.Stop command is used.

Example:

```
Gear.Initiate
Wait for ModuleInput.2=ON
Gear.Stop
```

This instruction should not be used within a loop in the user program. If the Gear.Initiate instruction is processed while gearing is active, the program will hang on this instruction.

Home.Initiate

This program instruction is used to initiate the home.

A comment is automatically inserted after the Home.Initiate instruction that shows key data about the particular home. The comment starts with the apostrophe ' character. A "Wait For Home.AnyCommandComplete" instruction is not required because the home is actually a program which already has a "Wait For" instruction.

Index.Initiate

This program instruction is used to initiate a single index. The index is preset to include an acceleration up to speed, a run at speed and a deceleration to a stop.

A comment is automatically inserted after the index instruction which shows key data about the particular index. The comment starts with the apostrophe 'character.

A Wait For Index.AnyCommandComplete instruction is also automatically inserted after each index. This insures that the index has completed before the program continues on to the next line of code. It is also possible make the program wait until the index is complete and the following error is less than a specified amount. This is accomplished by changing the Wait For Index.AnyCommandComplete instruction to Wait For InPosn. The In Position Window is configured in the Position view.

Example:

```
Index.0.Initiate
Wait For Index.AnyCommandComplete
```

Example:

```
Index.37.Initiate
Wait For InPosn
```

Index.CompoundInitiate

This program instruction is used to initiate an index which has no deceleration ramp. The index accelerates or decelerates towards the next index velocity using the next index acceleration ramp. The index will finish at velocity. The program then moves on to the next index. It smoothly transitions into the second index without stopping. The second index then ramps to its pre-configured velocity. Multiple indexes can be "compounded" to create a complex velocity profile. The last index in a complex profile must have a deceleration ramp.

This is accomplished using a standard Index.Initiate rather than a Index.CompoundInitiate.

The final index will honor the deceleration ramp. If the last index is not long enough to perform a decel ramp at the programmed rate, the motor will backup at the end of the last index.

Each index can be used in multiple places as both a standard index with a deceleration ramp, and a compound index without a deceleration ramp. The program instruction (Index.0.Initiate or Index.0.CompoundInitiate), not the index itself, determines whether or not the index will execute a deceleration ramp. For example, Index.0 can be used multiple times in multiple programs. It can be initiated at different times using the Index.0.Initiate instruction and the Index.0.CompoundInitiate instruction.

A comment is automatically inserted after the index instruction that shows key data about the particular index. The comment starts with the apostrophe ' character.

Example:

```
Index.0.CompoundInitiate
Index.1.CompoundInitiate
Index.2.Initiate
Wait For Index.AnyCommandComplete
Index.0.CompoundInitiate
DriveIO.1.OUT=ON
Index.1.CompoundInitiate
DriveIO.2.OUT=ON
Index.2.Initiate
DriveIO.3.OUT=ON
Wait For Index.AnyCommandComplete
DriveIO.1.OUT=OFF
DriveIO.2.OUT=OFF
DriveIO.3.OUT=OFF
```

Index.BlendInitiate

This program instruction is used to allow an index to complete its move at the velocity of another index. A blended index accelerates towards its index velocity using its accel ramp. The index will run at velocity before using its deceleration ramp to accelerate or decelerate towards the velocity of the next index specified. This differs from the compound index where the index finishes using the accel ramp of the next index.

The index that is to be "blended into" is on the command line in parenthesis immediately after the Index.BlendInitiate command.

Index.0.BlendInitiate into (1)

This command will cause index zero to finish at the velocity of index 1. The value within the parenthesis can also be a variable. The following example will operate the same as the previous.

Index.0.BlendInitiate into (var.var0)

The next index that is to be blended must:

- ·Exist
- ·Have the same time base as the present index (synch versus real time)

If the index does not exist or the time base is different, the blended index will convert into a regular compound index.

The direction of the next index (blended into index) is not looked at. Hence, blending an index into another index will not cause the index to cross through zero velocity.

Example:

```
Index.0.BlendInitiate into (1)
DriveIO.1.OUT=ON
Index.1.BlendInitiate into (2)
DriveIO.2.OUT=ON
Index.2.Initiate
DriveIO.3.OUT=ON
Wait For Index.AnyCommandComplete
DriveIO.1.OUT=OFF
DriveIO.2.OUT=OFF
DriveIO.3.OUT=OFF
```

Index.Stop Tracking

This command is used to cancel the position tracker continuous index once it has been initiated within the program using the Index.#.Initiate program command. If the Index.#.Initiate input function starts the position tracker index, then Index.#.StopTracking will not stop the tracking index.

Example:

Home.0.Initiate'Sensor,Offset=2.000in,Vel=-10.0in/s

Jog.Stop

This program instruction is used to halt jogging using the deceleration ramp setup for the currently operating jog.

Examples:

```
Wait For ModuleInput.2=ON
Jog.0.MinusInitiate
Wait For ModuleInput.2=OFF
Jog.Stop
Do While(TRUE)
   If (ModuleInput.2=ON) Then
           Jog.0.PlusInitiate
           Wait For ModuleInput.2=OFF
           Jog.Stop
   EndIf

   If (ModuleInput.3=ON) Then
           Jog.0.MinusInitiate
           Wait For ModuleInput.3=OFF
           Jog.Stop
   EndIf
Loop
```

Jog.PlusInitiate

This program instruction is used to initiate jogging in the positive direction. The Jog.Stop instruction is used to stop jogging motion.

A comment is automatically inserted after the Jog.PlusInitiate instruction which shows key data about the particular jog. The comment starts with the apostrophe ' character.

Examples:

```
Jog.0.PlusInitiate 'Vel=27.2in/s

Jog.1.PlusInitiate 'Sync,Vel=1.000in/in
```

Jog.MinusInitiate

This program instruction is used to initiate jogging in the negative direction. The Jog.Stop instruction is used to stop jogging motion.

A comment is automatically inserted after the Jog.MinusInitiate instruction which shows key data about the particular jog. The comment starts with the apostrophe ' character.

Examples:

```
Jog.0.MinusInitiate 'Vel=27.2in/s

Jog.1.MinusInitiate 'Sync,Vel=1.000in/in
```

Program.ProgramStop

The Program.ProgramStop instruction is used to stop processing a specific program. The ProgramStop instruction can be used in a program to stop itself or any other program. The ProgramStop instruction DOES NOT stop the motion that has been initiated by the program being stopped. Either the MotionStop or the Profile.Stop instruction must be used to stop motion from a program.

If the ProgramStop instruction is used to stop a program that is not running, the Stop will be issued, but will be ignored.

If the program that is stopped was called by another program, the call is killed. This means that program flow will not return to the program that originally called the program that was stopped. For example, Program 1 calls Program 2. While Program 2 is running, Program 3 (which is running on a different task) issues a Program.2.ProgramStop command. If Program 2 ended because of normal conditions (i.e. the "End" instruction), then program flow would return back to Program 1. Because Program 2 was terminated using the Program.Stop instruction, program flow does not return to Program 1.

The ProgramComplete signal will not activate if a program has been stopped using the Program.ProgramStop instruction.

Profile.ProfileStop

Profile.ProfileStop is used to stop motion on a single profile, without stopping the other profiles.The user must specify the profile they with to stop (i.e.0 or 1).

Example:

```
If (DriveInput.4-ON) Then
   Profile.1.ProfileStop
EndIf
```

Program.Initiate

This instruction allows the user to start another program from within a program. This is different from a Call Program instruction because the program this instruction is in does not stop when the other program starts. Therefore the program that is initiated must be on a different task. See *Program Multi-Tasking* on page 161.

Example:

```
Program.2.Initiate
```

9.8.5 Motion Modifier Instructions

On Profile

The On Profile instruction can be inserted after any motion type Initiate, Dwell for Time, or Dwell for Master Dist instructions. By inserting the On Profile modifier, it specifies which Profile the instruction will run on (See Multiple Profiles section for more information). Select from Profile 0 or Profile 1. Both Profiles sum to give a single commanded position and commanded velocity. If no On Profile modifier is used, the motion/dwell will operate on Profile 0. All motion that is initiated from the assignments screen operates on Profile 0.

The On Profile modifier is also used with the Jog.Stop and Gear.Stop. When stopping jog or gear motion that is operating on Profile 1, the On Profile.1 modifier must also be used on the stop instruction.

Example:

```
Index.0.Initiate
Index.1.Initiate On Profile.1
Gear.Initiate On Profile.1
Wait For ModuleInput.3 = ON
Gear.Stop On Profile.1
```

Example:

```
Jog.0.PlusInitiate On Profile.1
Index.0.Initiate
Wait For Index.AnyCommandComplete
Wait For ModuleInput.2 = ON
Jog.Stop On Profile.1
```

Using Capture

The Using Capture instruction can be inserted after any Jog Initiate, Index Initiate, Dwell for Time, and Dwell for Master Dist instructions. By inserting the Using Capture instruction, it specifies that data captured by the position capture object is to be used as the starting point for the motion initiate. If the motion time base is realtime, then the captured time is used as the starting point for the motion profile. If the motion time base is synchronized, then the captured master position is used as the starting point for the profile.

Example:

```
Wait For (Capture.0.CaptureTriggered)
Index.0.Initiate Using Capture.0
```

Using Last

When the Using Last instruction is inserted after a motion initiate instruction, the time (or master position in synch motion) of the last command complete is used as the starting point of the motion profile. Whenever a motion profile is complete, the time/position is automatically captured behind the scenes. The Using Last instruction simply references this "automatically" captured time or position.

The PTi210 module performs motion based on a concept called the timeline. The timeline allows for accurate and repeatable motion with respect to a single point in time. The timeline guarantees that all motion profiles occur at the right time with respect to each other.

If Index0 takes 3 seconds to complete, and Index1 takes 5 seconds to complete, by initiating Index0 and then Index 1 in a program, the user would expect these profiles to take a total of 8 seconds to complete. It is possible though, that because of processor timing, Index.1 does not start at exactly the same time Index0 is complete. Therefore, the two profiles could take slightly more than 8 seconds to complete. Although the amount of time lost is extremely small (less than 5 milliseconds), over a long period of time, this lost time can accumulate.

Keeping the timeline intact is most important in applications using synchronized motion. This is because in synchronized motion, time is replaced by master encoder motion. If time is lost in a synchronized motion application, then master distance is lost, and the follower position is off with respect to the master.

Example:

```
Index.0.Initiate
Dwell For Time 1.000 Using Last
Index.1.Initiate Using Last
```

9.9 Red Dot Error Bar

The Red Dot Error Bar is used to display if an error exists on a particular line of code in the user program. PowerTools Studio uses a program parser that reads the user program, and translates it directly into the language that the PTi210 module processor understands. If the parser detects a mistake (i.e. syntax error) in an instruction, a Red Dot will appear next to that line of code in the Red Dot Error Bar.

To find out what the error is, PowerTools Studio has a utility called Red Dot Help. To use Red Dot Help, click the ***Red Dot Help*** button on the program toolbar. This will place PowerTools Studio into "Red Dot Help Mode". While in this mode, the mouse pointer should

have a red and yellow question mark next to it. After going into Red Dot Help Mode, simply click on the line of the program that has a red dot next to it. Some description of the error should appear that helps indicate what the error is, or how to fix it. When finished checking Red Dot Errors, click ***Red Dot Help*** button on the program toolbar to exit Red Dot Help Mode.

A user program that has a red dot on an empty line often indicates that the second part of a two-part instruction is missing (i.e. Do While with a missing Loop, If with a missing EndIf, etc.).

If a variable or parameter name is spelled incorrectly, the Red Dot Help message will indicate "XXXX" Error - Couldn't find Variable named from Text" (where XXXX is the parameter name typed by the user). If this message is displayed, make sure that the parameter names on the specific line are spelled correctly.

9.10 Program Code Window

The Program Code Window is where the program instructions are inserted to create the user program.

Program instructions can be inserted into the Program Code Window using either one of two methods. The two methods available are drag & drop or type the instructions into the window.

Drag & Drop Method

The most popular method for inserting program instructions is to drag the instruction from the instruction list and drop it into the program window. To drag & drop an instruction, position the mouse pointer over the instruction to be inserted. Press and hold the left mouse button. While holding the left button, place the mouse pointer on the line of the program that the instruction is to be inserted on. Release the left button to insert the instruction.

Typing Method

Once users become more familiar with the syntax of the PTi210 module programming language, they often prefer to type instructions in. To do this, simply click the left mouse button on the line of the program where instruction is to be inserted. Then simply type the instruction on that line, being sure to use proper syntax.

The user can enter program code using both methods interchangeably.

9.11 Program Blocking

A user program (or task) can be blocked from operation for a period of time. When a program or task is blocked, execution is simply passed on to the next task. The following program instructions will cause a program to be blocked:

```
Index.#.Initiate
Home.#.Initiate
Jog.#.PlusInitiate
Jog.#.MinusInitiate
Dwell For Time
Dwell For Master Dist
Wait For Time
Wait For (XXXX)
```

The motion instructions block processing only if they attempt to initiate motion on a profile that is already performing a motion profile. For instance, if a program initiates Index0 and the next program instruction initiates Index1. The program will be blocked until Index0 is complete because Index1 cannot start until motion on that profile is finished.

```
Index.0.Initiate
Index.1.Initiate
ModuleOutput.2 = ON
```

A Dwell instruction is also a motion instruction and can block the program in the same way.

```
Index.0.Initiate
Dwell For Time 0.550 'sec
ModuleOutput.2 = ON
```

The Dwell cannot start until other motion on the same profile is complete, and therefore the program (or task) is blocked until Index0 is finished.

The Wait For instruction will block the program until the Wait For condition is satisfied. The Wait For condition does not have to be TRUE at the exact time the task is processed. If the Wait For condition is satisfied at any time (even when that task is not being processed) the task will be scheduled to run the next time through the loop. Figure 9-23 shows the same time-slicing diagram as above, but Task 0 is blocked in this example. Notice how Task 0 is skipped when the processor recognizes the task is blocked and processor execution switches to Task 1.

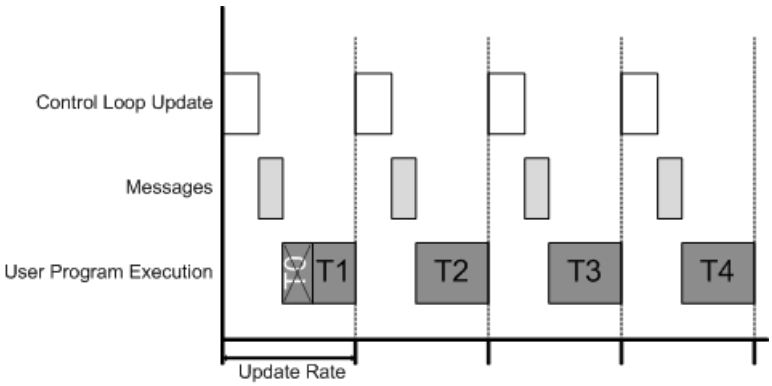


Figure 9-23: Time Slicing Diagram (Task 0 blocked)

The time taken to process the blocked task and pass on to the next available task is between 50 and 100 microseconds.

Figure 9-24 is a flowchart that reflects the time-slicing process. It shows the complete loop based on whether Modbus messages need processing and if programs (tasks) are blocked.

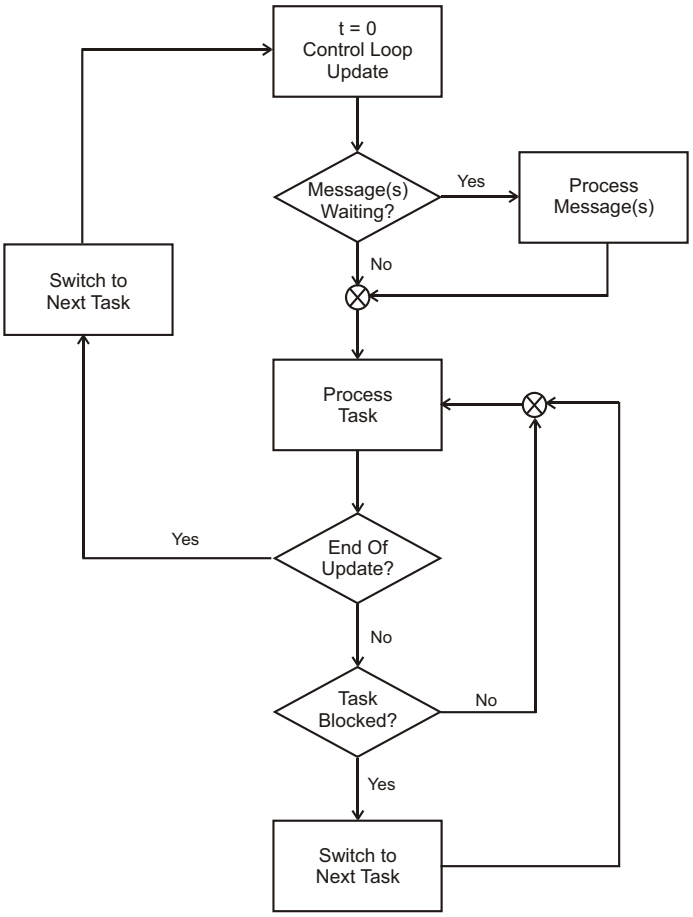


Figure 9-24: Multitasking Flow Chart

9.11.1 Program Math Functions

Cos

This trig function can be used in formulas from within a program.

Example:

```
var.var0 = Cos(var.var1)
```

Returns the trigonometric cosine in degrees. Cos(x) x is in degrees and accurate to 6 decimal places.

Sin

This trig function can be used in formulas from within a program.

Example:

```
var.var0 = Sin(var.var1)
```

Returns the trigonometric sine in degrees. Sin(x) x is in degrees and accurate to 6 decimal places.

Tan

This trig function can be used in formulas from within a program.

Example:

```
var.var0 =Tan(var.var1)
```

Returns the trigonometric tangent in degrees. Tan(x) x is in degrees and accurate to 6 decimal places.

ArcCos

This trig function can be used in formulas from within a program.

Example:

```
var.var0 = ArcCos(var.var1)
```

Returns the trigonometric ArcCos in degrees. The ArcCosine is the angle whose cosine is the given number.

ArcSin

This trig function can be used in formulas from within a program.

Example:

```
var.var0 = ArcSin(var.var1)
```

Returns the trigonometric ArcSin in degrees. The ArcSin is the angle whose Sine is the given number.

ArcTan

This trig function can be used in formulas from within a program.

Example:

```
var.var0 = ArcTan(var.var1)
```

Returns the trigonometric ArcTan in degrees. The ArcTan is the angle whose Tan is the given number.

Modulus

Returns the remainder (Modulus) resulting when a numerator is divided by a denominator. The result has the same sign as the denominator. The floating-point operators are NOT rounded to integers as would be in the Mod operator.

Examples:

Example 1: `Modulus(5,1.4)` Returns 0.8

Example 2: `Modulus(5,-1.4)` Returns -0.6

Example 3: `Modulus(-5,1.4)` Returns 0.6

Example 4: `Modulus(-5,-1.4)` Returns -0.8

The exact mathematical function for the Modulus function is as follows:

```
Modulus(x,y) = x - (FLOOR (x/y)) * y
```

Where FLOOR is defined as rounding the argument down to the next whole integer value towards negative infinity.

Example: `FLOOR(-3.5715) = -4`

The FLOOR function itself is not available to the user within a user program.

10 Starting and Stopping Motion

10.1 Starting Motion

All of the motion types described in *How Motion Works* on page 31 can be initiated using a couple of different methods. The two different methods are from assignments, and from user programs. The following sections details how each of the different motions are initiated from both assignments and from user programs.

10.1.1 From Assignments

In order to initiate motion from an assignment, a Source must be assigned, on the Assignments view in PowerTools Studio, to one of the Destinations listed below. Some Destinations are level sensitive while others are edge sensitive. Each of the Destinations associated with starting motion are listed below.

Jog.PlusActivate

Jog.PlusActivate will, when active, cause the motor to run at a specified Jog Velocity in the positive direction. Jog.PlusActivate is a level sensitive assignment meaning that as long as the Destination is active, the Jog will be active. The status of the Jog.Select destination determines which jog profile is used when Jog.PlusActivate activates. If Jog.Select is off (or inactive) then the Jog 0 profile will be used. If Jog.Select is active, then the Jog 1 profile is used.

To stop the jogging motion, simply deactivate the Jog.PlusActivate Destination. The motor will then decelerate using the selected Jog Deceleration ramp.

If Jog.Select activates or deactivates while Jogging is active, the motor will change to the new Jog Velocity using the new Jog Acceleration ramp. Jog Deceleration is only used when Jogging motion is stopped. Figure 10-1 shows an example of changing Jog.Select while Jog is active.

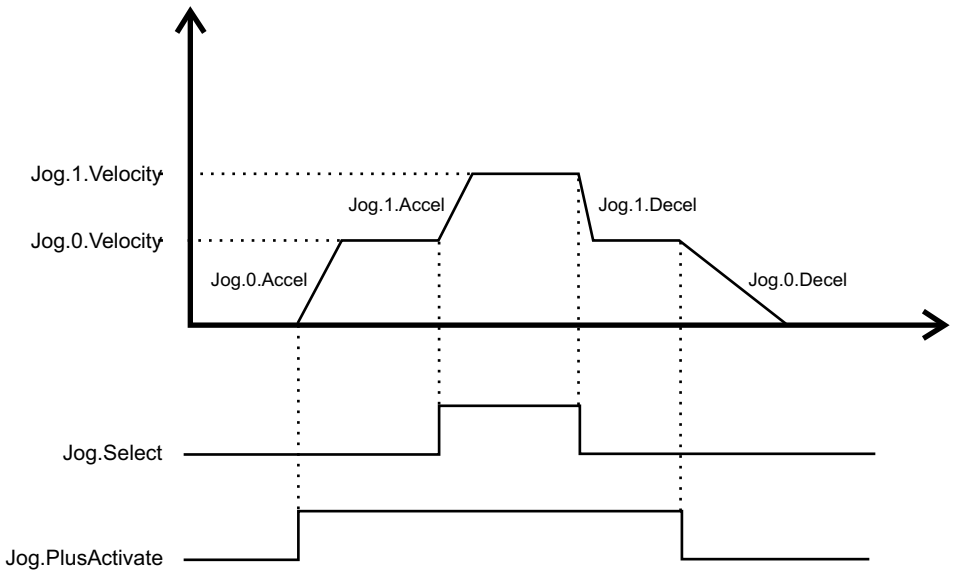


Figure 10-1: Jog Select Example

Jog.MinusActivate

Jog.MinusActivate will, when active, cause the motor to run at a specified Jog Velocity in the negative direction. Jog.MinusActivate is a level sensitive assignment meaning that as long as the Destination is active, the Jog will be active. The status of the Jog.Select destination determines which jog profile is used when Jog.MinusActivate activates. If Jog.Select is off (or inactive) then the Jog 0 profile will be used. If Jog.Select is active, then the Jog 1 profile is used.

To stop the jogging motion, simply deactivate the Jog.MinusActivate Destination. The motor will then decelerate using the selected Jog Deceleration ramp.

If Jog.Select activates or deactivates while Jogging is active, the motor will change to the new Jog Velocity using the new Jog Acceleration ramp. Jog Deceleration is only used when Jogging motion is stopped. Figure 10-1 above shows an example of changing Jog.Select while Jog is active.

Jog.Select

Jog.Select is used solely to determine which Jog Profile is selected when the Jog.PlusActivate or Jog.MinusActivate are activated. To select Jog 0, Jog.Select should be inactive. To select Jog 1, Jog.Select should be active.

Home.#.Initiate

The Home.#.Initiate Destination is used to start the home sequence as defined on the Home view. Home.#.Initiate is an edge-sensitive assignment meaning that the home will start on the rising edge of the Source signal. If the Home.#.Initiate Destination is held active, the Home will not initiate again until the next rising edge.

Index.#.Initiate

The Index.#.Initiate Destination is used to start the specified index instance. When the Source assigned to the Index.#.Initiate Destination activates, the index will accelerate to the specified index velocity. If the Index.#.Initiate Destination is held active, the Index will not initiate again until the next rising edge.

Gear.Activate

Gear.Activate will, when active, cause the motor to run at the specified Gear Ratio. The direction that the motor turns is dependent upon the direction the master axis is traveling, and the specified ratio. Gear.Activate is a level sensitive assignment meaning that as long as the Destination is active, the Gear will be active.

To stop the gearing motion, simply deactivate the Gear.Activate Destination. The motor will ramp to zero velocity if deceleration is enabled, or it will attempt to stop the motor without acceleration within one update if deceleration is not enabled.

10.1.2 From Programs

The various motion types function the same whether they are initiated from an assignment, or from a user program. In certain cases however, the syntax used to initiate the motions from a program are slightly different than from an assignment. Following is a list of instructions used to initiate motion types from a user program.

Jog.#.PlusInitiate

Jog.#.PlusInitiate is used in a program to initiate jogging motion in the positive direction. The user must specify which jog is to be used in the instance location (# should be replaced with 0 or 1 for Jog 0 or Jog 1 respectively). Since Jog is level sensitive, the jog will remain active until a Jog.Stop instruction is used in the same program.

If the program ends while jogging motion is active, the jog will automatically be stopped.

Jog.#.MinusInitiate

Jog.#.MinusInitiate is used in a program to initiate jogging motion in the negative direction. The user must specify which jog is to be used in the instance location (# should be replaced with 0 or 1 for Jog 0 or Jog 1 respectively). Since Jog is level sensitive, the jog will remain active until a Jog.Stop instruction is used in the same program.

If the program ends while jogging motion is active, the jog will automatically be stopped.

Home.#.Initiate

Home.#.Initiate is used in a program to initiate the Home sequence. The user must specify which Home is to be initiated in the instance location (# should be replaced with 0 for Home 0) even though there is only one instance available. Since Home is edge sensitive, the Home will initiate only once until the next Home.#.Initiate instruction.

The program will wait on the Home.#.Initiate instruction until the home is complete before moving on to the next line of the program.

Index.#.Initiate

Index.#.Initiate is used in a program to initiate an index profile. The user must specify which Index is to be initiated in the instance location (# should be replaced with the index number). If Index.#.Initiate is held active, the index will initiate only once until the next rising edge of the Destination signal.

The program will wait on the Index.#.Initiate instruction momentarily until the index starts before moving on to the next line of the program.

Index.#.CompoundInitiate

The Index.#.CompoundInitiate instruction is unique to user programs. There is no way to compound indexes using assignments. Compounding indexes is a way to link two or more indexes together so that the motor does not come to a stop in between the given indexes. Instead of stopping between indexes, the initial index will complete its distance at its programmed velocity. Once the initial index distance is complete, the motor will ramp to the secondary indexes' programmed velocity using the secondary indexes' acceleration ramp. Regardless of whether the secondary index velocity is less than or greater than the primary index velocity, the ramp from the primary velocity to the secondary velocity is the secondary indexes' acceleration. In the program, the user can compound together as many indexes as desired. The last index in the group of compounded indexes should use a standard Index.#.Initiate to signify that it should stop at the end of the index.

Figure 10-2 and Figure 10-3 show examples of Compound Indexing.

Example: 1:

```
Index.0.CompoundInitiate 'Dist=10, Vel=100, Accel=100, Decel=100
Index.1.CompoundInitiate 'Dist=20, Vel=500, Accel=5000, Decel=5000
Index.2.Initiate 'Dist=30, Vel=1000, Accel=10000, Decel=10000
```

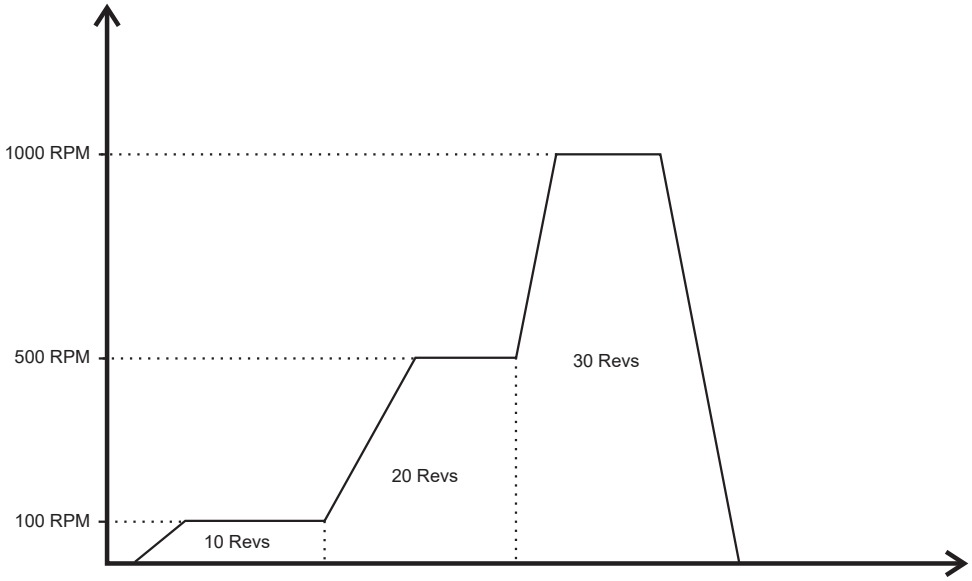


Figure 10-2: Compound Index Example 1

Example: 2:

```
Index.2.CompoundInitiate 'Dist=30, Vel=1000, Accel=10000, Decel=10000
Index.1.CompoundInitiate 'Dist=20, Vel=500, Accel=5000, Decel=5000
Index.0.Initiate 'Dist=10, Vel=100, Accel=100, Decel=100
```

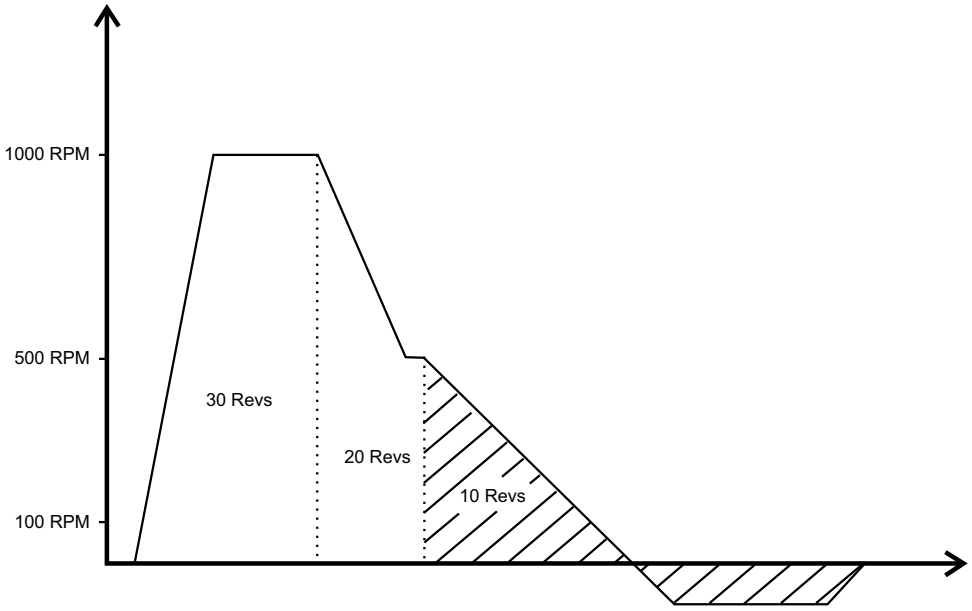


Figure 10-3: Compound Index Example 2

It is important to understand that when compounding indexes together, the acceleration and deceleration will override the index distance. Figure 10-3 above shows the scenario where the final index in the compound string has a short distance and a slow acceleration ramp. Index 0 will start from a velocity of 500 rpm (Index.1.Vel) and accelerate down to its programmed velocity of 100 rpm at a rate of 100 Revs/Min./Sec. The distance required to accelerate from 500 to 100 rpm is greater than the index distance of 10 Revs. The programmed ramp is always maintained, and therefore the final index will travel greater than 10 Revs, and then travel backwards and finally come to a stop 10 Revs from the point that the index started.

Index.BlendInitiate

This program instruction is used to allow an index to complete its move at the velocity of another index. A blended index accelerates towards its index velocity using its accel ramp. The index will run at velocity before using its deceleration ramp to accelerate or decelerate towards the velocity of the next index specified. This differs from the compound index where the index finishes using the accel ramp of the next index.

The index that is to be "blended into" is on the command line in parenthesis immediately after the Index.BlendInitiate command.

```
Index.0.BlendInitiate into (1)
```

This command will cause Index 0 to finish at the velocity of Index 1. The value within the parenthesis can also be a variable. The following example will operate the same as the previous.

```
Index.0.BlendInitiate into (var.var0)
```

The next index that is to be blended into must:

- Exist
- Have the same time base as the present index (synch versus real time)

If the index does not exist or the time base is different, the blended index will convert into a regular compound index.

The direction of the next index (blended into index) is not looked at. Hence, blending an index into another index will not cause the index to cross through zero velocity.

Example: 1:

```
Index.0.BlendInitiate into (1)'Index0,Dist=5, Vel=100, Accel=100, Decel=1000
Index.1.BlendInitiate into (2)'Index1,Dist=20, Vel=500, Accel=5000, Decel=500
Index.2.Initiate'Index2,Dist=25, Vel=1000, Accel=10000, Decel=1000
```

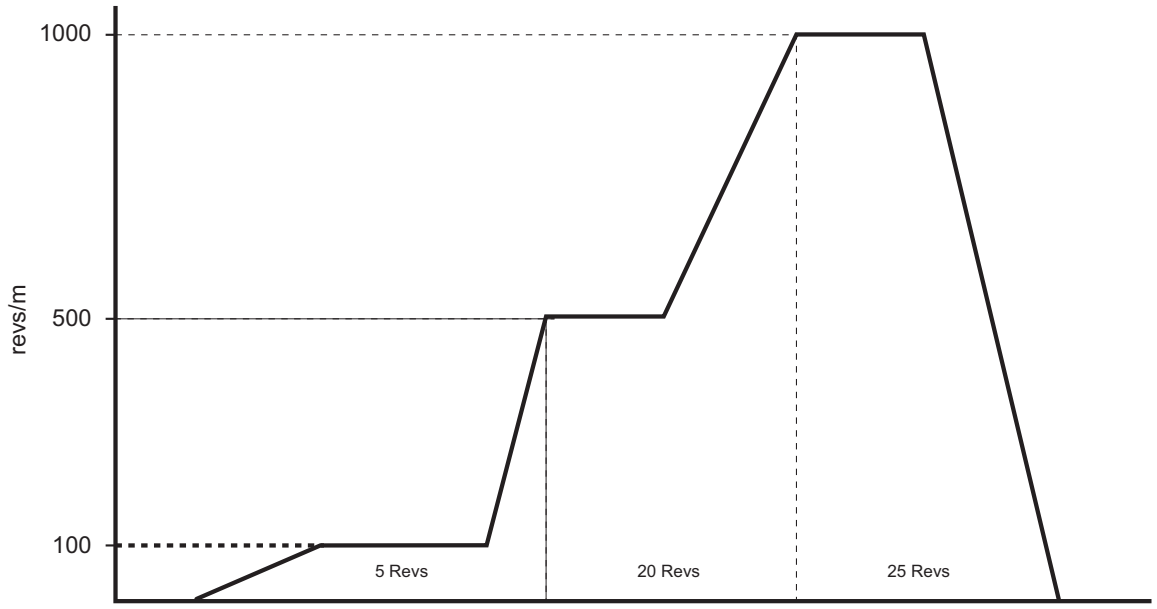


Figure 10-4: Blended Index Example 1

Example: 2:

```
Index.2.BlendInitiate into (1)'Index2,Dist=25, Vel=1500, Accel=10000, Decel=1000
Index.1.BlendInitiate into (0)'Index1,Dist=20, Vel=1000, Accel=5000, Decel=500
Index.0.Initiate'Index0,Dist=5, Vel=500, Accel=1000, Decel=250
```

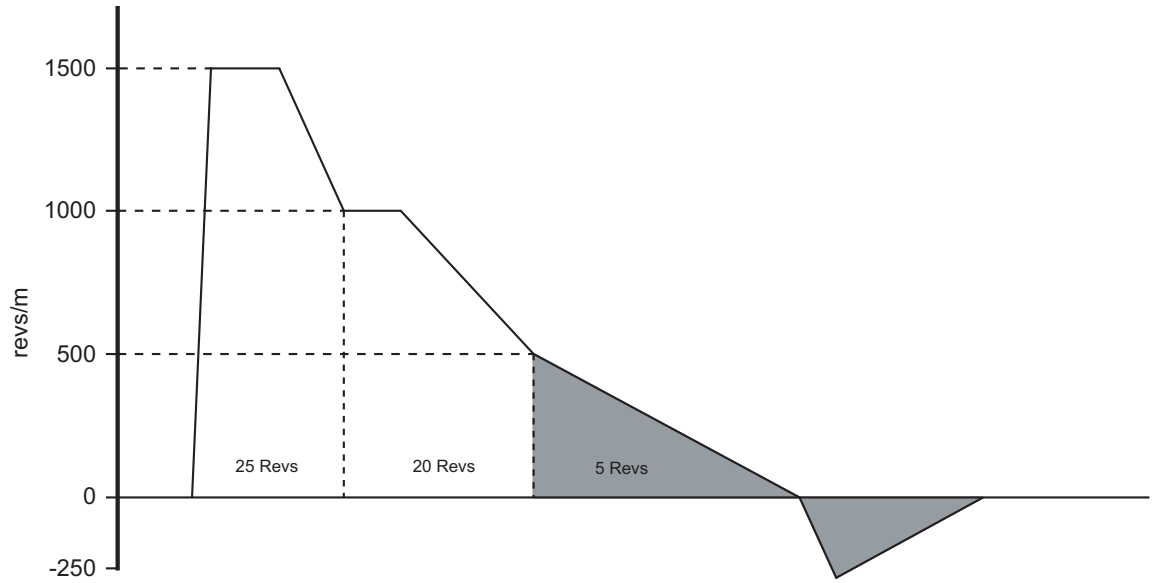


Figure 10-5: Blended Index Example 2

Gear.Initiate

Gear.Initiate is used in a program to initiate Gearing motion. The user must specify the Gear Ratio that will determine the velocity and direction of the motion. Since Gear is level sensitive, the gear will remain active until a Gear.Stop instruction is used in the program.

If the program ends while gearing motion is active, the gear will automatically be stopped.

This instruction should not be used within a loop in the user program. If the Gear.Initiate instruction is processed while gearing is active, the program will hang on this instruction.

10.1.3 From PowerTools Studio

Some views in PowerTools Studio have an Online tab that allow the user to initiate motions while online with the device. The specific views that have Online tabs to initiate motion are the Home view, Jog view, and Index view.

On the Home view Online tab, there is a Start button that will initiate the Home sequence from PowerTools Studio.

On the Jog view Online tab, there are two buttons that will initiate jog motion in the positive and negative directions. The button must be held active to maintain the jog motion. If the button is released, the jog will stop.

On the Index view Online tab, there is a Start button that initiates the specific index that is being displayed in the Index view.

10.2 Stopping Motion

10.2.1 From Assignments

There are numerous different methods that can be used to stop or pause motion using Assignments in the PTi210 module. Listed below are the various assignment Destinations.

Stop

The Stop Destination will, when activated, stop any motion that is active, along with any user programs that are active. Stop is level sensitive, so that while it is active, all motion and programs are prevented from starting. Stop will override any other functions that are active.

MotionStop

The MotionStop Destination will stop all motion that is currently active. It will not stop any user programs. Stop Motion is level sensitive so that all motion will be prevented from starting while Stop Motion is active.

Feedhold

The Feedhold Destination will pause all motion (other than gearing) by effectively bringing time to a halt. Feedhold is level sensitive meaning that Feedhold will be active as long as the Destination is held active. When Feedhold deactivates, the motion will continue.

While Feedhold is active, any active motion profiles do not stop, even though the motor will decelerate to zero speed. Even though the motor is stopped, the motion profile remains active. All of the motion attributes will remain active (i.e. In Progress, At Velocity, Accelerating, Decelerating, etc.). For instance, if Feedhold activates while an index is in progress, the motor will stop, but the Index.#.CommandInProgress and Index.#.AtVel will be active just as though the motor is still moving.

The Feedhold ramp is configured in units of Seconds. The same time and distance will be covered during acceleration back to velocity as traveled during deceleration to zero velocity.

Programs continue to process while Feedhold is active. If a program encounters a motion instruction while Feedhold is active, the program will be stuck on that line of the program until Feedhold is deactivated.

Profile.#.Feedhold

The Profile.#.Feedhold Destination acts the same as Feedhold listed above, however this destination will Feedhold only the motion on a single profile rather than all motion. The profile on which to feedhold motion is specified in the instance segment (# is replaced by 0 or 1 for Profile 0 or Profile 1 respectively).

Profile.#.MotionStop

The Profile.#.MotionStop Destination acts the same as MotionStop listed above, however this destination will stop only the motion on a single profile rather than all motion. The profile on which to stop motion is specified in the instance segment (# is replaced by 0 or 1 for Profile 0 or Profile 1 respectively).

10.2.2 From Programs

Following are the instruction used to stop the various motion types within a user program.

Jog.Stop

The Jog.Stop command is used to stop any jogging motion that is currently active. This command will stop either Jog 0 or Jog 1 without specifying which one to stop.

If jog is active on the alternate profile (Profile 1), then the On Profile.1 modifier must be added to the Jog.Stop command.

If this instruction is used while gear motion is not active, it will be ignored, and the program will move to the next instruction.

Gear.Stop

The Gear.Stop command is used to stop gearing motion that is active.

If gear is active on the alternate profile (Profile 1), then the On Profile.1 modifier must be added to the Gear.Stop command.

If this instruction is used while gear motion is not active, it will be ignored, and the program will move to the next instruction.

Stop

The Stop command can be used in a program to stop all programs and motion that are active at that time. To use this command, Stop must be set equal to ON to activate it (i.e. Stop = ON). This presents a bit of a problem since the program that turns the stop on will also stop, and then there is no way to deactivate the stop. If Stop is activated from a program, it is necessary to create an Assignment to clear the Stop function.

Programs and motion will be prevented from initiating while Stop is active.

Feedhold

The Feedhold instruction functions identically in a program as it does from an assignment. To activate Feedhold from a program, it must be set equal to ON. Feedhold will remain active until it is deactivated from a program (Feedhold = OFF). See Feedhold from Assignments above for further details.

Profile.#.Feedhold

The Profile.#.Feedhold instruction functions identically in a program as it does from an assignment. To activate Profile.#.Feedhold from a program, it must be set equal to ON. Feedhold will remain active on the specified profile until it is deactivated from a program (Profile.#.Feedhold = OFF). See Profile.#.Feedhold from Assignments above for further details.

Profile.#.ProfileStop

The Profile.#.ProfileStop instruction in a program acts much the same as the Profile.#.MotionStop assignment. When the Profile.#.ProfileStop instruction is processed, any motion on the specified profile is stopped using the stop deceleration ramp. This instruction acts like an edge sensitive instruction, therefore after the active motion is stopped, any motion can again be started on the profile. Profile.#.ProfileStop does not prevent then next motion from starting.

10.2.3 From PowerTools Studio

Motion can be stopped while online with PowerTools Studio in a number of different ways. Following is a list of ways to stop motion while online with the device.

Stop All - Main Toolbar

The Stop All button on the PowerTools Studio toolbar can be used to stop all motion and programs that are active. The Stop All can be toggled on and off from the toolbar. Click the button once to toggle the Stop on. Motion will be prevented from running until the button is clicked again.

Feedhold

The ***Feedhold*** button on the PowerTools Studio toolbar can be used to Feedhold all motion that is currently active. The button toggles on and off each time it is clicked. Click the button once to activate Feedhold then click it a second time to deactivate Feedhold.

Stop All - Program Toolbar

Motion and Programs can also be stopped from the Program Toolbar using the ***Stop All*** button. This button is like an edge sensitive function that will not prevent motion or programs from being initiated again.

Pause/Break Key

When online it is possible to stop any motion program from the keyboard on your PC by pressing the Pause/Break key. Pressing Pause/Break is identical to clicking the ***Stop All*** button on the PowerTools Studio toolbar. Once motion has been stopped using the Pause/Break key on the keyboard, it is necessary to press the Pause/Break key again to allow motion.

11 Starting and Stopping Programs

11.1 Starting Programs

The user can initiate programs using multiple different methods. The following section describes how programs can be initiated from Assignments, from Programs, and from PowerTools Studio while online with the system.

11.1.1 From Assignments

In order to initiate a user program from an assignment, a Source must be assigned to the Destination listed below on the Assignments view in PowerTools Studio.

Program.#.Initiate

A rising edge on the Program.#.Initiate Destination will cause the specified program to start. The program number that is to be started is determined by the instance field of the destination (represented by # above). Program.#.Initiate is edge sensitive, so the program will only start once until the next rising edge of the destination.

If a program is already running on the same task as the program being initiated, the Program.#.Initiate signal will be ignored.

11.1.2 From Programs

Following is a list of program instructions that can be used to start user programs.

Call Program

The Call Program.# instruction acts like a subroutine call by suspending the program that is active, and starting the program that is called. When the called program completes, program flow will return to the next line of the program that it was called from.

The program being called must be assigned to the same task as the program that is calling it.

Program.#.Initiate

The Program.#.Initiate instruction in a program allows the user to start another program that is assigned to a different task. If a program is already running on the task as the program being initiated, then the Program.#.Initiate instruction will be ignored.

If the program to be initiated is not assigned to a different task, then the instruction will have a Red Dot Error in the program editor.

11.1.3 From PowerTools Studio

Individual programs can be initiated from PowerTools Studio while online with the system. To initiate a program while online, click on the Run Program button on the Program Toolbar. Doing so will initiate the program number that is currently displayed.

11.2 Stopping Programs

11.2.1 From Assignments

There are numerous different methods that can be used to stop programs using Assignments in the PTi210 module. Listed below are the various assignment destinations.

Stop

The Stop Destination will, when activated, stop all motion and all programs that are active. Stop is level sensitive, so that while it is active, all motion and programs are prevented from starting. Stop will override any other functions that are active.

Program.#.Stop

The Program.#.Stop Destination is used to stop a specific user program without stopping all of the programs. Program.#.Stop is edge sensitive, so the stopped program can be initiated again immediately after it has been stopped.

11.2.2 From Programs

Following are the instructions used to stop user programs from within a user program.

Stop

The Stop command is used to stop all motion and programs that are currently active. Stop is level sensitive, so it must be activated by being set equal to ON (Stop = ON). Since programs cannot be initiated while Stop is active, the user must create an assignment to deactivate the Stop function if it is activated from a user program.

Program.#.ProgramStop

The Program.#.ProgramStop command is used to stop a specific user program. If the intended program is not active, the instruction will be ignored.

Program.#.ProgramStop is edge sensitive, therefore the program that is stopped can immediately be initiated again.

11.2.3 From PowerTools Studio

Programs can be stopped while online with PowerTools Studio using the following methods.

Stop All - Main Toolbar

The ***Stop All*** button on the PowerTools Studio toolbar can be used stop all motion and programs that are active. The ***Stop All*** can be toggled on and off from the toolbar. Click the button once to toggle the Stop on. Motion and programs will be prevented from running until the button is clicked again.

Stop All - Program Toolbar

Motion and Programs can also be stopped from the Program Toolbar using the ***Stop All*** button. This button is like an edge sensitive function that will not prevent motion or programs from being initiated again.

Pause/Break Key

When online it is possible to stop any motion program from the keyboard on your PC by pressing the Pause/Break key.
Pressing Pause/Break is identical to clicking on the Stop All button on the PowerTools Studio toolbar. Once motion has been stopped using the Pause/Break key on the keyboard, it is necessary to press the Pause/Break key again to allow motion.

12 Parameter Descriptions

This section lists all programmable and feedback parameters available. The parameters are listed alphabetically by variable name (shown in italics below the on screen name) and give a description. Range is dynamic and depends on User Unit scaling. The units of the parameters are dynamic and depend on selected User Units.

Absolute Home Defined
AbsHomeDefined

This parameter is used when the Absolute Position Auto-Calculate Enable check box is selected (active). If the check box is active, and the system is homed either with a Home motion profile or using the DefineHome function, the AbsHomeDefined parameter is set to TRUE and stays active. The AbsHomeDefined is an indicator that an absolute encoder is being used AND that the system has been homed. Therefore, there should be no need to home the machine again.

If this parameter is TRUE, the PTi210 module will calculate the position feedback of the machine based on the absolute position from the encoder, and the stored absolute home position. If FALSE, the PTi210 module will not account for any stored absolute home position when loading the position feedback on power-up or after warm-start.

This parameter is stored in a section of Non-Volatile Memory (NVM) that is not accessible to the user. Therefore, the value is not lost when a download procedure is performed. To manually reset this parameter, you need to use the UndefineHome function. See the Reasons for Re-Homing section to see other scenarios that could cause AbsHomeDefined to turn off.

Absolute Home Position Decimal Places
AbsHomePosnDecimalPlaces

This read-only parameter is used to automatically store the number of decimal places being used at the time the absolute home was defined. This value is used by the PTi210 module so that it can correctly calculate the machine position based on the absolute encoder position and the absolute home position on power-up. This number is stored is so that if the user changes the number of decimal places on the distance user units after the absolute home position is defined, the system does not need to be re-homed. This parameter is not overwritten on a download.

Absolute Home Position in User Units
AbsHomePosnUserUnits

This read-only parameter is used to store the position of the system when at the absolute home position. This parameter is written to automatically by the PTi210 module after a home is performed when using the Absolute Position Auto-Calculate Enable function. This value will come either from the End of Home Position if using a Home motion profile to home the machine, or the Define Home Position if using the DefineHome function to home the machine. This value is stored so that the correct machine position can be calculated on power-up based on the absolute encoder position and the absolute home position. This value is not overwritten on a download.

Absolute Home Rev Count
AbsHomeRevCount

This read-only parameter stores the whole revolution counter of the absolute encoder feedback when the absolute home position is defined. This value is then used as an offset to correctly calculate the machine position on power up. This value is not overwritten on a download.

Absolute Home Rev Position
AbsHomeRevPosn

This read-only parameter stores the non-integer portion of the absolute encoder feedback when the absolute home position is defined. Divide this parameter by 2^32 and add to the Absolute Home Rev Count to get the whole position feedback value of the absolute encoder when at the absolute home position. This value is then used as an offset to correctly calculate the machine position on power up. This value is not overwritten on a download.

Absolute Position Valid
AbsolutePosnValid

This source is activated when either the DefineHome destination is activated, or any home is successfully completed (sensor or marker found). This source is deactivated if the drive is rebooted, powered down, or a home is re-initiated.

Absolute Position Auto-Calculate Enable
AbsPosnAutoCalculateEnable

This parameter is used to enable or disable the automatic calculation of the position for the PTi210 module on power-up, based on a defined absolute home position.

If this parameter is FALSE (or clear), the PTi210 module will only read the position from the absolute feedback device on power-up, and then set the position feedback equal to that value. However, the PTi210 module will NOT take into account any previously defined home position. If clear, the Absolute Position Mode parameters will remain unavailable on the Absolute Position View.

If this parameter is TRUE, the PTi210 module will internally calculate the correct feedback position of the machine in user units based on a previously defined home position and the position feedback from the absolute feedback device. The previous sentence is valid only if the system has been previously homed (either using a homing routine or the DefineHome function). The Absolute Position Mode setting then determines the method used for calculating the position feedback on power-up or after a warm-start (see Absolute Position Mode).

Absolute Position Mode
AbsPosnMode

This parameter is used to define how the machine position is calculated on power-up, based on the position feedback from the absolute encoder, and the stored absolute home position. Options can be either One-Sided Mode, or Two-Sided Mode.

One-Sided indicates to the PTi210 module that the application will use the entire resolution of the multi-turn absolute encoder on the positive side of the absolute home position. This allows maximum travel in one direction without reaching the rollover point of the absolute feedback device. If the motor travels in the negative direction from the absolute home position, or more than the maximum resolution of the encoder in the positive direction in this scenario, the machine position likely will not be correctly calculated on the subsequent power-up.

Two-Sided indicates that the entire resolution of the multi-turn absolute encoder is to be evenly distributed on both sides of the absolute home position. This allows for travel in both directions from the absolute home position and still the PTi210 module can correctly calculate the machine position on power-up. If the motor travels more than half the maximum resolution of the absolute feedback device in one direction from the absolute home position, the machine position likely will not be correctly calculated on the subsequent power-up.

For examples on One-Sided or Two-Sided Modes, see the Absolute Position View section of this manual.

Accelerating
Accelerating

This source is active when the PTi210 module is executing an acceleration ramp. A normal index consists of 3 segments: Accelerating, At Velocity, and Decelerating. The Accelerating source will be set (active) during this acceleration segment regardless of whether the motor is speeding up or slowing down. Therefore, this source can sometimes be active when the motor is decelerating. This could be true when compounding indexes together.

Acceleration Type
AccelType

This parameter is used to select the accel/decel type for all motion (homes, jogs and indexes). The “S-Curve” ramps offer the smoothest motion, but lead to higher peak accel/decel rates. “Linear” ramps have the lowest peak accel/decel rates but they are the least smooth ramp type. “5/8 S-Curve” ramps and “1/4 S-Curve” ramps use smoothing at the beginning and end of the ramp but have constant (linear) accel rates in the middle of their profiles. The “5/8 S-Curve” is less smooth than the “S-Curve” but smoother than the “1/4 S-Curve”. S-Curve accelerations are very useful on machines where product slip is a problem. They are also useful when smooth machine operation is critical. Linear ramps are useful in applications where low peak torque is critical. Below is a comparison of the 4 ramp types:

- S-Curve: Peak Accel = 2 x Average Accel
- 5/8 S-Curve: Peak Accel = 1.4545 x Average Accel
- 1/4 S-Curve: Peak Accel = 1.142857 x Average Accel
- Linear: Peak Accel = Average Accel

Acceleration Decimal Places
AccelUnits.Decimal

This parameter is the decimal point location for all real-time accel./decel. ramps.

Invert Acceleration Units
AccelUnits.InvertAccel

When enabled, all accelerate/decelerate units will be inverted, or put another way, revs/min/ms becomes ms/rev/min. For example, acceleration in units of rev/min/ms would be displayed as ms/rev/min after it is inverted. This feature allows users to enter accelerate/decelerate values in the units that make the most sense to them. Some prefer to think in terms of in/s/s while others prefer ms/rev/min.

Units Name
AccelUnits.Name

This is a 6-character name for the acceleration user units you want to use in your application.

Acceleration Time Scale
AccelUnits.TimeScale

This parameter is the time units for accel./decel. ramps. Possible selections are milliseconds or seconds.

At Velocity
AtVel

This source is active when the PTi210 module is executing a constant velocity motion segment. One example would be during an index. The source would activate after the motor has finished accelerating up to speed and before the motor begins to decelerate to a stop. A normal index consists of 3 segments: Accelerating, At Velocity, and Decelerating. This source is active during the At Velocity segment, and is activated based on the commanded velocity, not the feedback velocity. During synchronized motion, AtVel can be active without actual motor movement.

Bit Number Value
Bit.B# or Bit.BitName

This read/write bit may be used in a program as an intermediary variable bit controlled by the user. Bit.B# is one of 32 bits that make up the BitRegister parameter. Assigned to communication networks such as DeviceNet, Profibus and Modbus, Bit.B# may be used to transfer events that have occurred in a PLC to the PTi210 program.

NOTE

When the value of Bit.B# is changed, the value of BitRegister.#.Value is changed as well.

Bit Register Number Value
BitRegister.#.Value

This parameter is made up of the combination of the 32 Bit.B#. The BitRegister.#.Value. The BitRegister.#.Value register may be accessed bitwise by using Bit.B#, or double word-wise by using BitRegister.#.Value.

Bit Register Number Value Mask
BitRegister.#.ValueMask

This parameter is the Mask for the BitRegister.#.Value. Each bit location is set to either transfer the current data in the corresponding bit location of BitRegister.#.Value (by setting the bit location to 1) or to clear the current data in BitRegister.#.Value (by setting the bit location to 0).

Braking Resistor Overload Accumulator
BrakeOverloadAccumulator

This parameter gives an indication of braking resistor temperature based on a simple thermal model. The formula for the thermal model is a function of parameters Pr **10.030** and Pr **10.031** (set in the initialization file). A value of zero indicates that the resistor is close to ambient, and a value of 100 % is the maximum temperature (trip level). A Brake Resistor warning is given if this parameter is above 75 % and the braking IGBT is active. This parameter can be found on the Online tabs of the Status and Current views.

Braking Resistor Alarm
BrakeResistorAlarm

This parameter is set when the braking IGBT is active and the braking overload accumulator is greater than 75 %. This parameter when activated is automatically kept active for 0.5 seconds so that it can be displayed on the keypad display. This parameter can be found on the Online tabs of the Status and Current views.

Bus Voltage
BusVoltage

This read-only parameter displays the instantaneous voltage value of the DC bus in the drive. This parameter is found on the Status Online tab on the Status view and is read directly from parameter Pr **05.005** in the database.

Active Point
Cam.ActivePoint

This parameter is the point within a cam table that is being executed. This is useful to determine the cam location when a fault occurs. This only available when online.

Active Table
Cam.ActiveTable

This parameter is active when the cam table instance number is being executed. This is useful to determine the cam location when a fault occurs. Only available when online.

Direction
Cam.CamDirection

When ComPlus is selected the cam only moves in the forward direction and only chains in the forward direction. If in ComPlus mode and the master axis moves backwards, the cam stops motion and will not continue its forward motion until the master has moved back into position where it started to move backwards.

When Bidirectional is selected the cam will follow the path backwards and will backwards chain to another cam table. This parameter is accessible on the Cam view. The user can change the Cam.CamDirection in a program, it will take effect in the next Cam.#.Initiate.

Realtime cams move only in the ComPlus direction. If you backup from a bidirectional sync cam into a realtime cam the cam will fault. If you have this configuration, we recommend you add a backup cam table for a sync table to handle transition jitter.

Captured Master Position Homed
Cam.CaptureMasterPosHomed

This parameter is the master commanded position captured when the current running cam table was initiated, value is referenced to home. If the cam is not running it is the “Master Position Homed ” when the cam was exited (by stop, suspend, or normal completion).

Captured Master Position
Cam.CapturedMasterPosition

The master commanded position captured when the current running cam table was initiated. If the cam is not running it is the “Master Position” when the cam was exited (by stop, suspend, or normal completion).

Captured Position Command
Cam.CapturedPositionCommand

The follower commanded position captured when the current running cam table was initiated. If the cam is not running it is the "Position Command" when the cam was exited (by stop, suspend, or normal completion).

Captured Position Feedback
Cam.CapturedPositionFeedback

The follower feedback position captured when the current running cam table was initiated. If the cam is not running it is the "Position Feedback" when the cam was exited (by stop, suspend, or normal completion).

Captured Time
Cam.CapturedTime

Time the current running cam table was initiated. If the cam is not running it is the time when the cam was exited (by stop, suspend, or normal completion).

Command Complete
Cam.CommandComplete

This pararmeter is active when any cam motion command is completed. If a stop is activated before the cam has completed, this parameter will not activate. This parameter is deactivated when any cam command is initiated.

Commanding Motion
Cam.CommandingMotion

This source is active (on) when any Cam table is executing. Note this source may be on even when no physical motor motion is occurring. For example the master may not be moving, but the follower is still considered to be in a Commanding Motion state.

Stop Decel Enable Check Box
Cam.DecelEnable

When the check box is selected (activated) the Cam.StopDecel parameter is used when Cam.Stop and Cam.Suspend commands are activated.

Resume

Cam.Resume

Resumes the cam execution from a Cam.Suspend command (or SetCamMasterOffset(MasterPosn), or SetCamFollowerOffset(FollowerPosn). The cam points are all relative to the start of the Cam table. On Cam.Resume the current physical position becomes the cam start point and the cam aligns it's resume position to the current physical position without any physical movement. This means if you suspend a cam and move the physical position; for example with a Jog, you will need to position the motor back to the desired position before resuming.

Cam.Resume will be ignored if executed without Cam.ResumeAvailable set.

Resume Available

Cam.ResumeAvailable

Indicates that the cam is in a state where it will accept the Cam.Resume command. The cam can be resumed after executing a Cam.Suspend, SetCamMasterOffset(masterPosn), or SetCamFollowerOffset(FollowerPosn). Cam.Resume will be ignored if executed without Cam.ResumeAvailable set.

This source is active when a cam Resume is available.

To alter the pattern a program is run concurrently with the cam motion and the Cam.#.ForwardChain and Cam.#.BackwardsChain are changed on the fly to alter the motion pattern. To get even more flexibility you can dynamically alter the cam data table itself with the cam[t,e]xxx commands. You can even alter the size of the table.

Accepts Motion Modifiers	Action
Cam[table,element].Master=	Writes to a cam table's master value
Cam[table,element].Follower=	Writes to a cam table's follower value
Cam[table,element].Interpolation=	Writes to a cam table's master value. There is no range checking of interpolation until cam execution. If the cam type does not support Interpolation, a parm not found fault is generated
= Cam[table,element].Master	Reads the cam table's master value
= Cam[table,element].Follower	Reads a cam table's follower value
= Cam[table,element].Interpolation	Reads a cam table's interpolation value

In the table above "table" and "element" are expressions.

The time based index is defined as a cam component. Time based index allows easy entry to dynamically define either the distance at a fixed time or dynamically define time at a fixed distance or dynamically alter both distance and time.

Resume Follower Position

Cam.ResumeFollowerPosn

This will be the initial Follower Position the cam will be set to on execution of the Cam.Resume command. This value is valid only after Cam.Suspend, SetCamMasterOffset(MasterPosn), or SetCamFollowerOffset(FollowerPosn) commands.

Resume Master Position

Cam.ResumeMasterPosn

For cam tables running in synchronized mode, this will be the initial Master Position the cam will be set to on execution of the Cam.Resume command. This value is valid only after Cam.Suspend, SetCamMasterOffset(MasterPosn), or SetCamFollowerOffset(FollowerPosn) commands.

Resume Master Time

Cam.ResumeMasterTime

For cam tables running in real time mode, This will be the initial Master Time the cam will be set to on execution of the Cam.Resume command. This value is valid only after Cam.Suspend, SetCamMasterOffset(MasterPosn), or SetCamFollowerOffset(FollowerPosn) commands.

Stop

Cam.Stop

When Cam.Stop is activated the cam will stop using the Cam.Decel ramp rate (if Cam.DecelEnable is enabled).

Stop Decel

Cam.StopDecel

This parameter is the deceleration rate of the cam after a Cam.Suspend or Cam.Stop command is initiated. A value of zero disables the stop decel ramp as well as clearing the Stop Decel Enable check box.

Suspend
Cam.Suspend
When Cam.Suspend is activated the current cam will stop using the Cam.Decel ramp rate (if Cam.DecelEnable is enabled). The cam will accept a Cam.Resume after a suspend. The suspend records the master and follower positions at the point of the suspend for future Cam.Resume execution.

Max Accel
Cam.#.Accel
This parameter is the maximum acceleration for the Time Base Index cam type. The time base index will not accelerate faster than this value.

Backward Chain
Cam.#.BackwardChain
This parameter holds the next cam table to initiate if this cam table completes in reverse (master axis is moving from larger to smaller position values). This is only available in bidirectional mode. The next cam table will be initiated at the end (which is really the starting point) of the current table. If no cam is to be initiated, this value should be set to –1 (minus one) to stop at the end of the cam table execution.

Cam Table Complete
Cam.#.CamTableComplete
This parameter is active when the specified cam table motion command is completed, if a stop is activated before the cam has completed this parameter will not be activated. Inactivated when the specified can command is executed.

Cam Table In Motion
Cam.#.CamTableInMotion
This parameter is active (on) when the specified cam table is executing. Note this source may be on even when no physical motor motion is occurring. For example the master may not be moving, but the follower is still considered to be in a Commanding Motion state.

Cam Table Size
Cam.#.CamTableSize
This parameter is the number of elements entered in the specified cam table.

Writable Check Box
Cam.#.CamTableWritable
Moves the cam table into ram memory so the user can change the cam table values using a program.

Cam Type
Cam.#.CamType
Cam types to choose from are; Master Follower, Absolute MFI, Incremental MFI, Cubic Spline, or Time Based Index. Most data entries are “Absolute” which means each point is an absolute distance from the start of the cam table which is an implied zero, although the starting value does not have to be zero. The Incremental MFI (Master/Follower/Interpolation) has the entries as distance deltas from point to point which means the point is relative to the previous point.

Max Decel
Cam.#.Decel
This parameter is the maximum deceleration for the Timed Index. The timed index will not decelerate faster than this value.

Distance
Cam.#.Dist
This parameter is the incremental distance the timed index will move.

Final Velocity
Cam.#.FinalVelocity
The cam table profile will exit at this velocity. See Initial Velocity for more information. When using a single cam table, with no chaining, simply set the Initial Velocity to zero.

Forward Chain
Cam.#.ForwardChain

This parameter holds the next cam table to initiate when this cam table is completed. If no cam table is to be initiated, this parameter should be set to -1 (negative one) to stop the cam at the end of the current table. The cam will then conclude execution.

Forward chain and Backward chain can be dynamically changed by a user program. For example, a program monitors the cam motion flow and alters the forward and backward chain variables to switch the flow. You can have a start up sequence of tables, a running sequence of tables, a shut down sequence of tables and a alternate operation sequence of tables. The monitoring program adjusts the chains to dynamically change the cam sequence.

The cam tables themselves can be altered on the fly as long as you are altering the non-executing cam table. Each cam point is accessible by the user program provided the Writable check box has been selected.

Index Time
Cam.#.IndexTime

The Timed Base Index will slow down the index velocity so the index will run for this parameter value. The acceleration and deceleration will be reduced to meet the distance and index time of this timed base index.

Initial Velocity
Cam.#.InitialVelocity

This is the entry point velocity for the cam table. For Master Follower this is a calculated value. The user should take care in matching the velocity transitions when chaining one cam profile into another.

For MFI and Spline cam tables: If the first segment interpolation type is Linear, the initial velocity is the calculated velocity of the first segment.

When using a single cam table, with no chaining, simply set the Initial Velocity to zero.

Initiate
Cam.#.Initiate

Cam.#.Initiate has two forms - Program Instruction and Assignment (as a Destination). Both are used to initiate a specific Cam. For the Destination, the Cam is initiated on the rising edge of this event. Using the Destination, a Cam cannot be initiated if there is an Index, Home, Jog, or Program in progress, or if the Cam.#.Initiate is an instruction. If any motion is active, the program will hold on this instruction until that motion is complete (unless it is run on a different profile).

Interpolation
Cam.#.Interpolation

This a Timed Index parameter. The Interpolation method to be used for the acceleration and deceleration portion of the timed Index. Interpolation types are Square, S-Curve and Jerk.

Name
Cam.#.Name

The user can specify a cam name of up to 12 alphanumeric characters. This allows assigning a descriptive name to each cam table indicating different machine operations.

Repeat
Cam.#.Repeat

This parameter specifies how many times in a row the cam table is to run before proceeding on to the next cam table as defined by Forward Chain or Backward Chain text boxes. To repeat a single cam table forever use the Forward Chain and Backward Chain and set the chain value to it own cam table number.

Table Limit
Cam.#.TableLimit

The cam table execution is limited to this value, so you can programmatically change the size of the cam table execution. On download and restart, this is initialized to match the Cam.#.CamTableSize. If you attempt to set Cam.#.TableLimit > Cam.#.CamTableSize it will be set to the maximum, Cam.#.CamTableSize.

Time Base
Cam.#.TimeBase

This list box selects the Time Base for the cam master position entries. Realtime and Synchronized (to the Master Encoder) are allowed selections.

Max Velocity
Cam.#.Vel
This parameter is the maximum velocity of the Timed Base Index.

Number
Capture.Number
This parameter defines the size of the Capture list.

Capture Activate
Capture.#.CaptureActivate
If the Capture component is enabled and has been reset (CaptureTriggered is inactive), then the rising edge of CaptureActivate will capture the four data parameters and cause CaptureTriggered to be activated. If the Capture component is not enabled, or has not been reset, the CaptureActivate will be ignored.

Capture Clear
Capture.#.CaptureClear
This command may be used within a program to rearm the capture for the next cycle. This command can be used instead of Capture.#.Reset destination which needs to be cleared after a control loop cycle to allow another reset.

Capture Enable
Capture.#.CaptureEnable
The CaptureEnable is used to enable or “arm” the capture component. If the CaptureEnable is not active, then the CaptureActivate has no effect, and the CaptureTriggered remains inactive. Once the CaptureEnable is activated, the Capture component is ready and waiting for a CaptureActivate signal to capture data. CaptureEnable is a read-only destination on the Assignments view, and is accessible through a user program.

Capture Reset
Capture.#.CaptureReset
The CaptureReset is used to reset or re-arm the capture component after it has been activated. If the capture has been activated, the CaptureTriggered destination will be active. The capture component cannot capture data again until it has been reset. The capture component will automatically reset itself if the CaptureEnable signal is removed.

Capture Triggered
Capture.#.CaptureTriggered
The CaptureTriggered signal is read-only and indicates that the Capture component was activated and that data has been captured. CaptureTriggered will activate on the leading edge of CaptureActivate if the Capture component is enabled and reset.
Capture Triggered will remain active until CaptureReset is activated.

Name
Capture.Name
You can assign a descriptive name to each capture, making the setup easier to follow. The length of the text string is limited by the column width with a maximum of 12 characters. Simply double click on the Name field of any capture’s line to assign a name to it.

Number
Capture.Number
This parameter defines the size of the Capture list.

Captured Master Position Homed
Capture.#.CapturedMasterPosHomed

On the rising edge of the Capture Activate event, the master axis position is captured (in counts). The captured counts value is then converted into Master Distance User Units and stored in the Captured Master Position parameter. If the user redefines the zero position of the master axis, the master position in User Units is zeroed out, however the counts parameter is not. Therefore, if another capture occurs after the position has been zeroed, the value stored in the Captured Master Position parameter will be off by the value of the master axis before the position was zeroed out. This parameter is to be used if the master axis position is redefined after power-up.

Captured Master Position
Capture.#.CapturedMasterPostion

The master axis feedback position, in master axis distance units, at the time when CaptureTriggered activated.

Captured Position Command
Capture.#.CapturedPositionCommand

The command position, in user units, at the time when CaptureTriggered activated.

Captured Position Feedback
Capture.#.CapturedPositionFeedback

The feedback position, in user units, at the time when CaptureTriggered activated.

Captured Time
Capture.#.CapturedTime

The time, in microseconds, from a free-running 32-bit binary counter at which CaptureTriggered activated.

NOTE

This parameter has a resolution of 1 µs when used with suitable high precision capture inputs, the value of which is taken from a 32-bit counter started when the PTi210 module has initialised, so can be used to indicate the time difference between captured positions.

Clear Following Error
ClearFollowingError

Clear Following Error is a destination found in the Position group on the Assignments view. When this destination is activated, any following error that has accumulated will be erased. Following Error is cleared by setting the commanded position to the feedback position, automatically resulting in a zero following error. The PTi210 module will deactivate the Clear Following Error destination as soon as Following Error is zero.

Commanding Motion
CommandingMotion

This source activates when VelCommand is non-zero.

Current Demand
CurrentDemand

The current demand is read from the drive parameter Pr **04.004**. The units for the Current Demand are % of rated current. 100 % rated current is defined as the rated current of the system (Motor Rated Current < System Rated Current < Drive Rated Current).

This read-only parameter can be found on the Online tabs of the Status and Current views.

Current Level
CurrentLevel

The PTi210 module constantly monitors the amount of current being output from the drive. The user can set the Current Level parameter such that when the Current Demand from the drive is equal to or greater than the Current Level, the Current Level Active event will activate. This is simply a flag that can be used to indicate to the user that a certain amount of current/torque is being generated by the drive.

Units for the Current Level are % Continuous. This means that 100 % is equal to the system rated current (Motor Rated Current < System Rated Current < Drive Rated Current). This parameter is found on the Current view.

Current Level Active
CurrentLevelActive

The Current Level Active event activates when the Current Demand meets or exceeds the Current Level parameter value. This event can be found on the Assignments view and can be assigned to a digital output or other event to signal to the user when the current level is exceeded. The event will deactivate when the Current Demand is less than the Current Level value.

Current Limit
CurrentLimit

The user can limit the amount of current that the drive can generate by configuring the Current Limit in the PTi210 module. The Current Limit Enable signal must be active for the current to be limited to the Current Limit value. By default. the Current Limit Enable signal is not active.

Units for the Current Limit are % Continuous. This means that 100 % is equal to the system rated current (Motor Rated Current < System Rated Current < Drive Rated Current). This parameter is found on the Current view.

Current Limit Active
CurrentLimitActive

The Current Limit Active event activates when the Current Demand meets or exceeds the Current Limit parameter value. This event can be found on the Assignments view and can be assigned to a digital output or other event to signal to the user when the current is being limited by the PTi210 module. The event will deactivate when the Current Demand is less than the Current Limit value.

Current Limit Enable
CurrentLimitEnable

The user can limit the amount of current that the drive can generate by configuring the Current Limit in the PTi210 module. The Current Limit Enable signal must be active for the current to be limited to the Current Limit value. The Current Limit Enable signal is not active by default.

Decelerating
Decelerating

This source is active when the PTi210 module is decelerating. A normal index consists of 3 segments: Accelerating, At Velocity, and Decelerating. Decelerating follows the accelerating segment and the At Velocity segment. When indexes are compounded to create a complex motion profile, only the last index may contain a decelerating segment.

Define Home
DefineHome

This destination is used to set the Commanded Position to the value specified in the DefineHomePosn variable. On the rising edge of this input function the absolute position is set equal to the DefineHomePosn and the AbsolutePosnValid output function (source) is activated.

The UndefineHome parameter must be active before the DefineHome destination can be activated and the DefineHomePosn can be changed.

DefineHomeNow
DefineHomeNow

This destination is used to reset the position command parameter PosnCommand. On a rising edge of this input function the position command parameter will be set to 0 at the current shaft position without the need to use the UndefineHome destination first, also the AbsHomeDefined parameter is activated.

UndefineHome
UndefineHome

This destination is used to clear the AbsHomeDefined parameter in order to reset the Home Position. On a rising edge of this input function the AbsHomeDefined parameter is deactivated. The DefineHome destination can then be activated to set the Home position to the current value in the DefineHomePosn parameter.

Define Home Position
DefineHomePosn

The DefineHome parameter is used to set the motors absolute position to the value stored in the DefineHomePosn variable. On the rising edge of the DefineHome function the Commanded Position is set equal to the DefineHomePosn and the AbsolutePosnValid source is activated.

Acceleration
DistanceRecovery.Accel
This parameter in user units is the acceleration rate for the distance recovery index.
DistanceRecovery.Decel

Deceleration
DistanceRecovery.Decel
This parameter in user units is the deceleration rate for the distance recovery index.

Enable Distance Recovery
DistanceRecovery.DistanceRecoveryEnable
Select the checkbox to enable the additive distance recovery index feature.

Velocity
DistanceRecovery.Vel
This parameter, in user units, is the velocity limit of the distance recovery index.

Characteristic Distance
DistUnits.CharacteristicDistance
This parameter is the distance the load travels (in user units) when the motor travels the characteristic length (in motor revolutions). This parameter is used along with the DistUnits.CharacteristicLength to establish the relationship between user distance and actual motor travel distance. See *User Units View* on page 84.

Characteristic Length
DistUnits.CharacteristicLength
This parameter is the distance the motor travels (in whole number of revolutions) to achieve one characteristic distance of load travel. This parameter is used along with the DistUnits.CharacteristicDist to establish the relationship between user distance and motor travel distance. See *User Units View* on page 84.

Distance Decimal Places
DistUnits.Decimal
This parameter is used to select the number of decimal places used in the DistUnits.CharacteristicDist. Using a high number of decimal places will improve positioning resolution, but will also limit the maximum travel distance. The number of decimal places set in this parameter determines the number of decimal places used in all distance parameters throughout the software. You can select from zero to six decimal places of accuracy.

Distance Units Name
DistUnits.Name
This is a text variable which is used as the label for the distance/position user units. It can be up to 12 characters in length.

Drive Active
DriveActive
The DriveActive event, when active, indicates that the bridge on the drive is closed (drive is enabled). This parameter is read-only, and is read directly from parameter Pr **10.002** in the database. If/When a drive trip occurs, this parameter will deactivate automatically.

Drive Enable Status
DriveEnableStatus
This source is active when the drive is enabled and healthy.

Drive Healthy
DriveHealthy
The Drive Healthy signal indicates that the drive is not in the trip state as indicated from the drive parameter Pr **10.001** (*Drive Active*). This bit is active when no trip is active. On activation of a drive trip, this signal will deactivate automatically. When the trip is cleared, Drive Healthy will activate again.

Drive Serial Number
DriveSerialNumber

This displays the serial number of the Drive to which the PTi210 module is attached.

Drive OK
Error.DriveOK

Active when there are no errors. Inactivated when any error or trip, except travel limits occur. Drive enable has no effect on this event.

PTi210 Errors Bitmap
Error.ModuleMotionErrorBitmap

This parameter is a decimal value that equates to any errors, see table below.

Bit #	Decimal Value	Error Name
0	1	Watchdog Timer Error
1	2	Invalid Configuration Error
2	4	NVM Invalid Error
3	8	Power Up Test Error
4	16	Module Following Error
5	32	Travel Limit Plus Active
6	64	Travel Limit Minus Active
7	128	User Program Error
8	256	No Program Loaded Error
9	512	Not Used
10	1024	Not Used
11	2048	Not Used
12	4096	Trajectory Error
13	8192	Parameter Access Error
14	16384	Not Used
15	32768	Module Overtemperature Error
16	65536	Trajectory Update Rate Overrun Error
17	131072	Digital Output Shorted Error
18	262144	Not Used
19	524288	Not Used
20	1048576	Not Used
21	2097152	Not Used
22	4194304	Not Used
23	8388608	Not Used
24	16777216	Not Used
25	33554432	Not Used
26	67108864	Not Used
27	134217728	Not Used
28	268435456	Not Used
29	536870912	Not Used
30	1073741824	Not Used
31	2147483648	Not Used

Reset Errors
Error.Reset

Resets errors that do not require a power down. This event is "or"ed with the reset button on the drive.

PTi210 Module I/O Status Word
PTi210Connect.DigitalIOReadWord

This parameter is a bitmap that contains the status of the digital I/O on the PTi210 module. This parameter is read-only and is used to control the virtual LEDs on the PTi210 module I/O Setup view.

ModuleInput Debounce Time
ModuleInput.#.DebounceTime

The digital inputs on the PTi210 module can be debounced to reject electrical noise on a wire or switch bounce. When an input activates, it must be active for at least the specified Debounce Time before the input is considered active in the PTi210 module firmware. Once the input has been active for the debounce time, a rising edge on the input will occur in the PTi210 module. This parameter can be found on the PTi210 module I/O Setup view.

ModuleInput Force
ModuleInput.#.Force

The digital inputs on the PTi210 module can be forced On or Off within a user program. To do so, the Force Enable event for the desired input must first be activated. Once the Force Enable for the input is active, the input can be forced on or off using the following instructions:

```
ModuleInput.#.Force = ON
ModuleInput.#.Force = OFF
```

Forcing an input On or Off does not prevent it from changing state again. It is solely a means of changing the state from within a user program.

ModuleInput Force Enable
ModuleInput.#.ForceEnable

ModuleInput.#.Force must be active to force a specified input on or off in a user program. See ModuleInput Force above for further information.

ModuleInput Status
ModuleInput.#.In

ModuleInput.#.In is the status of the specified digital input on the PTi210 module. The ".In" at the end of the parameter is optional, so this signal is most often referred to simply as ModuleInput.#. This signal can be used in a user program or on the Assignments view. These inputs are updated at the Trajectory Update Rate specified on the Setup view.

These inputs are also used for the high-speed capture functionality. The PTi210 module is capable of capturing the motor position, master encoder position, and time within 1 µsec of the rising edge of the ModuleInputs.

ModuleOutput Name
ModuleOutput.#.Name

Each digital output on the PTi210 module can be given a name. The name can be used in a user program to reference a specific input. The name can be up to 12 alphanumeric characters, but must begin with a non-numeric character.

ModuleOutput State
ModuleOutput.#.Out

ModuleOutput.#.Out is used to change the state of a specific digital output on the PTi210 module. The ".Out" at the end of the parameter name is optional, so this signal is often referred to simply as ModuleOutput.#. In a user program, ModuleOutput.#.Out is set On or Off to activate or deactivate the digital outputs. From the Assignments view, any Source can be assigned to the ModuleOutput to activate them automatically.

Drive OK
Fault.DriveOK

Active when there are no faults. Inactivated when any fault except travel limits occur. Drive enable has no effect on this event.

Enable Feedforwards
FeedforwardsEnable

This parameter may be setup on the Tuning view or through a program, and enables feedforward compensation. When feedforwards are enabled, the accuracy of the Inertia and Friction settings are very important. If the Inertia setting is larger than the actual inertia, the result could be a significant overshoot during ramping. If the Inertia setting is smaller than the actual inertia, following error during ramping will be reduced but not eliminated. If the Friction is greater than the actual friction, it may result in velocity error or instability. If the Friction setting is less than the actual friction, velocity error will be reduced, but not eliminated.

Feedhold
Feedhold
When this destination is activated the motor will decelerate to a stop in the time specified by the FeedholdDecelTime parameter. When it is deactivated the motor will accelerate back up to the programmed speed in the same amount of time. It is used to hold motion without cancelling the move in progress. If a feedhold is activated during an index the motor will come to a halt, but the index's velocity command remains at the velocity it was at before the feedhold was activated. When the feedhold is deactivated time will ramp back up and the index will continue on to its programmed distance or position. Feedhold affects indexes, homes, and programs. A jog is not affected by the feedhold unless it is initiated from a program. This is level sensitive.

Feedhold Deceleration Time
FeedholdDecelTime
When Feedhold destination is activated the motor will decelerate to a stop in the time specified by the FeedholdDecelTime parameter. While the feedhold destination is active, the motion profile is stopped.

FeedRate Deactivate
FeedRateDeactivate
This destination allows the user to deactivate the FeedRate Override feature. When FeedRate Deactivate is enabled, FeedRate Override will be disabled and all index or home motion will operate at its programmed velocity. When FeedRate Deactivate is disabled, FeedRate Override will be enabled, and index and home motion is subject to scaling by the FeedRate Override parameter. The default value for FeedRate Override is 100 %, so even when FeedRate Override is enabled, default motion will run at programmed velocity.

FeedRate Override
FeedRateOverride
This parameter is used to scale all motion. It can be described as “scaling in real time.” The default setting of 100 % will allow all motion to occur in real time. A setting of 50 % will scale time so that all motion runs half as fast as it runs in real time. A setting of 200 % will scale time so that all motion runs twice as fast as it would in real time. FeedRate Override is always active, and this parameter may be modified via Modbus or in a program. When changed, the new value takes effect immediately.

Following Error
FollowingError
Following Error displays the difference between the Position Command and the Position Feedback.

Enable Following Error
FollowingErrorEnable
This parameter can be setup from the Position view or from a program. When enabled, a following error fault will be generated if the absolute value of the Following Error exceeds the Following Error Limit.

Following Error Limit
FollowingErrorLimit
This parameter is used when the FollowingErrorEnable bit is set. This limit is compared to the absolute value of the FollowingError. If the FollowingError is greater than the FollowingErrorLimit, a following error fault will be generated.

Module Free Running Timer Time
FreeRunTime
The PTi210 module has a free-running timer with 1 microsecond accuracy that is always running. The user has access to this timer value in a user program using the FreeRunTime parameter. This value can be used to wait for a period of time, find the difference in time between two point in a program, or any other application. The timer is not resettable by the user. Since the timer is a signed 32-bit parameter with microsecond accuracy, its range is between 0.000000 seconds and 2147.483648 seconds.

Gear Accel
Gear.Accel
This parameter sets the acceleration of the realtime gearing ramp. Gear.Accel units are in Follower Units/Velocity Time Base/ Acceleration Time Base. The Gear.Accel functions only when the follower is ramping its speed up to meet the Masters at the specified Gear.Ratio.

Gear Accel Enable
Gear.AccelEnable

Gear.AccelEnable is a Destination that when it is "on" allows a gear to run a specified accel ramp after the gearing command is turned on.

Gear Accelerating
Gear.Accelerating

If Gear.AccelEnable is activated, this source is activated during the time between Gear.Initate = On and Gear.AtVel = On.

Gear Activate
Gear.Activate

The Gear.Activate destination is used to start gearing from an assignment. It is a level-sensitive function, which means that as long as Gear.Activate is active, gearing will be in progress. When deactivated, gearing motion will come to a stop without deceleration. This function is only available through the assignments screen. When gearing from a program, the Gear.Initiate and Gear.Stop instructions are used to start and stop gearing

The gearing function must be stopped by the same method that started it. If it is started from an assignment, then it must be stopped from an assignment or if started in a user program, it must be stopped in a user program.

Gear at Velocity
Gear.AtVel

The Gear.AtVel source indicates that the motor is running at the programmed gear ratio. In early releases of Gearing, acceleration and deceleration will not be used with gearing, so this source will always be active when gearing is active.

Gear Command Complete
Gear.CommandComplete

This source will activate when gearing has been stopped, and will remain active until the gear is initiated again. Gearing does not use a deceleration ramp, so this source will activate immediately after a Gear.Activate destination is deactivated.

Gear Command In Progress
Gear.CommandInProgrees

This source will activate when Gearing is initiated either from a program or through an assignment. The source will remain active as long as gearing is in operation. This source can be active even if the motor is not in motion as long as gearing is active.

Gear Decel
Gear.Decel

This parameter sets the deceleration of the realtime gearing ramp. Gear.Decel units are in Follower Units/Velocity Time Base/ Acceleration Time Base. The Gear.Decel functions only when the follower is ramping its speed down after the gearing function has turned off.

Gear Decel Enable
Gear.DecelEnable

Gear.DecelEnable is a Destination that when it is "on" allows a gear to run a specified decel ramp after the gearing command is turned off.

Gear Decelerating
Gear.Decelerating

If Gear.DecelEnable is activated, this source is activated during the time between Gear.Initate = Off and Gear.CommandComplete = On.

Direction
Gear.Direction

Allows the user to select which master axis command the follower is going to follow and the choices are Bidirectional, ComMinus and ComPlus.

Bidirectional

The follower will follow both the plus and minus master axis command at the specified ratio

ComMinus

The follower will follow only the minus master axis command

ComPlus

The follower will follow only the plus master axis command

Gear Initiate
Gear.Initiate

Gearing can be activated through an assignment, or from a program instruction, Gear.Initiate. If initiated from an assignment, the Gear.Activate destination is a level-sensitive event. This means that gearing will be active as long as the source to which it is assigned is active. If gearing from a user program, the Gear.Stop instruction is used to stop the gearing motion. The gearing function must be stopped by the same method that started it. If it is started from an assignment, then it must be stopped from an assignment or if started in a user program, it must be stopped in a user program.

Gear Ratio
Gear.Ratio

The ratio is defined as the number of follower distance units to move the motor per distance unit of travel. The gear ratio can be positive or negative and is a signed 32-bit parameter. The resolution of the parameter is determined by the number of decimal places configured for the Master Velocity Units on the Master Setup screen.

Gear Recovery Distance
Gear.RecoveryDist

This variable measures the distance the follower loses from the master. This distance lost is measured between a Gear Initiate and the Gear At Velocity.

Scale
Gear.ScaleDenominator

The denominator (bottom value of the scaling fraction) is the # of master distance units. Scaling is defined as the number of follower distance units that will be traveled per the number of master distance units of travel. The follower distance units is defined in the top value (gear.scalenumerator) of the fraction.

Scale
Gear.ScaleNumerator

The numerator (top value of the scaling fraction) is the # of follower units. Scaling is defined as the number of follower distance units that will be traveled per the number of master distance units of travel. The master distance units is defined in the bottom value (gear.scaledenominator) of the fraction.

Stop
Gear.Stop

Gear.Stop will stop gearing motion that has been initiated from a program using Gear.Initate.

Use Scale Check Box
Gear.UseScaleEnable

Select this check box to enable the scaling and disable the gear ratio. Scaling allows the user to use irrational ratios such as 1.0/7.0 that which is not possible with the single parameter, Gear.Ratio.

Graph.TriggerState
TriggerState

This parameter is used to control the Graph function from within a program.

0 = Filling Buffer: The whole buffer must be filled with selected data before trigger sampling starts

1 = Filled - Waiting Trigger: Continue putting data into the buffer, check back into previous sampled data for the trigger condition

2 = Manually Triggered: User pressed the stop button, the buffer will have what ever data is available

3 = Filled and Triggered: Stop loading buffer, trigger is found and offset data after the trigger is in the buffer ready for upload

Graph.TriggerState2
Graph State

This parameter is read-only and can be used in a program to show the state the graph is in.

0 = Filling Buffer: The whole buffer must be filled with selected data before trigger sampling starts

1 = Filled - Waiting Trigger: Continue putting data into the buffer, check back into previous sampled data for the trigger condition

2 = Manually Triggered: User pressed the stop button, the buffer will have what ever data is available

3 = Filled and Triggered: Stop loading buffer, trigger is found and offset data after the trigger is in the buffer ready for upload

Graph Run
GraphRun

This instruction allows the user to enable the graph from a user program. It is similar to pressing the Run button on the Graph view. This instruction will wait until the graph buffer is full, then it exits into the looking for trigger state.

SlotX Module Type
Hardware.SlotXModuleType

This parameter defines the type of module fitted in the specified slot number. This parameter can be found on each of the SlotX views. This parameter is read-only in a user program.

If the module type specified in the PowerTools Studio configuration does not match the actual module type fitted, the PTi210 module will generate an error, and the drive will trip.

Home Any Command Complete
Home.AnyCommandComplete

This source is active when any home motion command is completed, if a stop is activated before the home has completed the function will not be activated. Inactivated when a home command is executed.

Home Acceleration
Home.#.Accel

This parameter sets the average Acceleration rate used during the home, units are specified on the User Units page.

Home Accelerating
Home.#.Accelerating

Active during any acceleration while the specified home is in progress. Accelerating may turn off and on again based on the type of Home selected. Accelerating will activate during the Home back off sensor motion.

Home At Velocity
Home.#.AtVel

This source is activated when the home velocity is reached when a the specified home is in progress. It will activate and deactivate base on the home. Home At Velocity will not be activated during back off sensor portion of the home.

Home Calculated Offset
Home.#.CalculatedOffset

The Calculated offset is the distance travelled during the deceleration ramp from the home velocity to a stop. Calculated by PowerTools Studio.

Home Command Complete
Home.#.CommandComplete

This source is active when the specified home command is completed, if a stop is activated before the home has completed the function or if the Home Limit Distance has been exceeded it will not be activated. Inactive when a home command is executed.

Home Command In Progress
Home.#.CommandInProgress

This source is activated when the Home is initiated and remains active until all motion related to the Home has completed.

Home Deceleration
Home.#.Decel

The Deceleration ramp parameter is used during all the home moves specified in user units.

Home Decelerating
Home.#.Decelerating
This source is active during any deceleration while the specified home is in progress. Decelerating will turn off and on based on the type of Home selected. Decelerating will activate during the Home back off sensor motion.

End of Home Position
Home.#.EndPosn
This parameter defines the drive position at the completion of a home. Typically used to define the machine coordinate home position.

Home Initiate
Home.#.Initiate
When activated, this destination will initiate the specified home. Home will not initiate if an index, jog, program, or stop is currently active.

Home Limit Distance
Home.#.LimitDist
This parameter places an upper limit on the incremental distance traveled during a home. If no home reference is found the motor will decelerate to a stop at the limit distance and activate the Home.#.LimitDistHit event.

Enable Limit Distance
Home.#.LimitDistEnable
This parameter enables the specified Home.#.LimitDist. If not enabled, the home will run indefinitely until the home reference is found.

Home Limit Distance Hit
Home.#.LimitDistHit
This source is activated when the home sensor is not found before the Home Limit Distance is traveled.

Home Name
Home.#.Name
User name for the specified home.

Home Offset Type
Home.#.OffsetType
Selects calculated or specified home offset. Calculated offset is the distance traveled during the deceleration ramp from the home velocity. The specified offset allows the user to choose an exact offset from the Home Reference.

If On Sensor
Home.#.OnSensorAction
If the home sensor input is active when the home is initiated, this parameter determines the direction of motion. Two selections are possible. If "Back off before homing" is selected, the motor will turn in the opposite direction of the home until the home sensor is clear and then begin the home. If "Go forward to next sensor" is selected, the motor will turn in the commanded direction until the next rising edge of the sensor is seen. If using Modbus to view or modify this parameter, 1= Back off before homing, 0 = Go forward to next sensor.

Home Reference
Home.#.Reference
This parameter determines how the reference position is determined. The parameter can have one of three different values: 'Sensor', 'Marker', 'Sensor then Marker'. When the home reference is 'Sensor' the rising edge of the 'Home Sensor' input function is used to establish the reference position. When the home reference is 'Marker' the rising edge of the motor encoder's marker channel is used to establish the reference position. When the home reference is 'Sensor then Marker' the reference position is established using the first marker rising edge after the Home Sensor input function goes active.

Home Sensor Trigger
Home.#.SensorTrigger
This destination is usually a sensor input used as a reference for the home. This event is only used if the home is defined by sensor or by sensor and marker.

Home Specified Offset
Home.#.SpecifiedOffset

The specified offset parameter allows the user to specify an exact offset relative to the Home Reference. The commanded motion will stop at exactly the offset distance away from the sensor or the marker as specified.

Home Time Base
Home.#.TimeBase

The time base selects either realtime, which allows velocities, acceleration and deceleration to be based on real time, or synchronized, which allows for an external synchronization signal.

Home Velocity
Home.#.Vel

This parameter sets the target velocity for all of moves in the home. The sign determines the home direction. Positive numbers cause motion in the positive direction and negative numbers cause motion in the negative direction in search of the home sensor.

Index Any Command Complete
Index.AnyCommandComplete

This source is active when any index motion command is completed. If a stop is activated before the index has completed, this destination will not activate. Deactivated when any index command is initiated.

Index Profile Limited
Index.ProfileLimited

For timed indexes, if the values for Max. Velocity, Max. Acceleration and Max. Deceleration are such that the distance cannot be covered in the specified time, the Index.ProfileLimited flag will activate when the index is initiated. The Index.ProfileLimited flag will remain active until cleared using the Index.ResetProfileLimited assignment or program instruction. In this situation, the index will still operate, but the time will be extended. In other words, the profile will be performed using the maximum values and still cover the specified distance, but not in the specified time.

Index Reset Profile Limited
Index.ResetProfileLimited

If a timed index was not completed in the specified time, the Index.ProfileLimited source will activate. Index.ResetProfileLimited is used to clear the ProfileLimited flag and acknowledge that the index was not completed in the specified time. This can be activated through an assignment, or through a user program. This function is edge-sensitive, so holding Reset Profile Limited active will not prevent ProfileLimited from activating.

Index Acceleration
Index.#.Accel

This parameter is the average Acceleration rate used during the index. Units are specified on the User Units view in the PowerTools Studio software.

Index Accelerating
Index.#.Accelerating

This source is active while an index is accelerating to its target velocity. Once the index reaches the target velocity, or begins to decelerate, the Index.#.Accelerating source will deactivate.

Analog/Sensor
Index.#.AnalogLimitType

If the Analog option is selected, one of the analog signals must be selected in the analog list box. Available selections are Analog In Ch1, Analog Ch2, or Current Feedback. Then a comparison operator must be selected from the operator list box. Available selections are > (greater than) and < (less than). Last, a value must be entered for comparison.

Satisfying the comparison triggers the index registration event.

Index At Velocity
Index.#.AtVel

This source activates when the target index velocity is reached. If Feedrate override is changed or FeedHold is activated AtVelocity shall remain active. Index.#.AtVel will deactivate at the start of any deceleration or acceleration. During a synchronized index, this source could be active even without any motor motion if the master axis stops.

Index Blended Initiate
Index.#.BlendedInitiate(#)
The Index.#.BlendedInitiate into (#) destination is used to initiate a blended index. The index will run at velocity before using its deceleration ramp to accelerate or decelerate towards the velocity of the next index specified. The next index that is to be "blended into" from the first is on the command line is in parenthesis. The value within the parenthesis can also be a variable.

Index Command Complete
Index.#.CommandComplete
The Index.#.CommandComplete source will activate when the specific index completes its deceleration ramp. It will remain active until the specific index is initiated again. If the Stop destination is used during an Index, then the Index.#.CommandComplete will not activate.

Index Command In Progress
Index.#.CommandInProgress
The Index.#.CommandInProgress source is active throughout an entire index profile. The source activates at the beginning of the index acceleration ramp, and deactivates at the end of the index deceleration ramp.

Index Compound Initiate
Index.#.CompoundInitiate
When activated will initiate the specified Index to compound into the next index in the program. Only allowed in a program.

Index Deceleration
Index.#.Decel
This parameter is the Average Deceleration rate used during the index. Units are specified on the User Units page.

Index Decelerating
Index.#.Decelerating
This source is active while an index is decelerating from its target velocity. Once the index reaches zero velocity, or its next target velocity, the Index.#.Decelerating source will deactivate.

Index Distance
Index.#.Dist
This parameter is the Incremental distance that the index will travel or the absolute position for the specified index in user units. If an index type of Registration is selected, then this is a limit distance, or the maximum distance the index will travel if a registration sensor is not seen.

Index Time
Index.#.IndexTime
This parameter is used in conjunction with the Index.TimeIndexEnable parameter. If TimeIndexEnable is activated, then this is the time in which an index should complete its programmed distance. The units for this parameter depend on the current setting of the TimeBase parameter for the specific index. If TimeBase is set to "Realtime" (default), then the units are Seconds. The user can program the index time with resolution on 0.001 Seconds (or milliseconds). If TimeBase is set to "Synchronized", the units defined by the Master Distance Units found on the Master Setup screen.

Index Initiate
Index.#.Initiate
The Index.#.Initiate destination is used to initiate the specific index. The Index is initiated on the rising edge of this function. An Index cannot be initiated if there is an Home, Jog, or Program in progress, or if the Stop destination or if a travel limit is active. It can be activated from an assignment or from a program.

Index Limit Distance Hit
Index.#.LimitDistHit
This source is activated when the registration sensor is not found before the Limit Distance is traveled. If the Registration Window is enabled the sensor must be activated inside the window to be recognized.

Index Name
Index.#.Name
The user can specify an Index name of up to 12 characters.

Enable Index PLS
Index.#.PLSEnable

When Activated, this parameter enables the PLS (programmable limit switch) function for the specified index. It can be controlled from index view check box or from a program.

Index PLS Off Point
Index.#.PLSOffDist

This an incremental distance from the start of the index to the Index PLS off point. This is an unsigned value and is relative only to starting position of this index. Index direction does not affect this parameter. Index.#.PLSStatus will be active if the distance traveled from the start of the index is greater than the Index.#.PLSOnDist and less than the Index.#.PLSOffDist.

Index PLS On Point
Index.#.PLSOnDist

This an incremental distance from the start of the index to the Index PLS On Point. This is an unsigned value and is relative only to starting position of this index. Index direction does not affect this parameter. Index.#.PLSStatus will be active if the distance traveled from the start of the index is greater than the Index.#.PLSOnDist and less than the Index.#.PLSOffDist.

Index PLS Status
Index.#.PLSStatus

Controlled by the PLSOn and PLSOff Points, this is relative to the distance commanded since the start of the index. Index.#.PLSStatus will be active if the distance traveled from the start of the index is greater than the Index.#.PLSOnDist and less than the Index.#.PLSOffDist.

Index Registration Offset
Index.#.RegistrationOffset

This parameter is the Distance the motor will travel after a valid registration sensor or analog limit value has been detected.

Index Registration Type
Index.#.RegistrationType

This selects either sensor or analog as the registration mark for a registration index.

Index Enable Registration Window
Index.#.RegistrationWindowEnable

This check box enables (if checked) the Registration Sensor valid Window. When active, only registration marks that occur inside the registration window are seen as valid.

Index Window End
Index.#.RegistrationWindowEnd

This parameter defines the end of the Registration Sensor Valid Window relative to start position of this index. This is an unsigned value and is relative only to starting position of this index. Index direction does not affect this parameter. The Registration Window start position is greater than or equal to the Registration point and less than the Registration Window End position. If a registration sensor is seen outside of this window (not between the WindowStart and WindowEnd positions) then it will be ignored.

Index Window Start
Index.#.RegistrationWindowStart

This parameter defines the start of the Registration Sensor Valid Window relative to start position of this index. This is an unsigned value and is relative only to starting position of this index. Index direction does not affect this parameter. The Registration Window start position is greater than or equal to the Registration point and less than the Registration Window End position. If a registration sensor is seen outside of this window (not between the WindowStart and WindowEnd positions) then it will be ignored.

Index Registration Sensor
Index.#.SensorTrigger

If registration to Sensor is selected, when this destination activates, motor position is captured and is used as the registration point for registration type indexes.

Index Time Base
Index.#.TimeBase
The time base selects either realtime, which allows velocities, acceleration and deceleration to be based on real time, or synchronized, which allows for an external synchronization signal.

Index Timed Index Enable
Index.#.TimedIndexEnable
This parameter is used in conjunction with the Index.#.IndexTime parameter. If Index.#.TimedIndexEnable is active, then the programmed Velocity, Acceleration, and Deceleration will be used as maximum values, and the Index Time parameter will determine how long it takes to perform an index.

Index Velocity
Index.#.Vel
This parameter sets the target velocity of the specific index. The units for this parameter are specified in the User Units Setup view. When an index is initiated, it will ramp up to this velocity at the specified acceleration rate and run at speed until it decelerates to a stop (assuming the index is not compounded).

Inertia Ratio
InertiaRatio
This specifies the load to motor inertia ratio. For example, a value of 25.0 specifies that the load inertia is 25 times the inertia of the motor.

Initially Active
InitiallyActive
This source, when assigned to a destination, will activate the destination on power-up or upon the PTi210 module being reset. InitiallyActive can be assigned to any destination that does not create motion (i.e. indexes, jogs, homes, programs).

In Position
InPosn
This source activates when commanded velocity is zero and the absolute value of the following error is less than the InPosnWindow for at least the amount of time specified in the InPosnTime parameter.

In Position Time
InPosnTime
This parameter is the minimum amount of time that commanded motion must be complete and the absolute value of the following error is less than the InPosnWindow parameter for the InPosn source to activate.

In Position Window
InPosnWindow
The absolute value of the following error must be less than this value at the completion of a move for the InPosnTime before InPosn will activate.

Jog Any Command Complete
Jog.AnyCommandComplete
The Jog.AnyCommandComplete bit will activate when either Jog 0 or Jog 1 completes its deceleration ramp and reaches zero commanded speed. It deactivates when another jog is initiated.

Jog Minus Activate
Jog.MinusActivate
This destination is used to initiate jogging motion in the negative direction using the jog parameters of the jog selected by the Jog select input function. Jogging will continue as long as the destination is active. The motor will decelerate to a stop when the destination is deactivated. This is level sensitive.

Jog Plus Activate
Jog.PlusActivate
This destination is used to initiate jogging motion in the positive direction using the jog parameters of the jog selected by the Jog select input function. Jogging will continue as long as the destination is active. The motor will decelerate to a stop when the destination is deactivated. This is level sensitive.

Jog Select
Jog.Select0
This destination is used to select between the jogs. It is used along with the Jog.PlusActivate and Jog.MinusActivate destinations. If the Jog.Select0 destination is not active then the Jog.0 setup parameters will be used for jogging. If the Jog.Select0 input function is active, the Jog.1 setup parameters will be used for jogging. If the Jog.Select destination is changed during jogging motion the axis will ramp smoothly from the previously selected jog velocity to the new jog velocity using the specified jog acceleration. This is level sensitive.

Jog Stop
Jog.Stop
This is used only in programs to halt jogging motion. Jogging motion is initiated in programs using the Jog.#.MinusActivate or Jog.#.PlusActivate instructions, and using the Jog.Stop will cause the motor to decelerate to a stop at the Jog.#.Decel rate for the jog that is active.

Jog Acceleration
Jog.#.Accel
This parameter is the average acceleration ramp for the specific jog.

Jog Accelerating
Jog.#.Accelerating
This source is active while a jog is accelerating to its target velocity. Once the jog reaches the target velocity, the Jog.#.Accererating bit will turn off.

Jog At Velocity
Jog.#.AtVel
This source activates when the particular jog has reached its target velocity. It deactivates when accelerating or decelerating to another target jog velocity.

Jog Command Complete
Jog.#.CommandComplete
The Jog.#.CommandComplete source activates when the specific Jog completes its deceleration ramp and reaches zero commanded speed. It deactivates when the specific Jog is initiated again.

Jog Command In Progress
Jog.#.CommandInProgress
The Jog.#.CommandInProgress source is high throughout an entire jog profile. The bit goes high at the start of a jog acceleration ramp, and turns off at the end of a jog deceleration ramp.

Jog Deceleration
Jog.#.Decel
This parameter is the average deceleration ramp for the specific jog.

Jog Decelerating
Jog.#.Decelerating
This source turns on at the beginning of a jog deceleration ramp and turns off at the completion of the ramp.

Jog Initiate Minus
Jog.#.MinusInitiate
This is used inside a program to initiate a specific jog. When this bit is active, jogging motion will be initiated in the negative direction at the specified jog velocity.

Jog Initiate Plus
Jog.#.PlusInitiate
This is used inside a program to initiate a specific jog. When this bit is active, jogging motion will be initiated in the positive direction at the specified jog velocity.

Jog Time Base
Jog.#.TimeBase

The time base selects either realtime, which allows velocities, acceleration and deceleration to be based on real time, or synchronized, which allows for an external synchronization signal.

Jog Velocity
Jog.#.Vel

This parameter specifies the velocity used for jogging with the Jog.PlusActivate and Jog.MinusActivate destinations or the Jog.#.PlusInitiate and Jog.#.MinusInitiate inside a program. The units for this parameter are specified in the User Units view.

Master Absolute Home Defined
MasterAxis.AbsHomeDefined

See Absolute Home Defined. This parameter is a duplicate of Absolute Home Defined, except it applies to the Master Axis absolute position feedback instead of the Motor Axis.

Master Absolute Home Position Decimal Places
MasterAxis.AbsHomePosnDecimalPlaces

See Absolute Home Position Decimal Places. This parameter is a duplicate of Absolute Home Position Decimal Places, except it applies to the Master Axis instead of the Motor Axis.

Master Absolute Home Position in User Units
MasterAxis.AbsHomePosnUserUnits

See Absolute Home Position in User Units. This parameter is a duplicate of Absolute Home Position in User Units, except it applies to the Master Axis instead of the Motor Axis.

Master Absolute Home Rev Count
MasterAxis.AbsHomeRevCount

See Absolute Home Rev Count. This parameter is a duplicate of Absolute Home Rev Count, except it applies to the Master Axis instead of the Motor Axis.

Master Absolute Home Rev Position
MasterAxis.AbsHomeRevPosn

See Absolute Home Rev Position. This parameter is a duplicate of Absolute Home Rev Position, except it applies to the Master Axis instead of the Motor Axis.

Master Absolute Position Auto-Calculate Enable
MasterAxis.AbsPosnAutoCalculateEnable

See Absolute Position Auto-Calculate Enable. This parameter is a duplicate of Absolute Position Auto-Calculate Enable, except it applies to the Master Axis instead of the Motor Axis.

Master Absolute Position Mode
MasterAxis.AbsPosnMode

See Absolute Position Mode. This parameter is a duplicate of Absolute Position Mode, except it applies to the Master Axis instead of the Motor Axis.

Master Axis Encoder Revolution Counter
MasterAxis.MasterEncRevCount

This read-only parameter displays the number of whole revolutions that the master encoder signal has moved since powered up (absolute) and can be found on the Online tabs on the Status and Master Setup views. The units for this parameter are always revolutions since those are the units used by the drive and it's option modules. The value for this parameter comes from parameter **x.028** or **x.128** depending on the feedback encoder source selection.

In order for this parameter to work properly, a master encoder must be connected to the drive encoder interface or the SI-Universal Encoder option module, and the MasterAxis.SpeedFeedbackSelector must be configured properly.

Master Axis Encoder Position
MasterAxis.MasterEncRevPosition

This read-only parameter displays the number of fractions of a revolution that the master encoder signal has moved, and can be found on the Online tabs of the Status and Master Setup views. The units for this parameter are always 1/(2^16) of a revolution since those are the units used by the drive and its option modules. The value for this parameter comes from parameter **x.029**/**x.030** or **x.129**/**x.130** depending on the feedback encoder source selection.

In order for this parameter to work properly, a master encoder must be connected to the drive encoder interface or the SI-Universal Encoder option module, and the MasterAxis.SpeedFeedbackSelector must be configured properly.

Master Axis Feedback Selector
MasterAxis.SpeedFeedbackSelector

The MasterAxis.Speed Feedback Selector parameter determines where the feedback device for the master signal is connected to.

This parameter is read-only from within a user program.

Master Axis Characteristic Distance
MasterDistUnits.CharacteristicDistance

The Master Axis Characteristic Distance is the numerator in the scaling function used to define the relationship between user units of the master and revolution(s) of the master encoder. The user must define how many master user units there are in X number of revolutions of the master encoder. X in this case is the Master Axis Revs parameter (see below).

Example:

A master encoder is attached directly to a pulley measuring 3 inches in diameter. The user wishes to use units of Inches on the master axis. The circumference of the pulley is therefore, Diameter *PI = 3.0 * 3.14 = 9.425 inches. This means the master encoder would travel 9.425 inches for each revolution of the pulley. Therefore, the user would set the Characteristic Distance to 9.425 and the Master Rev to 1. If the encoder was attached instead to a different pulley that runs at a 3:1 ratio as the original pulley, the user could still set the Character Distance to 9.425, but instead set the Master Revs to 3.

This parameter can be modified within a user program. Extreme caution should be used to prevent the change of scaling when the synchronized motion is active. The motor should be stopped and the drive disabled when changing scaling factors in a program.

Changing this parameter within a user program or from an HMI will automatically clear the AbsolutePosnValid signal, and for Absolute encoders the AbsoluteHomeDefined signal. This means that when the values are changed, the machine must be homed again.

Master Axis Revs
MasterDistUnits.MasterRevs

The Master Axis Revs is the denominator in the scaling function used to define the relationship between user units of the master and revolution(s) of the master encoder. The user must define how many master user units there are in X number of revolutions of the master encoder. X in this case is the Master Axis Revs parameter. The number of user units in the ratio is the Master Axis Characteristic Distance (see above).

Example:

A master encoder is attached directly to a pulley measuring 3 inches in diameter. The user wishes to use units of Inches on the master axis. The circumference of the pulley is therefore, Diameter *PI = 3.0 * 3.14 = 9.425 inches. This means the master encoder would travel 9.425 inches for each revolution of the pulley. Therefore, the user would set the Characteristic Distance to 9.425 and the Master Rev to 1. If the encoder was attached instead to a different pulley that runs at a 3:1 ratio as the original pulley, the user could still set the Character Distance to 9.425, but instead set the Master Revs to 3.

This parameter can be modified within a user program. Extreme caution should be used to prevent the change of scaling when the synchronized motion is active. The motor should be stopped and the drive disabled when changing scaling factors in a program.

Changing this parameter within a user program or from an HMI will automatically clear the AbsolutePosnValid signal, and for Absolute encoders the AbsoluteHomeDefined signal. This means that when the values are changed, the machine must be homed again.

Module Gains Enable
ModuleGainsEnable

By default, the PTi210 module will calculate values for the drive's Current Loop and Velocity Loop based on motor/load information entered by the user. These calculated gains are then sent from the module to the drive on every power cycle or warm-start. In some cases, it may be desirable for the user to enter their own custom gain values using the drives keypad. In order to avoid overwriting those custom gains on the next power up, the PTi210 module must be commanded not to send the calculated gains. This can be done by clearing the Module Gains Enable check box found on the Tuning View. By default, the check box is active (selected) implying that the PTi210 module will calculate gains and send them to the drive.

Module Serial Number
ModuleSerialNumber

This is the PTi210 module serial number.

PTi210 Module Temperature

ModuleTemperature

This parameter displays the current temperature measured on the PTi210 module. A module error will be generated if the module temperature reaches or exceeds 89 °C. Available only in "Too Much Mode".

When the temperature reported by the module reaches or exceeds 84 °C, the internal fan of the drive will be forced to full speed. The fan will remain at full speed until the temperature drops below 79 °C.

Motion Stop

MotionStop

This destination is used to stop all motion operating without stopping programs. MotionStop can be activated through an assignment, or in a user program. This function is level sensitive, meaning that as long as MotionStop is active, all motion will be prevented. If a program has a motion statement, the program will wait on that line of code until the MotionStop function has been deactivated. If motion is in progress when MotionStop is activated, the profile will decelerate to zero velocity at the deceleration rate specified in the Stop.Decel parameter. All motion will stop using a realtime deceleration, regardless of the motions original timebase.

Motor Type

MotorType

This parameter is used to select the motor type.

Name

Name

User name for this PTi210 module axis can have a length up to 12 characters. This can be used to help differentiate setup files.

Overload Accumulator

OverloadAccumulator

This parameter gives an indication of the motor temperature based on a simple thermal model. The formula for the thermal model is a function of the current demand and a thermal time constant of the motor (parameter Pr **04.014** and found in the .ddf file). This read-only parameter gives an estimated motor temperature as a percentage of maximum motor temperature. When the calculated temperature reaches 100 %, the drive will limit the amount of current available to the motor until this parameter falls below 95 %.

NOTE

The overload accumulator is reset to zero on power-up. Therefore, if a motor is already hot (near its thermal limit), and power is cycled quickly, the accumulator is no longer a good indicator of motor temperature.

Packed Bits Control Word Value

PackedBits.ControlWord.#.Value

When accessing the Packed Bits Control Words for network mapping or use in a program, the Value parameter is what holds the data. When used in a program, the "PackedBits." and the ".Value" portions of the parameter name are optional. Therefore, to write to the Control Word Value in a program, the following lines of code are all interchangeable:

```
PackedBits.ControlWord.0.Value = 5
PackedBits.ControlWord.0 = 5
ControlWord.0.Value = 5
ControlWord.0 = 5
```

Packed Bits Status Word Value

PackedBits.StatusWord.#.Value

When accessing the Packed Bits Status Words for network mapping or use in a program, the Value parameter is what holds the data. When used in a program, the "PackedBits." and the ".Value" portions of the parameter name are optional. Therefore, to read the Status Word Value in a program, the following lines of code are all interchangeable:

```
Var.MyVar = PackedBits.StatusWord.0.Value
Var.MyVar = PackedBits.StatusWord.0
Var.MyVar = StatusWord.0.Value
Var.MyVar = StatusWord.0
```

PLS Direction
PLS.#.Direction
This parameter specifies the direction of motion that a particular PLS output will function. If set to Both, the PLS will activate regardless of whether the axis is moving in the positive or negative direction. If set to Plus, the PLS will activate only when the axis is moving in the positive direction. If set to Minus, the PLS will activate only when the axis is moving in the negative direction. A flying cutoff or flying shear application may use this feature to activate the PLS to fire the knife only when the axis is moving in the positive direction.

PLS Off Point
PLS.#.OffPosn
PLS.#.Status will be active when the selected source position is between the PLS.#.OnPosn and the PLS.#.OffPosn. The terms On and Off assume you are traveling in a positive direction. Assume that the PLS.#.Direction is set to "Both". When traveling in the positive direction and the position feedback reaches the OnPosn, the PLS.#.Status will activate. As the motor continues in the same direction, the PLS.#.Status will deactivate when feedback position reaches or exceeds the OffPosn. If motor travel changes to the negative direction, the PLS.#.Status will activate when position feedback reaches the OffPosn, and will deactivate when it continues past the OnPosn. The important thing to remember is that the PLS.#.Status will be active if between the PLS On and Off points.

If using negative values for the OnPosn and OffPosn, the most negative value should go in the OnPosn parameter, and the least negative value should go in the OffPosn.

If the PLS has a rollover point, and the OnPosn is greater than the OffPosn, the PLS will be active whenever the axis is not between the On and Off positions, and inactive whenever the axis is between the two positions. However, the PLS.#.Status will not turn on until it reaches the OnPosn the first time.

PLS On Point
PLS.#.OnPosn
PLS.#.Status will be active when the selected source position is between the PLS.#.OnPosn and the PLS.#.OffPosn. The terms On and Off assume the motor is traveling in a positive direction. Assume that the PLS.#.Direction is set to "Both". When traveling in the positive direction and the position feedback reaches the OnPosn, the PLS.#.Status will activate. As the motor continues in the same direction, the PLS.#.Status will deactivate when feedback position reaches or exceeds the OffPosn. If motor travel changes to the negative direction, the PLS.#.Status will activate when position feedback reaches the OffPosn, and will deactivate when it continues past the OnPosn. The important thing to remember is that the PLS.#.Status will be active if between the PLS On and Off points.

If using negative values for your OnPosn and OffPosn, the most negative value should go in the OnPosn parameter, and the least negative value should go in the OffPosn.

If the PLS has a rollover point, and the OnPosn is greater than the OffPosn, the PLS will be active whenever the axis is not between the On and Off positions, and inactive whenever the axis is between the two positions. However, the PLS.#.Status will not turn on until it reaches the OnPosn the first time.

PLS Enable
PLS.#.PLSEnable
This destination is used to enable an individual PLS. A PLS can be enabled though the Assignments view in PowerTools Studio or from a program. If enabled, the PLS will begin to function as soon as the drive has been homed or a DefineHome destination has been activated. Master Posn Valid must be active (Master Define Home is activated) if using a master signal for PLS source.

PLS Rollover Enable
PLS.#.RotaryRolloverEnable
This parameter is used to enable the RotaryRolloverPosn for the individual PLS.

PLS Rollover Position
PLS.#.RotaryRolloverPosn
This parameter is the absolute position of the first repeat position for this PLS. When enabled it causes the PLS to repeat every time this distance is passed. The repeating range begins at an absolute position of zero and ends at the RotaryRolloverPosn. For example in a rotary application a PLS could be setup with an OnPosn of 90 degrees and an OffPosn of 100 degrees. If the RotaryRolloverPosn is set to 360 degrees the PLS would come on at 90, go off at 100, go on at 450 (360+90), go off at 460 (360+100), go on at 810 (2*360+90), go off at 820 (2*360+100), and continue repeating every 360 degrees forever.

PLS Source
PLS.#.Source
PLSs can be assigned to three different sources: MotorPosnFeedback, MotorPosnCommand, or MasterPosnFeedback.
This parameter determines which position signal the PLS uses to reference its OnPosn and OffPosn in order to determine its PLS.#.Status parameter.

PLS Status
PLS.#.Status

This source is active when the position of the PLS source (motor or master) is greater than or equal to the OnPosn and less than the OffPosn.

Position Loop Response
PositionLoopResponse

The Position Loop Response is effectively a proportional gain for the PTi210 module position loop. The units for the parameter are Hz. A larger value for the Position Loop Response will result in a tighter position loop (less following error), however if it is set too high, instability can occur.

See the Tuning section of this manual for tuning procedure.

Positive Direction
PositiveDirection

This bit is used to select which direction of motor rotation is considered motion in the positive direction. Select from CW or CCW.

Position Command
PosnCommand

Position command sent to the drive by the PTi210 module. This parameter does not take following error into account. See also PosnFeedback and FollowingError. Units are in user units.

Position Feedback
PosnFeedback

Feedback position is the actual motor position in user units. PosnCommand minus the PosnFeedback is the FollowingError.

PTi210 Module Power Up Count
PowerUpCount

Power Up Count is the current value of how many times the PTi210 module has been powered up. Each time power is cycled to the system, this number increments by one. This parameter is stored in the PTi210 module, and is not reset if the module is switched to another drive. This parameter is read-only and can be found on the Errors view while online with PowerTools Studio.

PTi210 Module Power Up Time
PowerUpTime

The Power Up Time is the time elapsed since power has been cycled to the PTi210 module. The units for the parameter are Hours with a resolution of 0.1 Hours. This parameter is read-only and can be found on the Errors view while online with PowerTools Studio.

Profile Accelerating
Profile.#.Accelerating

This source will be active when the motion being run on the specified profile is accelerating to its programmed velocity. When the motion has reached its programmed velocity, this function will deactivate. This allows the user to see when any motion being run on this profile is accelerating rather than having to monitor each motion object individually.

Profile At Velocity
Profile.#.AtVel

This source is active when the motion being run on the specified profile is running at the programmed velocity. This function will activate after the acceleration ramp is completed, and before the deceleration ramp begins. This allows the user to see when any motion being run on this profile is at its programmed velocity rather than having to monitor each motion object individually.

Profile Command Complete
Profile.#.CommandComplete

This source activates when the commanded motion for a motion object running on the specified profile is completed. The function will remain active until the next motion is initiated on the same profile. If the MotionStop of the Stop function is used to stop the motion running on the specified profile, the CommandComplete will not activate. The CommandComplete does not activate after a stop because the motor may not be in the desired end position of the motion. This allows the user to see when any motion being run on this profile is complete rather than having to monitor each motion object individually.

NOTE

Activation of the CommandComplete signal does not mean that the motor is no longer moving. If there is any following error at the end of the motion, the CommandComplete will turn in before the actual motor motion is stopped.

Profile Command In Progress
Profile.#.CommandInProgress

This source is active while any motion is being commanded on the specified profile. This function is active during all segments of a motion (Accel, AtVel, and Decel). This function will deactivate when the CommandComplete signal activates.
The CommandInProgress signal can be active without actual motor movement if the master encoder stops during gearing or synchronized motion. This allows the user to see when any motion being run on this profile is in progress rather than having to monitor each motion object individually.

Profile Decelerating
Profile.#.Decelerating

This source will be active when the motion being run on the specified profile is decelerating to zero velocity (or to the next programmed velocity). When the motion has reached zero velocity, or its next programmed velocity, this function will deactivate. This allows the user to see when any motion being run on this profile is decelerating rather than having to monitor each motion object individually.

Profile Feedhold
Profile.#.Feedhold

This function is used to suspend or pause a profile in motion without stopping it altogether. The Feedhold effects all types of motion except for Gearing. When activated, any motion being run on the specified profile will decelerate to a stop in the time programmed in the FeedholdDecelTime parameter. The motion will remain stopped as long as the function is active. When deactivated, the motion will accelerate back up to the programmed speed in the same amount of time to finish its profile.

Profile Motion Stop
Profile.#.MotionStop

This function is used to stop any motion operating on the specified profile. This allows the user to stop motion running on one profile without stopping motion on both profiles. When activated, motion running on the specified profile will decelerate to a stop using the deceleration rate programmed in the StopDecel parameter. The profile will decelerate using a real-time deceleration ramp regardless of the original timebase of the move.

Program Any Complete
Program.AnyComplete

This source is activated when any program ends normally. If a program ends due to a fault or the stop destination, this source does not activate. Deactivates when any program is initiated.

Program X Global Where Am I Enable
Program.#.GlobalWhereAmIEnable

PowerTools Studio has a feature called “Global Where Am I” that, while active, causes a blue arrow to follow the program as it is processed, thereby showing the user what part of a program is being run. The blue arrow will cause the PowerTools Studio to change views when processing switches from one program to another (on the same Task). Sometimes it may be desirable to prevent PowerTools Studio from changing views when it switches programs. Therefore, the user can enable/disable the ability of PowerTools Studio to switch views automatically. If the Global Where Am I Enable check box is active, the view will switch to that program when it is being processed. If the Global Where Am I Enable is not active, then the view will not switch to that given program when program flow is passed to it.

Program Initiate
Program.#.Initiate

When activated, this destination initiates the specified program unless an index, home, or jog is already executing, a stop is active, or a program is already executing with the same task number.

Program Name
Program.#.Name

This is a character string which the user can assign to an individual program. It allows the user to give a descriptive name to programs for ease of use.

Program Complete
Program.#.ProgramComplete

This source is activated when a specific program ends normally. If the program ends due to a fault or the stop destination, this source does not activate. Deactivates when the specific program is initiated again.

Program X Run Anytime Enable
Program.#.RunAnytimeEnable

By default, the drive must be enabled for user programs to operate. There are some cases where a user may want a program to run even when the drive is not enabled. Therefore, if the user checks the “Run Anytime” check box, that particular program will be allowed to run under any condition. It is recommended to not put motion instructions in a Run Anytime program because if the motion instruction is processed while the drive is disabled, the program will stick on that particular instruction.

Program Stop
Program.#.Stop

This destination is used to stop a specific program from processing. It can be used to halt a program that is currently in operation, or to prevent a program from initiating. If a program has initiated some motion, and the program is stopped while that motion is still in progress, the motion will NOT be stopped. The motion initiated by the stopped program will continue until it is complete (i.e. indexes), or until it is stopped by another program (i.e. jog, gear). This function is edge sensitive meaning that when the Program.#.Stop activates, the specified program will be stopped, but not prevented from starting again.

Queue Name
Queue Name

You can assign a descriptive name to each queue, making the setup easier to follow. The length of the text string is limited by the column width with a maximum of 12 characters. Simply double click on the Name field of any queue’s line to assign a name to it.

Queue Size
Queue.#.QueueSize

This is the maximum number of elements that can be stored in the queue. If more than this number of pieces of data is in the queue at a time, then a Queue Overflow event will activate.

Queue Data In
Queue.#.DataIn

Data is loaded into the queue using the Queue.#.DataIn instruction in a program. When DataIn is set equal to value, that value is entered into the queue and the queue offset is added to it. If Queue Overflow is active, then no more data can be put into the Queue.

Queue Data Out
Queue.#.DataOut

Queue.#.DataOut is the value of the oldest piece of data in the queue. The sum of Queue,#,DataOut and the Queue offset is equal to the parameter Queue.#.ExitPosition.

Queue Full Level
Queue.#.FullLevel

The amount of data in the queue is constantly monitored and the Queue Full source will activate when the number of pieces of data in the queue exceeds the Full Level parameter. This is only a flag and does not indicate a fault of any kind.

Queue Clear
Queue.#.QueueClear

This destination automatically clears all of the data out of the queue. The cleared data is not saved and there is no way to recover the cleared data. This is typically activated on power-up of the system to make sure no old data remains in the queue.

Queue Compare Enable
Queue.#.QueueCompareEnable

The Compare Enable is what causes the comparator internal to the queue to function. If the Compare Enable is inactive, then the Queue Exit source will never activate. If activated, then the Queue Exit source will activate when the Queue Data plus the Queue Offset is equal to the Comparator Select parameter.

Queue Empty
Queue.#.QueueEmpty

This source is active if no data is stored in the queue. It will become inactive when the first piece of data is loaded into the queue and remain inactive until all data has been removed from the queue.

Queue Exit
Queue.#.QueueExit
This event activates when the source parameter is equal to the QueueExitPosition. Queue Exit deactivates when the Queue Remove instruction is processed.

Queue Exit Position
Queue.#.ExitPosition
Queue Exit Position is the sum of Queue Data Out and the Queue Offset. When the Queue Exit Position is equal to the selected source parameter. Then the Queue Exit event activates. Queue Exit position is only updated when the first piece of data is put into the queue or when a Queue Remove instruction is processed.

Queue Full
Queue.#.QueueFull
The Queue Full source will activate if the number of pieces of data in the queue equals or exceeds the Full Level parameter. The source will deactivate when the number of pieces of data in the queue is less than the Full Level.

Queue Offset
Queue.#.QueueOffset
The Queue Offset is the value that is added to the Queue Data Out and then compared to the source parameter to determine when the Queue Exit event activates. For instance, if Comparator Select is set to Feedback Position, and the Queue Offset is set to 10, and the user puts the value 5 into the queue, the queue exit function will activate when the Feedback Position is equal to 5 + 10 or 15.

Queue Overflow
Queue.#.QueueOverflow
This source activates when there is no more room in the queue to store data. The maximum number of pieces of data is determined by the Queue Size parameter.

Queue Remove
Queue.#.Remove
The Queue Remove instruction is used in the program to remove data from the queue. When processed, the oldest piece of data will be deleted out of the queue. The Queue Remove instruction also deactivates the Queue Exit function.

Queue Source
Queue.#.Source
The Queue Source determines which parameter the sum of the Queue Data and Queue Offset are compared to in order to activate the Queue Exit function. If set to Position Feedback, the sum of the data and offset are compared to the Position Feedback parameter. If set to Master Position, then the sum is compared to the Master Feedback Position parameter, and if set to Command Position, then the sum is compared to the Motor Commanded Position.

Rotary Rollover Enable
RotaryRolloverEnable
This parameter is used in applications with a predefined repeat length. One example would be a rotary table with a rotary rollover position of 360 degrees. The position will rollover to zero when the axis position gets to 360 degrees. (358, 359, 359.999, 0.0000, 1, 2, and so on.) The rollover point is defined to be exactly the same position as 0.

Selector Input Destinations
Selector.SelectLinesUsed
The selector is a binary to decimal decoder. This parameter selects the number of destinations (input lines) to be used by the selector. The number of lines used determines the number of sources (selections) that can be made by the selector; that is 2 input lines can select 4 destinations (selections), 5 input lines can select 32 destinations. Range is 1 to 8.

Initiate
Selector.SelectorInitiate
When this destination is activated, the selector checks the status of all Selector.Select destinations to determine which Selector.Selection to activate. The Selector.SelectorInitiate function is level sensitive meaning that as long as this signal is active the selection outputs from the selector will automatically update.

Select
Selector.#.Select

This source selects Binary inputs to the selector, usually assigned to input lines. This is level sensitive.

Selection
Selector.#.Selection

This source selects Decimal outputs from the selector, assigned to indexes, homes or programs.

Slot 1 Error Status
Slot1.ErrorStatus

If the Unidrive M/Digitax HD trips due to an error in an option module populated in Slot 1, this parameter will store/display the Error Code for the specific error.

For details on the specific error code, refer to the user guide for the particular option module.

Slot 2 Error Status
Slot2.ErrorStatus

If the Unidrive M/Digitax HD trips due to an error in an option module populated in Slot 2, this parameter will store/display the Error Code for the specific error.

For details on the specific error code, refer to the user guide for the particular option module.

Slot 3 Error Status
Slot3.ErrorStatus

If the Unidrive M trips due to an error in an option module populated in Slot 3, this parameter will store/display the Error Code for the specific error.

For details on the specific error code, refer to the user guide for the particular option module.

SlotX Encoder Comms Baud Rate
SlotX.EncoderCommsBaudRate

This parameter is available only when using an Encoder with communications protocol. The Encoder Comms Baud Rate list box provides direct access to parameter **x.037** from the SI-Universal Encoder module configuration menu. This parameter defines the baud rate used for communications between the encoder hardware and the SI-Universal Encoder module. See the SI-Universal Encoder section of this manual or the *SI-Universal Encoder User Guide* for more information.

SlotX Encoder Comms Resolution
SlotX.EncoderCommsResolution

This parameter is available only when using an Encoder with communications protocol. The Encoder Comms Resolution text box provides direct access to parameter **x.035** from the SI-Universal Encoder module configuration menu. This parameter defines the maximum resolution of the absolute position of the encoder transmitted from the encoder to the SI-Universal Encoder module. The value is entered in a number of bits of resolution. See the SI-Universal Encoder section of this manual or the *SI-Universal Encoder User Guide* for more information.

SlotX Encoder Enable Auto Configuration
SlotX.EncoderEnableAutoConfiguration

This parameter is available only when using an Encoder with communications protocol. The Enable Auto Configuration check box provides direct access to parameter **x.041** from the SI-Universal Encoder module configuration menu. When a SC.Hiper, SC, EnDat, or EnDat encoder is being used, the SI-Universal Encoder can interrogate the encoder for the necessary configuration parameters (**x.033**, **x.034**, **x.035**, **x.037**, **x.052**, **x.053**, **x.054**, **x.060**, **x.061**, **x.062**, **x.063** depending on encoder type) automatically. To enable the auto configuration, the check box should be active (checked), to disable the feature and manually configure the parameters, the check box should be unchecked. See the SI-Universal Encoder section of this manual or the *SI-Universal Encoder User Guide* for more information.

SlotX Encoder Lines Per Revolution
SlotX.EncoderLinesPerRev

This parameter is only available when an SI-Universal Encoder or SI-I/O 24 Plus option module has been populated in one of the Unidrive M/Digitax HD slots.

The X in “SlotX” defines the slot number that the Encoder option module has been fitted in. For example, if the Universal Encoder option module is in slot 2, the parameter would be named Slot2.EncoderLinesPerRev.

The EncoderLinesPerRev defines the resolution of the encoder hardware. This is equal to the number of lines on the encoder (pre-quadrature value). The range for the Lines Per Rev is 0 to 100000. This parameter is found on the SlotX view after a SI-Universal Encoder or SI-I/O 24 Plus option module has been selected for Module Type. For more information on this parameter, please refer to the *SI-Universal Encoder User Guide* or *SI-I/O 24 Plus User Guide* (parameter **x.034**).

SlotX Simulated Encoder Denominator
SlotX.EncoderSimulationDenominator

This parameter is only available when an SI-Universal Encoder option module has been populated in one of the Unidrive M/Digitax HD slots.

The X in “SlotX” defines the slot number that the SI-Universal Encoder module has been fitted in. For example, if the Universal Encoder module is in slot 2, the parameter would be named: Slot2.EncoderSimulationDenominator.

The SI-Universal Encoder is capable of sending out a simulated encoder signal. To configure the simulated encoder signal, the user specifies a source parameter and a ratio multiplier. The multiplier allows the user to scale the source parameter before it is sent out the “Encoder Out” port. The multiplier is made up of a numerator parameter and a denominator parameter. The formula for the value sent out the encoder output is as follows:

Output = Enc. Sim. Source * (Enc. Sim. Numerator / Enc. Sim. Denominator)

The range for the Denominator is 1 to 65536. This parameter is found on the SlotX view after a SI-Universal Encoder module has been selected for the Module Type. For more information on this parameter, please refer to the *SI-Universal Encoder User Guide* (parameter **x.094**).

SlotX Simulated Encoder Numerator
SlotX.EncoderSimulationNumerator

This parameter is only available when an SI-Universal Encoder option module has been populated in one of the Unidrive M/Digitax HD slots.

The X in “SlotX” defines the slot number that the SI-Universal Encoder option module has been fitted in. For example, if the Universal Encoder module is in slot 2, the parameter would be named: Slot2.EncoderSimulationNumerator.

The SI-Universal Encoder is capable of sending out a simulated encoder signal. To configure the simulated encoder signal, the user specifies a source parameter and a ratio multiplier. The multiplier allows the user to scale the source parameter before it is sent out the “Encoder Out” port. The multiplier is made up of a numerator parameter and a denominator parameter. The formula for the value sent out the encoder output is as follows:

Output = Enc. Sim. Source * (Enc. Sim. Numerator / Enc. Sim. Denominator)

The range for the Numerator is 1 to 65536. This parameter is found on the SlotX view after a SI-Universal Encoder module has been selected for the Module Type. For more information on this parameter, please refer to the *SI-Universal Encoder User Guide* (parameter **x.093**).

SlotX Simulated Encoder Source
SlotX.EncoderSimulationSource

This parameter is only available when an SI-Universal Encoder option module has been populated in one of the Unidrive M/Digitax HD slots.

The X in “SlotX” defines the slot number that the SI-Universal Encoder option module has been fitted in. For example, if the Universal Encoder module is in slot 2, the parameter would be named: Slot2.EncoderSimulationSource.

The SI-Universal Encoder is capable of sending out a simulated encoder signal. To configure the simulated encoder signal, the user specifies a source parameter and a ratio multiplier. The multiplier allows the user to scale the source parameter before it is sent out the “Encoder Out” port. The multiplier is made up of a numerator parameter and a denominator parameter. The formula for the value sent out the encoder output is as follows:

Output = Enc. Sim. Source * (Enc. Sim. Numerator / Enc. Sim. Denominator)

The range for the source parameter is 0.000 to 59.999. This parameter is found on the SlotX view after a SI-Universal Encoder module has been selected for the Module Type. For more information on this parameter, please refer to the *SI-Universal Encoder User Guide* (parameter **x.085**).

SlotX Encoder Supply Voltage
SlotX.EncoderSupplyVoltage

This parameter is available when using a SI-Universal Encoder or SI-I/O 24 Plus module. The Encoder Supply Voltage list box provides direct access to parameter **x.036** from the SI-Universal Encoder module configuration menu. To support many different applications, the SI-Universal Encoder module can provide several different supply voltages. Select from 5 V, 8 V, or 15 V to be provided from the module hardware. See the SI-Universal Encoder section of this manual or the *SI-Universal Encoder User Guide* for more information.

SlotX Encoder Turns
SlotX.EncoderTurns

This parameter is available when using a SI-Universal Encoder or SI-I/O 24 Plus module. The Encoder Turns text box provides direct access to parameter **x.013** from the SI-Universal Encoder module configuration menu or parameter **x.033** from the SI-I/O 24 Plus module configuration menu. This parameter defines how many revolutions will be recorded before the Revolution Counter (**x.028**) rolls over. The parameter is entered in a number of bits of resolution. To determine the associated number of revs, use the formula: Revs before rollover = 2n where n is the value of Encoder Turns. See the SI-Universal Encoder section of this manual or the *SI-Universal Enc User Guide* for more information.

SlotX Encoder Type
SlotX.EncoderType

This parameter is available only when using a SI-Universal Encoder or SI-I/O 24 Plus module. The Encoder Type list box provides direct access to parameter **x.038** from the module configuration menu. This parameter allows the module to support different encoder types. Select the desired type of encoder from this list box. See the SI-Universal Encoder section of this manual or the relevant encoder option user guide for more information.

SI-I/O Module Input Name
SlotX.DigitalInput.#.Name

This parameter is only available for SI-I/O modules that have been populated in one of the Unidrive M/Digitax HD slots.

The X in “SlotX” defines the slot number that the SI-I/O module has been fitted in. For example, if the I/O module is in slot 2, the parameter would be named Slot2.Input.#.Name.

Each digital input on the SI-I/O module can be given a name. The name can be used in a user program to reference a specific input. The name can be up to 12 alphanumeric characters, but must begin with a non-numeric character.

SI-I/O Module IO Direction
SlotX.DIO.#.Direction

This parameter is only available for SI-I/O modules that has been populated in one of the Unidrive M/Digitax HD slots.

The X in “SlotX” defines the slot number that the SI-I/O module has been fitted in. For example, if the I/O module is in slot 2, the parameter would be named Slot2.DIO.#.Direction.

The first four digital I/O points on the SI-I/O module (pins 2, 3, 4 and 5) are configured as Inputs or Outputs using PowerTools Studio software. The SlotX.DIO.#.Direction parameter is used to configure whether the I/O point acts as a digital input or digital output. Valid entries for this parameter are selInput or selOutput. This parameter is modified automatically when the user changes the settings on the Slot X view.

SI-I/O Module IO Status
SlotX.DIO.#.In

This parameter is only available for SI-I/O modules that has been populated in one of the Unidrive M/Digitax HD slots.

The X in “SlotX” defines the slot number that the SI-I/O module has been fitted in. For example, if the I/O module is in slot 2, the parameter would be named Slot2.DIO.#.In.

The first four digital I/O points on the SI-I/O module (pins 2, 3, 4 and 5) are configured as Inputs or Outputs using PowerTools Studio software. If the I/O point has been configured as a digital input, then the SlotX.DIO.#.In signal can be used in a user program to check the status of inputs, or on the Assignments view to activate different signals.

NOTE

This parameter will appear on the Assignments view under the Inputs group of Sources even if the I/O point has been configured as a digital output.

SlotX IO Name
SlotX.DIO.#.Name

This parameter is only available for SI-I/O modules that have been populated in one of the Unidrive M/Digitax HD slots.

The X in “SlotX” defines the slot number that the SI-I/O module has been fitted in. For example, if an SI-I/O module is in slot 2, the parameter would be named Slot2.DIO.#.Name.

Each digital I/O point on the SI-I/O module can be given a name. The name can be used in a user program to reference a specific I/O point. The name can be up to 12 alphanumeric characters, but must begin with a non-numeric character.

Slot X IO State
SlotX.DIO.#.Out

This parameter is only available for SI-I/O modules that have been populated in one of the Unidrive M/Digitax HD slots.

The X in “SlotX” defines the slot number that the I/O module has been fitted in. For example, if a SI-I/O module is in slot 2, the parameter would be named Slot2.DIO.#.Out.

The first four digital I/O points on the SI-I/O module (pins 2, 3, 4 and 5) are configured as Inputs or Outputs using PowerTools Studio software. If the I/O points have been configured as digital outputs, then the SlotX.DIO.#.Out signal can be used in a user program to activate an output, or on the Assignments view to activate when a specified Source event activates. If assigned to a source on the Assignments view, when the source event activates, the output that it is assigned to will turn on.

NOTE

This parameter will appear on the Assignments view under the Outputs group of Destinations even if the I/O point has been configured as a digital input.

SlotX Relay Name
SlotX.Relay.#.Name

This parameter is only available for SI-I/O modules that have been populated in one of the Unidrive M/Digitax HD slots.

The X in “SlotX” defines the slot number that the I/O module has been fitted in. For example, if the I/O module is in slot 2, the parameter would be named Slot2.Relay.#.Name.

Each relay output on the I/O module can be given a name. The name can be used in a user program to reference a specific relay output if desired. The name can be up to 12 alphanumeric characters, but must begin with a non-numeric character.

SlotX Relay State
SlotX.Relay.#.Out

This parameter is only available for SI-I/O modules that have been populated in one of the Unidrive M/Digitax HD slots.

The X in “SlotX” defines the slot number that the I/O module has been fitted in. For example, if the I/O module is in slot 2, the parameter would be named Slot2.Relay.#.Out.

The SI-I/O (pins 21 and 23) modules have two relay outputs that can be used to control devices that require more current than a standard digital output. The “.Out” at the end of the parameter name is optional, so this parameter is often referred to simply as SlotX.Relay.#. This parameter can be used in a user program to activate the output (i.e. Slot2.Relay.8 = On), or can be assigned to a Source on the Assignment view.

SI-I/O 24 Plus Module Digital IO Read Word
SlotX.DigitalIOReadWord

This parameter is only available for SI-I/O 24 Plus modules that have been populated in one of the Unidrive M/Digitax HD slots.

The X in "SlotX" defines the slot number that the SI-I/O 24 Plus module has been fitted in. For example, if the I/O module is in slot 2, the parameter would be named Slot2.DigitalIOReadWord.

The states of all the SI-I/O 24 Plus module digital inputs are provided as a single 16-bit word in parameter Pr **2M.010** (*Input Register*), where **2M** refers to the slot specific setup menu for the IO parameters.

NOTE

This parameter is written to on initialization and cannot be written to from a user program.

SI-I/O 24 Plus Module Input Name
SlotX.IO24Input.#.Name

This parameter is only available for SI-I/O 24 Plus modules that have been populated in one of the Unidrive M/Digitax HD slots.

The X in "SlotX" defines the slot number that the SI-I/O 24 Plus module has been fitted in. For example, if the I/O module is in slot 2, the parameter would be named Slot2.IO24Input.#.Name.

Each digital input on the SI-I/O 24 Plus module can be given a name. The name can be used in a user program to reference a specific digital input and can be up to 12 alphanumeric characters, but must begin with a non-numeric character.

SI-I/O 24 Plus Module Input State
SlotX.IO24Input.#.In

This parameter is only available for SI-I/O 24 Plus modules that have been populated in one of the Unidrive M/Digitax HD slots.

The X in "SlotX" defines the slot number that the SI-I/O 24 Plus module has been fitted in. For example, if the I/O module is in slot 2, the parameter would be named Slot2.IO24Input.#.In.

NOTE

This is the default attribute for the SlotX.IO24Input.# parameter, and therefore it is not necessary to specify the complete reference in a user program.

SI-I/O 24 Plus Module Input Debounce Time
SlotX.IO24Input.#.DebounceTime

This parameter is only available for SI-I/O 24 Plus modules that have been populated in one of the Unidrive M/Digitax HD slots.

The X in "SlotX" defines the slot number that the SI-I/O 24 Plus module has been fitted in. For example, if the I/O module is in slot 2, the parameter would be named Slot2.IO24Input.#.DebounceTime.

Each digital input on the SI-I/O 24 Plus module can be configured with a debounce time of up to 2 seconds, the digital input state must be seen high for the entire period before the input is accepted as active.

SI-I/O 24 Plus Module Output Name
SlotX.IO24Output.#.Name

This parameter is only available for SI-I/O 24 Plus modules that have been populated in one of the Unidrive M/Digitax HD slots.

The X in "SlotX" defines the slot number that the SI-I/O 24 Plus module has been fitted in. For example, if the I/O module is in slot 2, the parameter would be named Slot2.IO24Output.#.Name.

Each digital output on the SI-I/O 24 Plus module can be given a name. The name can be used in a user program to reference a specific digital output and can be up to 12 alphanumeric characters, but must begin with a non-numeric character.

SI-I/O 24 Plus Module Output State
SlotX.IO24Output.#.Out

This parameter is only available for SI-I/O 24 Plus modules that have been populated in one of the Unidrive M/Digitax HD slots.

The X in "SlotX" defines the slot number that the SI-I/O 24 Plus module has been fitted in. For example, if the I/O module is in slot 2, the parameter would be named Slot2.IO24Output.#.Out.

NOTE

This is the default attribute for the SlotX.IO24Output.# parameter, and therefore it is not necessary to specify the complete reference in a user program.

Module Encoder Type
SlotX.EncoderType

This parameter is only available for option modules which support an encoder feedback signal that have been populated in one of the Unidrive M/Digitax HD slots.

The X in "SlotX" defines the slot number that the option module has been fitted in. For example, if the SI-I/O 24 Plus module is in slot 2, the parameter would be named Slot2.EncoderType.

This parameter selects the encoder type for the option module, it is equivalent to Pr **1M.038** (*Device Type*).

NOTE

This parameter is written to on initialization and cannot be written to from a user program.

Module Encoder Lines Per Revolution
SlotX.EncoderLinesPerRev

This parameter is only available for option modules which support an encoder feedback signal that have been populated in one of the Unidrive M/Digitax HD slots.

The X in "SlotX" defines the slot number that the option module has been fitted in. For example, if the SI-I/O 24 Plus module is in slot 2, the parameter would be named Slot2.EncoderLinesPerRev.

This parameter specifies the number of encoder lines per revolution, it is equivalent to Pr **1M.034** (*Lines Per Revolution*).

NOTE

This parameter is written to on initialization and cannot be written to from a user program

Module Encoder Lines Per Revolution Divider
SlotX.LinesPerRevDivider

This parameter is only available for option modules which support an encoder feedback signal that have been populated in one of the Unidrive M/Digitax HD slots.

The X in "SlotX" defines the slot number that the option module has been fitted in. For example, if the SI-I/O 24 Plus module is in slot 2, the parameter would be named Slot2.LinesPerRevDivider.

This parameter is the used as the denominator value in the scaling factor for the number of encoder lines per revolution, it is equivalent to Pr **1M.044** (*Reference Scaling*).

Module Encoder Turns
SlotX.EncoderTurns

This parameter is only available for option modules which support an encoder feedback signal that have been populated in one of the Unidrive M/Digitax HD slots.

The X in "SlotX" defines the slot number that the option module has been fitted in. For example, if the SI-I/O 24 Plus module is in slot 2, the parameter would be named Slot2.EncoderTurns.

This parameter specifies the number of encoder turns bits, it is equivalent to Pr **1M.033** (*Turns Bits*).

Module Encoder Additional Power Up Delay
SlotX.P1AdditionalPowerUpDelay

This parameter is only available for option modules which support an encoder feedback signal that have been populated in one of the Unidrive M/Digitax HD slots.

The X in "SlotX" defines the slot number that the option module has been fitted in. For example, if the SI-I/O 24 Plus module is in slot 2, the parameter would be named Slot2.P1AdditionalPowerUpDelay.

This parameter specifies an additional delay (in seconds) after the initial 100 ms delay before the PTi210 will communicate with the encoder, it is equivalent to Pr **1M.049** (*Additional Power Up Delay*).

Module Encoder Feedback Filter
SlotX.P1FeedbackFilter

This parameter is only available for option modules which support an encoder feedback signal that have been populated in one of the Unidrive M/Digitax HD slots.

The X in "SlotX" defines the slot number that the option module has been fitted in. For example, if the SI-I/O 24 Plus module is in slot 2, the parameter would be named Slot2.P1FeedbackFilter.

This parameter specifies a time period for a filter applied to the encoder feedback signal, and writes to Pr **1M.042** (*Feedback Filter*).

The value written to Pr **1M.042** is in accordance with the following table:

PowerTools Studio Setting	Value written to Pr 1M.042
Disabled	0
1 msec	1
2 msec	2
4 msec	3
8 msec	4
16 msec	5

NOTE

This parameter is written to on initialization and cannot be written to from a user program.

Module Encoder Feedback Reverse
SlotX.P1FeedbackReverse

This parameter is only available for option modules which support an encoder feedback signal that have been populated in one of the Unidrive M/Digitax HD slots.

The X in "SlotX" defines the slot number that the option module has been fitted in. For example, if the SI-I/O 24 Plus module is in slot 2, the parameter would be named Slot2.P1FeedbackReverse.

This parameter, when checked, specifies to reverse the direction of the encoder feedback signal, it is equivalent to Pr **1M.056** (*Feedback Reverse*).

Module Encoder Normalisation Turns
SlotX.P1NormalisationTurns

This parameter is only available for option modules which support an encoder feedback signal that have been populated in one of the Unidrive M/Digitax HD slots.

The X in "SlotX" defines the slot number that the option module has been fitted in. For example, if the SI-I/O 24 Plus module is in slot 2, the parameter would be named Slot2.P1NormalisationTurns.

This parameter specifies the number of turns bits used in the normalisation parameters, it is equivalent to Pr **1M.057** (*Normalisation Turns*).

Module Thermistor Fault Detection
SlotX.ThermistorFaultDetection

This parameter is only available for option modules which support a motor thermistor within the encoder feedback connector that have been populated in one of the Unidrive M/Digitax HD slots.

The X in "SlotX" defines the slot number that the option module has been fitted in. For example, if the SI-I/O 24 Plus module is in slot 2, the parameter would be named Slot2.ThermistorFaultDetection.

This parameter specifies the thermistor fault detection for the option module, and writes to Pr **1M.123** (*Thermistor Fault Detect*).

The value written to Pr **1M.123** is in accordance with the following table:

PowerTools Studio Setting	Value written to Pr 1M.123
None	0
Temperature	1
Temp or Short	2

NOTE

This parameter is written to on initialization and cannot be written to from a user program.

Module Thermistor Trip Threshold
SlotX.ThermistorTripThreshold

This parameter is only available for option modules which support a motor thermistor within the encoder feedback connector that have been populated in one of the Unidrive M/Digitax HD slots.

The X in "SlotX" defines the slot number that the option module has been fitted in. For example, if the SI-I/O 24 Plus module is in slot 2, the parameter would be named Slot2.ThermistorTripThreshold.

This parameter specifies the thermistor value (Ω) that will initiate the selected thermistor trip for the option module, it is equivalent to Pr **1M.120** (*Thermistor Trip Threshold*).

Module Thermistor Reset Threshold
SlotX.ThermistorResetThreshold

This parameter is only available for option modules which support a motor thermistor within the encoder feedback connector that have been populated in one of the Unidrive M/Digitax HD slots.

The X in "SlotX" defines the slot number that the option module has been fitted in. For example, if the SI-I/O 24 Plus module is in slot 2, the parameter would be named Slot2.ThermistorResetThreshold.

This parameter specifies the thermistor value (Ω) that will enable the thermistor trip to be reset for the option module, it is equivalent to Pr **1M.121** (*Thermistor Reset Threshold*).

Enable Software Travel Limits
SoftwareTravelLimitEnable

Software travel limits can be used to limit machine travel. They are often setup inside the hardware travel limits to add a level of protection from exceeding the machine's travel limits. The SoftwareTravelLimitMinusActive source (output function) is active when the SoftwareTravelLimitMinusPosn is reached or exceeded. Motion is halted using the TravelLimitDecel whenever a hardware or software travel limit is hit or exceeded. Software travel limits are not active unless Absolute Position Valid is active.

Software Travel Limit Minus Active
SoftwareTravelLimitMinusActive

The SoftwareTravelLimitMinusActive source is active when the SoftwareTravelLimitMinusPosn is reached or exceeded. Motion will come to a stop using the TravelLimitDecel ramp. Software travel limits are not active unless enabled and Absolute Position Valid is active.

Software Travel Limit Minus Position
SoftwareTravelLimitMinusPosn

The SoftwareTravelLimitMinusActive source will activate when the SoftwareTravelLimitMinusPosn is reached or exceeded. Motion will come to a stop using the TravelLimitDecel. Software travel limits are not active unless enabled and Absolute Position Valid is active.

Software Travel Limit Plus Active
SoftwareTravelLimitPlusActive

The SoftwareTravelLimitPlusActive source is active when the SoftwareTravelLimitPlusPosn is reached or exceeded. Motion will come to a stop using the TravelLimitDecel ramp. Software travel limits are not active unless enabled and Absolute Position Valid is active.

Software Travel Limit Plus Position
SoftwareTravelLimitPlusPosn

The SoftwareTravelLimitPlusActive source is active when the SoftwareTravelLimitPlusPosn is reached or exceeded. Motion is halted using the TravelLimitDecel whenever a hardware or software travel limit is hit or exceeded. Software travel limits are not active unless enabled and Absolute Position Valid is active.

Drive Analog Input X Channel Enable
AnalogInput.#.ChannelEnable

This check box, found on the Analog Inputs view, is used to enable the specific analog input channel for use by the PTi210 module. If the check box is clear, the analog input is not being used by the PTi210 module and the user is free to use it with alternate methods. However, once the enable check box is activated (selected), the user should not attempt to use the analog input from another module or from the keypad. When the check box is activated, several other configuration parameters will become available.

Drive Analog Input X Maximum Input Value
AnalogInput.#.MaxInputValue

The PTi210 module reads the value of the analog input in units of 0 to 100 % and then scales that using linear interpolation and then populates the specified destination parameter/variable. When the analog input reaches the Maximum Input Value, the destination variable will be equal to the Maximum User Value. This is used as a point on the curve for the linear interpolation.

Drive Analog Input X Maximum User Value
AnalogInput.#.MaxUserValue

The PTi210 module reads the value of the analog input in units of 0 to 100 % and then scales that using linear interpolation and then populates the specified destination parameter/variable. When the analog input reaches the Maximum Input Value, the destination variable will be equal to the Maximum User Value. This is used as a point on the curve for the linear interpolation.

Drive Analog Input X Minimum Input Value
AnalogInput.#.MinInputValue

The PTi210 module reads the value of the analog input in units of 0 to 100 % and then scales that using linear interpolation and then populates the specified destination parameter/variable. When the analog input reaches the Minimum Input Value, the destination variable will be equal to the Minimum User Value. This is used as a point on the curve for the linear interpolation.

Drive Analog Input X Minimum User Value
AnalogInput.#.MinUserValue

The PTi210 module reads the value of the analog input in units of 0 to 100 % and then scales that using linear interpolation and then populates the specified destination parameter/variable. When the analog input reaches the Minimum Input Value, the destination variable will be equal to the Minimum User Value. This is used as a point on the curve for the linear interpolation.

Drive Analog Input X Module Destination
AnalogInput.#.ModuleDestination

The analog input is read it units of 0 to 100 %. That value is then scaled and put into a destination variable of the users choice. This parameter is used to define which variable within the PTi210 module is the destination for the analog input. This can be any read/write parameter within the PTi210 module. For a list of the parameters, please refer to the Program – Drag In Variables tree.

Drive Analog Input X Raw Value
AnalogInput.#.RawValue

This parameter is read only and shows the value of the analog input before it is scaled into the user scaling. Therefore, the raw value will always range between 0 % and 100 %. This parameter can be found on the Analog Input view but is only functional while online with the PTi210 module.

Drive Analog Input X Set Maximum Input Value
AnalogInput.#.SetMax

This button can be found on the Analog Input view next to the Maximum Input text box. When this button is pressed, the PTi210 module reads the current value of the Analog Input and writes that value into the Maximum Input text box. This allows the user to easily set the analog input to its full-scale value and enter that value into the text box avoiding direct calibration of the input circuitry.

Drive Analog Input X Set Minimum Input Value
AnalogInput.#.SetMin

This button can be found on the Analog Input view next to the Minimum Input text box. When this button is pressed, the PTi210 module reads the current value of the Analog Input and writes that value into the Minimum Input text box. This allows the user to easily set the analog input to its minimum scale value and enter that value into the text box avoiding direct calibration of the input circuitry.

Drive Analog Input X Mode
AnalogInput.#.InputMode

On the Unidrive M700/M701, Analog Inputs 1 and 2 can function in a number of different operation modes, both are configured for Voltage Mode by default. From this list box, the user can select from 4-20 mA Low, 20-4 mA Low, 4-20 mA Hold, 20-4 mA Hold, 0-20 mA, 20-0 mA, 4-20 mA Trip, 20-4 mA Trip, 4-20 mA, 20-4 mA and Volt.

Analog Input 3 can be configured for either Voltage Mode or for thermistor protection with the default being Voltage Mode. From this list box, the user can select from Volt, Therm Short Cct, Thermistor and Therm No Trip.

The Unidrive M702 only provides one analog input (Analog Input 3) for thermistor protection and is therefore not accessible on the Analog Inputs view.

The Digitax HD only provides one analog input (Analog Input 1) configured for Voltage Mode. No configuration settings are accessible for this input.

Drive Analog Output X Channel Enable
AnalogOutput.#.ChannelEnable

This check box, found on the Analog Outputs view, is used to enable the specific analog output channel for use by the PTi210 module. If the check box is clear, the analog output is not being used by the PTi210 module, and the user is free to use it with alternate methods. However, once the enable check box is activated (selected), the user should not attempt to use the analog input from another module or from the keypad. When the check box is activated, several other configuration parameters will become available.

Drive Analog Output X Maximum Output Value
AnalogOutput.#.MaxOutputValue

The PTi210 module can read the value of a specified parameter and scale it into the Unidrive M700/M701's scale of 0 to 100 % of scale on the output. The PTi210 module uses linear interpolation to scale the value properly. When the selected source parameter is equal to the Maximum User Value, then the Analog Output will be equal to the Maximum Output Value. This parameter is used in conjunction with the Maximum User Value to define a single point on the curve used for linear interpolation.

Drive Analog Output X Maximum User Value
AnalogOutput.#.MaxUserValue

The PTi210 module can read the value of a specified parameter and scale it into the Unidrive M700/701's scale of 0 to 100 % of scale on the output. The PTi210 module uses linear interpolation to scale the value properly. When the selected source parameter is equal to the Maximum User Value, then the Analog Output will be equal to the Maximum Output Value. This parameter is used in conjunction with the Maximum Output Value to define a single point on the curve used for linear interpolation.

Drive Analog Output X Minimum Output Value
AnalogOutput.#.MinOutputValue

The PTi210 module can read the value of a specified parameter and scale it into the Unidrive M700/M701's scale of 0 to 100 % of scale on the output. The PTi210 module uses linear interpolation to scale the value properly. When the selected source parameter is equal to the Minimum User Value, then the Analog Output will be equal to the Minimum Output Value. This parameter is used in conjunction with the Minimum User Value to define a single point on the curve used for linear interpolation.

Drive Analog Output X Minimum User Value
SPAnalogOutput.#.MinUserValue

The PTi210 module can read the value of a specified parameter and scale it into the Unidrive M700/M701's scale of 0 to 100 % of scale on the output. The PTi210 module uses linear interpolation to scale the value properly. When the selected source parameter is equal to the Minimum User Value, then the Analog Output will be equal to the Minimum Output Value. This parameter is used in conjunction with the Minimum Output Value to define a single point on the curve used for linear interpolation.

Drive Analog Output X Module Variable Source
AnalogOutput.#.ModuleSource

If User Defined Module Variable is selected from the Source list box, then the user must define which variable from within the PTi210 module is to be used as the source for the analog output. When the user selects a variable, the current value of that variable will constantly be used to update the value of the analog output. For a list of variables that can be used, click on the ***Popup Variables*** button and the Select Variables From Tree window opens. Any 16-bit or 32-bit parameter can be used as the source variable.

Drive Analog Output X Scale
AnalogOutput.#.Scale

The Scale parameter is only available if the user has selected User Defined Drive Menu from the Source list box. The Scale parameter ranges from 0.000 to 10.000 and is multiplied with the selected source Menu Parameter to achieve the ultimate output value. This parameter directly accesses Pr **07.020** or Pr **07.023** depending on the specific analog output channel.

Drive Analog Output X Source
AnalogOutput.#.Source

This parameter is used to define the source value for the given analog output channel. The user can select from the following choices:

Predefined Module Variables

Following Error

Velocity Command

Velocity Feedback

User Defined Module Variable

User Defined Drive Menu

Predefined Drive Parameters

[Menu 3.02] Speed

[Menu 4.02] Active Current

[Menu 4.17] Reactive Current

[Menu 5.03] Output Power

Based on which Source parameter is selected, different scaling parameters will become available.

Drive Analog Output X Parameter Source
AnalogOutput.#.DriveMenuSource

If User Defined Drive Menu is selected from the Source list box, then the user must define which parameter from within the Drive's Menu structure is to be used as the source for the analog output. When the user selects a parameter, the current value of that parameter will constantly be used to update the value of the analog output. For a list of parameters that can be used, refer to the Drive Menu Watch view in PowerTools Studio, or the Unidrive M/Digitax HD *User Guide* or relevant drive *Parameter Reference Guide*.

I/O Status Word
DriveConnect.DigitalIOReadWord

This parameter is a bitmap that contains the status of the digital I/O on the Unidrive M/Digitax HD. This parameter is read-only and is used to control the virtual LEDs on the Drive I/O Setup view.

Unidrive M/Digitax HD Drive Mode
Drive.DriveMode

This parameter is accessible from the Drive/Encoder view and allows the user to define what type of application the Unidrive M/Digitax HD will be configured for. Available selections from this list box are RFC-A and RFC-S. Based on the setting of this parameter, information on the motor tab will change to allow configuration of the different motor parameters, and the PTi210 module will automatically change the drive's operating mode on power up or warm start. This parameter cannot be changed while online.

Unidrive M/Digitax HD Encoder Supply Voltage
Drive.EncoderSupplyVoltage

This list box found on the Motor/Encoder View provides direct access to parameter Pr **03.036** of the drive frequency/speed configuration menu. To support many different applications, the Unidrive M/Digitax HD can provide several different supply voltages. Select from 5 V, 8 V, or 15 V to be provided from the drive hardware to the encoder. Be sure to set this to a value equal to or lower than the encoder supply specification. Setting this too high could potentially damage the encoder.

Unidrive M/Digitax HD Software Sub-Version
Drive.SoftwareSubVersion

The Unidrive M/Digitax HD software (or firmware) revisions are named using a ww.xx.yy.zz format displayed in parameter Pr **11.029** of the database. This parameter can be found on the Information tab on the Status view while Online with PowerTools Studio.

Unidrive M/Digitax HD Software Version
Drive.SoftwareVersion

The Unidrive M/Digitax HD software (or firmware) revisions are named using a ww.xx.yy.zz format displayed in parameter Pr **11.029** of the database. This parameter can be found on the Information tab on the Status view while Online with PowerTools Studio.

Unidrive M/Digitax HD Encoder Revolution Counter
DriveEncRevCount

This read-only parameter displays the number of whole revolutions that the motor encoder signal has moved since powered up (absolute) and can be found on the Online tabs on the Status and Position views. The units for this parameter are always revolutions since those are the units used by the drive and its option modules. The value for this parameter comes from parameter Pr **03.028** of the database.

Unidrive M/Digitax HD Fine Encoder Position
DriveEncRevFinePosition

This read-only parameter displays the number of fractions of a revolution that the master encoder signal has moved, and can be found on the Online tabs of the Status and Position views. The units for this parameter are always $1/(2^{32})$ of a revolution since those are the units used by the drive and its option modules. The value for this parameter comes from parameter Pr **03.030** of the database.

Unidrive M/Digitax HD Encoder Position
DriveEncRevPosition

This read-only parameter displays the number of fractions of a revolution that the master encoder signal has moved, and can be found on the Online tabs of the Status and Position views. The units for this parameter are always $1/(2^{16})$ of a revolution since those are the units used by the drive and its option modules. The value for this parameter comes from parameter Pr **03.029** of the database.

Unidrive M/Digitax HD Encoder Revolution Counter
MotorEncRevCount

This read-only parameter displays the number of whole revolutions that the motor position feedback encoder signal has moved since powered up (absolute) and can be found on the Online tabs on the Status and Position views. The units for this parameter are always revolutions since those are the units used by the drive and its option modules. This parameter is only available in Dual Loop Mode, where the motor position is determined by the position feedback source in conjunction with the drive encoder feedback acting as the velocity feedback signal.

Unidrive M/Digitax HD Fine Encoder Position
MotorEncRevFinePosition

This read-only parameter displays the number of fractions of a revolution that the motor position feedback encoder signal has moved, and can be found on the Online tabs of the Status and Position views. The units for this parameter are always $1/(2^{32})$ of a revolution since those are the units used by the drive and its option modules. This parameter is only available in Dual Loop Mode, where the motor position is determined by the position feedback source in conjunction with the drive encoder feedback acting as the velocity feedback signal.

Unidrive M/Digitax HD Encoder Position
MotorEncRevPosition

This read-only parameter displays the number of fractions of a revolution that the motor position encoder signal has moved, and can be found on the Online tabs of the Status and Position views. The units for this parameter are always $1/(2^{16})$ of a revolution since those are the units used by the drive and its option modules. This parameter is only available in Dual Loop Mode, where the motor position is determined by the position feedback source in conjunction with the drive encoder feedback acting as the velocity feedback signal.

Drive Status Bitmap

DriveStatus

The DriveStatus parameter is a decimal value that equates to the state of the Status Bits in the Unidrive M/Digitax HD. This is a read-only parameter and is read directly from parameter Pr **10.040** in the database. The individual bits in the bitmap are detailed below.

Bit 15 = Not Used	Not Used
Bit 14 = Parameter 10.015	Mains Loss
Bit 13 = Parameter 10.014	Reverse Direction Running
Bit 12 = Parameter 10.013	Reverse Direction Commanded
Bit 11 = Parameter 10.012	Braking Resistor Alarm
Bit 10 = Parameter 10.011	Braking IGBT Active
Bit 9 = Parameter 10.010	Regenerating
Bit 8 = Parameter 10.009	Drive Output Is At Current Limit
Bit 7 = Parameter 10.008	Load Reached
Bit 6 = Parameter 10.007	Above Set Speed
Bit 5 = Parameter 10.006	At Speed
Bit 4 = Parameter 10.005	Below Set Speed
Bit 3 = Parameter 10.004	Running At Or Below Min Speed
Bit 2 = Parameter 10.003	Zero Speed
Bit 1 = Parameter 10.002	Drive Active
Bit 0 = Parameter 10.001	Drive Healthy

Speed Feedback Selector

SpeedFeedbackSelector

This parameter is used to define the location of the Motor Feedback Source connection. This parameter allows direct access to parameter Pr **03.026** from the Unidrive M/Digitax HD frequency/speed configuration menu, and can be found on the Setup View. Valid selections from this list are P1 Drive, P2 Drive, P1 Slot 1, P2 Slot 1, P1 Slot 2, P2 Slot 2, P1 Slot 3, and P2 Slot 3.

DriveInput Status

DriveInput.#.In

The last three digital I/O points on the Unidrive M700/M701 are defined as digital inputs only (pins 27, 28, and 29).

The last two digital Input points on the Unidrive M702 are defined as digital inputs only (pins 7 and 8) up to date code 1710, when digital input 5 (pin 8) can be configured for either digital input or thermistor input.

The last two digital I/O points on the Digitax HD are defined as digital inputs only (pins 11 and 13). The ".In" at the end of the parameter is optional, so this signal is most often referred to simply as DriveInput.#. These inputs can be used in a user program, or on the Assignments view to activate different destination events.

These inputs are updated at the Trajectory Update Rate found on the Setup view in PowerTools Studio.

DriveInput Name

DriveInput.#.Name

Each digital input on the drive can be given a name. The name can be used in a user program to reference a specific input if desired. The name can be up to 12 alphanumeric characters, but must begin with a non-numeric character.

DriveIO Direction

DriveIO.#.Direction

The first three digital I/O points on the Unidrive M700/M701 (pins 24, 25, and 26) are configured as Inputs or Outputs using PowerTools Studio software. The DriveIO.#.Direction parameter is used to configure whether the I/O point acts as a digital input or digital output. Valid entries for this parameter are selInput or selOutput. This parameter is modified automatically when the user changes the settings on the Drive I/O Setup view.

DriveIO Status
DriveIO.#.In

The first three digital I/O points on the Unidrive M700/M701 (pins 24, 25, and 26) are configured as Inputs or Outputs using PowerTools Studio software. If the I/O points have been configured as digital inputs, then the DriveIO.#.In signal can be used in a user program to check the status of inputs, or on the Assignments view to activate different signals. If assigned to a destination on the assignments view, when the digital input activates, the destination that it is assigned to will also activate.

NOTE

This parameter will appear on the Assignments view under the Inputs group of Sources even if the I/O point has been configured as a digital output.

DriveIO Name
DriveIO.#.Name

Each digital I/O point on the Unidrive M/Digitax HD can be given a name. The name can be used in a user program to reference a specific I/O point. The name can be up to 12 alphanumeric characters, but must begin with a non-numeric character.

DriveIO State
DriveIO.#.Out

The first three digital I/O points on the Unidrive M700/M701 (pins 24, 25, and 26) are configured as Inputs or Outputs using PowerTools Studio software. If the I/O points have been configured as digital outputs, then the DriveIO.#.Out signal can be used in a user program to activate an output, or on the Assignments view to activate when a specified Source event activates. If assigned to a source on the assignments view, when the source event activates, the output that it is assigned to will turn on.

NOTE

This parameter will appear on the Assignments view under the Outputs group of Destinations even if the I/O point has been configured as a digital input.

Drive Power Up Time – Hrs.Min
PowerUpTimeHoursMinutes

The Drive Power Up Time is the time elapsed since power has been cycled to the drive, two parameters are used to provide the total power up time, PowerUpTimeHoursMinutes and PowerUpTimeYearsDays. This parameter gives the power up time hours and minutes values. These values must be used in combination to find the actual time. This parameter can be found on the Errors view while Online with PowerTools Studio.

Total Power Up Time = PowerUpTimeYearsDays + PowerUpTimeHoursMinutes

Drive Power Up Time – Yrs.Days
PowerUpTimeYearsDays

The Drive Power Up Time is the time elapsed since power has been cycled to the drive, two parameters are used to provide the total power up time, PowerUpTimeHoursMinutes and PowerUpTimeYearsDays. This parameter gives the power up time years and days values. These values must be used in combination to find the actual time. This parameter can be found on the Errors view while Online with PowerTools Studio.

Total Power Up Time = PowerUpTimeYearsDays + PowerUpTimeHoursMinutes

DriveRelay Name
DriveRelay.#.Name

The relay output on the drive can be given a name. The name can be used in a user program to reference a specific output if desired. The name can be up to 12 alphanumeric characters, but must begin with a non-numeric character.

DriveRelay State
DriveRelay.#.Out

The Unidrive M700/M701/M702 each have one relay output that can be used to control devices that require more current than a digital output. The ".Out" at the end of the parameter name is optional, so this parameter is often referred to simply as DriveRelay.#. This parameter can be used in a user program to activate the output (i.e. DriveRelay.7 = On), or can be assigned to a Source on the Assignment view.

Drive Run Time – Hrs.Min
RunTimeHoursMinutes

The drive Run Time is the Total Time that the drive has been powered up with the Bridge Enabled since last reset by the factory two parameters are used to provide the total run time, RunTimeHoursMinutes and RunTimeYearsDays. This parameter gives the run time

hours and minutes values. These values must be used in combination to find the actual total run time. This parameter can be found on the Errors view while Online with PowerTools Studio.

Total Run Time = RunTimeYearsDays + RunTimeHoursMinutes

Drive Run Time – Yrs.Days
RunTimeYearsDays

The drive Run Time is the Total Time that the drive has been powered up and the Bridge Enabled since last reset by the factory two parameters are used to provide the total run time, RunTimeHoursMinutes and RunTimeYearsDays. This parameter gives the run time years and days values. These values must be used in combination to find the actual time. This parameter can be found on the Errors view while Online with PowerTools Studio.

Total Run Time = RunTimeYearsDays + RunTimeHoursMinutes

Drive Stack Temperature 1
StackTemperature1

This read-only parameter is read from the Unidrive M/Digitax HD parameter database (Pr **07.004**). StackTemperature1 can be used in a user program to monitor the temperature of the drive.

Start Up
StartUp

This source can be used to trigger an event to occur on startup (when the PTi210 module powers up or is rebooted). This source is typically used to initiate a program or to initiate a home so that a machine will automatically home on power up or reboot. StartUp will activate when the PTi210 module has powered up and no faults are active. Startup may take as long as five seconds to activate. Depending on what the Startup source is assigned to, the drive may need to be enabled to perform the function. If the drive is not enabled, the startup source cannot initiate programs or motion. The source will remain active until the PTi210 module is powered down.

Stop
Stop

Activate this destination to stop all motion and programs. If Stop is activated when a Jog, Index, Home or Program is in progress, they will decelerate to zero speed at the Stop Decel ramp. When Stop is active, all Jog, Home, Index and Program initiate destinations will be ignored. When it is deactivated, all level sensitive and active input functions (Jog.0.PlusActivate, Jog.0.MinusActivate, etc.) will become operational. For example, if the Jog.PlusActivate input function is active when the Stop input function is deactivated, the Jog.Plus motion will initiate using the acceleration found in the Jog.0.Accel parameter. This is level sensitive.

Stop Deceleration
StopDecel

Deceleration rate used when the Stop destination is activated.

Drive Switching Frequency
SwitchingFrequency

This parameter defines the switching frequency for the Unidrive M/Digitax HD (under normal conditions). Higher values for switching frequency will eliminate audible high-frequency noise, but can require derating of system performance in some cases. This parameter is found on the Setup view and can be changed within a user program. Valid selections for this parameter are 2 kHz, 3 kHz, 4 kHz, 6 kHz, 8 kHz, 12 kHz, and 16 kHz.

PTi210 Module Total Power Up Time
TotalPowerUpTime

Total Power Up Time is the total elapsed time that the PTi210 module has been powered up (since reset by the factory). The units for the parameter are Hours with a resolution of 0.1 Hours. This parameter is stored in the PTi210 module, and is not reset if the module is switched to another drive.

Travel Limit Deceleration
TravelLimitDecel

This parameter defines the ramp used to decelerate the motor to a stop when any travel limit is activated.

Travel Limit Disable
TravelLimitDisable

TravelLimitDisable can be used from the Assignments screen, or through a user program. It can be used to temporarily disable the travel limit fault capability of the PTi210 module. When TravelLimitDisable is activated, the PTi210 module travel limits (hardware or software) are no longer valid. If disabled using a program, the travel limits will automatically be re-enabled when the program ends, if

they haven't already been enabled. This feature is typically used when a machine must use one of its limit switches as a home switch. The user disables the travel limits, then homes to the limit switch, and then re-enables the travel limit.

Travel Limit Minus Activate
TravelLimitMinusActivate

This destination is used to activate the travel limit minus fault. It should be assigned to the travel limit minus sensor. When it is activated the drive will decelerate to a stop using the deceleration rate defined in the TravelLimitDecel parameter. This is level sensitive.

Travel Limit Minus Active
TravelLimitMinusActive

This source is active when the TravelLimitMinusActivate is active.

Travel Limit Plus Activate
TravelLimitPlusActivate

This destination is used to activate the travel limit plus fault. It should be assigned to the travel limit plus sensor. When it is activated the drive will decelerate to a stop using the deceleration rate defined in the TravelLimitDecel parameter. This is level sensitive.

Travel Limit Plus Active
TravelLimitPlusActive

This source is active when the TravelLimitPlusActivate is active.

Variable Decimal
Var.Var#.Decimal

This parameter specifies the number of decimal placed of resolution that this particular user variable will use. Minimum value is 0 (default), and the maximum number of decimal places in 6 (0.000000). When assigning the value of a User Variable to different parameters, make sure that the parameter and the User Variable have the same number of decimal places.

Variable Value
Var.Var#.Value

This parameter specifies the current value of a user variable. In a program, the ".Value" portion of the parameter name can be left off. For example:

Var.Var0.Value = 12345 is the same as Var.Var0 = 12345

When assigning the value of a User Variable to different parameters, make sure that the parameter and the User Variable have the same number of decimal places.

Velocity Command
VelCommand

The Velocity Command is the velocity that the PTi210 module is commanding the motor to run at. This command is generated by the drive velocity control loop. It is displayed in user units.

Velocity Feedback
VelFeedback

This is the feedback (or actual) velocity. It will always return the actual motor velocity, even in synchronized applications in which the master axis is halted during a move.

Velocity Feedforward Enable
VelocityFeedforwardEnable

The Velocity Feedforward applies the calculated velocity command directly to the Unidrive M/Digitax HD velocity loop. Enabling the Velocity Feedforward will generally yield faster velocity response (reaches programmed velocity much faster), but can introduce some velocity overshoot.

NOTE

If your application requires jogging motion, the Enable Velocity Feedforward should always be active.

Velocity Loop Bandwidth
VelocityLoopBandwidth

The Velocity Loop Bandwidth parameter is the theoretical bandwidth of the velocity controller (how fast the system can respond to change in velocity command). The value of this parameter depends heavily on the correct values for the motor data, particularly the Motor Rotor Inertia and the Motor Current Constant (Ke). Units for this parameter are Hz.

Time Scale
VelocityUnits.TimeScale

Velocity time scale can be set to user units per second or user units per minute, used for all real-time velocities throughout the PowerTools Studio software.

Acceleration
VirtualMaster.Accel

This parameter is the acceleration rate, in user units, that the virtual master will use to accelerate. This parameter is used when in either jog or indexing mode.

Deceleration
VirtualMaster.Decel

This parameter is the deceleration rate, in user units, that the virtual master will use to decelerate. This parameter is used when in either jog or indexing mode.

Scaling
VirtualMaster.CharacteristicDistance

The denominator (bottom value of the scaling fraction) is the VirtualMaster.CharacteristicDistance and is used with VirtualMaster.CharacteristicLength to create the virtual master conversion ratio. Converting the user units distance into virtual counts.

Scaling
VirtualMaster.CharacteristicLength

The numerator (top value of the scaling fraction) is the VirtualMaster.CharacteristicLength and is used with VirtualMaster.CharacteristicDistance to create the virtual master conversion ratio scaling. Converting the user units distance into virtual counts.

Scaling = Virtual MasterCharacteristicLength / VirtualMaster.CharacteristicDistance

Counts
VirtualMaster.Counts

This read only parameter is the number of virtual counts transmitted. It is zeroed on power up only.

Distance
VirtualMaster.Dist

This parameter is the incremental distance virtual master will move, in user units, if the virtual master is initiated as an index.

FeedRate Decel/Accel
VituralMasterFeedRateDecelerationTime

This parameter specifies the ramp used when velocity changes due to a change in the FeedRate Override value. The units of FeedRate Decel/Accel are seconds/100 %. Therefore, the user must specify the amount of time (in seconds) to accelerate or decelerate 100 % of programmed feedrate.

FeedRate Override
VirtualMaster.FeedRateOverride

This parameter is used to scale the Virtual Master counts. It can be described as “scaling in real time”. The default setting of 100 % will allow all counts to occur in real time. A setting of 50 % will scale time so that all counts are half as fast as they are at 100 %. A setting of 200 % will scale time so that all count run twice as fast as they would at 100 %. Feed Rate Override is always active, and this parameter may be modified via Modbus, Ethernet, or in a program.

Position Counts
VirtualMaster.PosnCmdInCounts

This parameter is the number of virtual counts transmitted. Write to this parameter to zero its value. It is only used as a user display for virtual master debugging.

Position Command
VirtualMaster.PosnCommand

This is the virtual position which uses VirtualMaster.PosnCmdInCounts and the conversion ratio to display the position in user units. This is zeroed by clearing VirtualMaster.PosnCmdInCounts.

Velocity
VirtualMaster.Vel

This parameter is the maximum virtual velocity that will be attained by the virtual master. This parameter is in user units.

Enable Virtual Master
VirtualMaster.VirtualMasterEnable

Enable Virtual Master check box by default is clear. Select the check box to enable virtual master (VirtualMaster.VirtualMasterEnable = ON).

13 Drive Parameters Used by the PTi210 Module

13.1 Description

The Unidrive M/Digitax HD is configured using a database of parameters. The parameters are grouped according to functionality. Each function group is given a Menu#. Each parameter in the drive is accessed using a Menu Number and Parameter Number in the following format:

Menu Number.Parameter Number (or **mm.ppp**)

An example of this is Menu Number 5, Parameter Number 7 is accessed using 5.007.

In order to configure the drive to operate as desired, the **mm.ppp** parameters must be set to a specific value. To make configuration of the base drive parameters behind the scenes.

Since some advanced users may wish to utilize various drive parameters, it is important to know which parameters are being used by the PTi210 module. and how they are being used, The chart in section 13.2 details the drive parameters used by the PTi210 module.

13.2 Chart

The parameters outlined in bold are used in some way by the PTi210 module. Each parameter outlined in bold has a note associated with it to further describe how the PTi210 module uses that particular parameter. The chart is followed by a detailed description of each of the notes.

Menu 1 Speed Ref.	Menu 2 Ramps	Menu 3 Feedback Setup	Menu 4 Current Control	Menu 5 Motor Control	Menu 6 Sequencer	Menu 7 Analog I/O	Menu 8 Digital I/O	Menu 9 Logic / Mot. Pot.	Menu 10 Status / Trips	Menu 11 General Drive Setup
							8.001[2]		10.001[2]	
	2.002[1]						8.002[2]		10.002[2]	
			4.003[2]				8.003[2]			
			4.004[2]			7.004[2]	8.004[2]			
			4.005[5]	5.005[2]			8.005[2]			
1.006[3]			4.006[5]				8.006[2]			
			4.007[3]	5.007[3]						
				5.008[3]						
									10.009[2]	
		3.010[3]								
		3.011[3]	4.011[5]	5.011[3]		7.011[3]	8.011[1]			
		3.012[3]	4.012[3]	5.012[3]			8.012[1]		10.012[2]	
			4.013[3]				8.013[1]			
1.014[5]			4.014[3]				8.014[1]			
1.015[5]			4.015[3]		6.015[5]	7.015[3]	8.015[1]			
			4.016[1]							
		3.017[1]		5.017[3]			8.017[1]			
				5.018[3]			8.018[1]			
			4.019[2]			7.019[3]				
					6.020[2]	7.020[3]	8.020[2]		10.020[2]	
1.021[5]					6.021[2]		8.021[1]		10.021[2]	
		3.022[4]	4.022[1]		6.022[2]	7.022[3]	8.022[1]		10.022[2]	
		3.023[5]			6.023[2]	7.023[3]	8.023[1]		10.023[2]	
				5.024[3]			8.024[1]		10.024[2]	
		3.025[3]					8.025[1]		10.025[2]	
		3.026[3]					8.026[1]		10.026[2]	
		3.027[2]					8.027[1]		10.027[2]	
		3.028[2]			6.028[1]		8.028[1]		10.028[2]	
		3.029[2]					8.029[1]		10.029[2]	11.029[2]
		3.030[2]								
		3.031[5]					8.031[3]			11.031[1]
							8.032[3]			
		3.033[3]					8.033[3]			
		3.034[3]			6.034[5]					11.034[2]
		3.035[3]		5.035[1]						
		3.036[3]								
		3.037[3]								
		3.038[3]							10.038[5]	
		3.039[1]							10.039[2]	
		3.040[1]							10.040[2]	
		3.041[3]							10.041[2]	
									10.042[2]	
									10.043[2]	
									10.044[2]	
									10.045[2]	
									10.046[2]	
									10.047[2]	
									10.048[2]	
									10.049[2]	
									10.050[2]	
									10.051[2]	

Notes:

1. Set at start-up or initialization
   These parameters are set to a specific value depending on the configuration every time the system is powered up. This parameter must not be written to or the system may not function correctly.
2. Mapped to Read Only variable
3. Mapped to Read / Write variable
4. Continuously updated parameter
   This parameter is written to on every trajectory update of the PTi210 module. Do not attempt to write to this parameter.
5. This parameters is accessed by the PTi210 module firmware
   This parameter should not be written to or the system may not function correctly.
6. Reserved for use by the PTi210 module only if an SI-I/O module is fitted.
   If more than one SI-I/O module is fitted, this parameter relates to the SI-I/O module in the lowest slot position.
7. Reserved for use by the PTi210 module only if a second SI-I/O module is fitted.
   This parameter relates to the SI-I/O module in the highest slot position.

Menu 12 Threshold / Var Selector	Menu 13 Position Control	Menu 14 User PID Controller	Menu 15 Option Module 1	Menu 16 Option Module 2	Menu 17 Option Module 3	Menu 18 User Application 1	Menu 19 User Application 2	Menu 20 User Application 3	Menu 21 2nd Motor Setup
		14.001[3]	Slot Option Setup Menu	Slot Option Setup Menu	Slot Option Setup Menu	18.001[5]			
		14.002[3]							
		14.003[3]							
		14.004[3]							
		14.005[3]							
		14.006[3]							
		14.007[3]							
		14.008[3]							
		14.009[3]							
		14.010[3]							
		14.011[3]							
		14.012[3]							
		14.013[3]							
		14.014[3]							
		14.015[3]							
		14.016[3]							
		14.017[3]							
		14.018[2]							
		14.019[2]							
		14.020[2]							
		14.021[2]							
		14.022[2]							
						18.031[6]	19.031[7]		
						18.032[6]	19.032[7]		
						18.033[6]	19.033[7]		
						18.034[6]	19.034[7]		
						18.035[6]	19.035[7]		
						18.036[6]	19.036[7]		
						18.037[6]	19.037[7]		
						18.038[6]	19.038[7]		
						18.039[5]		20.039[6]	
						18.040[5]		20.040[6]	
						18.041[5]			
						18.042[5]			

Notes:

1. Set at start-up or initialization
   These parameters are set to a specific value depending on the configuration every time the system is powered up. This parameter must not be written to or the system may not function correctly.
2. Mapped to Read Only variable
3. Mapped to Read / Write variable
4. Continuously updated parameter
   This parameter is written to on every trajectory update of the PTi210 module. Do not attempt to write to this parameter.
5. This parameters is accessed by the PTi210 module firmware
   This parameter should not be written to or the system may not function correctly.
6. Reserved for use by the PTi210 module only if an SI-I/O module is fitted.
   If more than one SI-I/O module is fitted, this parameter relates to the SI-I/O module in the lowest slot position.
7. Reserved for use by the PTi210 module only if a second SI-I/O module is fitted.
   This parameter relates to the SI-I/O module in the highest slot position.

2.

13.3 PTi210 Module Setup Parameters

Several parameters associated with the PTi210 option module can be found in either Menu 15, 16, or 17. Each of menus 15, 16, and 17 refer to one of the available slots into which the PTi210 option module can be fitted.

The following PTi210 parameters are available from the keypad on the Unidrive M700/M701/M702 or a remote keypad. The slot # that the module is plugged into directly determines which menu these parameters are found under.

Slot 1 – Menu 15

Slot 2 – Menu 16

Slot 3 – Menu 17 (Not available on Digitax HD)

1M.001 – Option ID Code

When no option module is fitted in the relevant slot, this parameter is zero. When a module is fitted in the relevant slot, this parameter displays the identification code of the module as shown in the System Integration Module Compatibility Chart in section 2.4.

When the drive parameter database is saved by the user, the option code of the currently fitted module is saved in EEPROM. If the drive is subsequently powered-up with a different module fitted, or a module is fitted where one wasn't previously fitted, the drive will have a SlotX Different trip. If no module is fitted where one was previously fitted the drive will have a SlotX Not Fitted trip. To clear the SlotX Different trip or SlotX Not Fitted trip, follow the “Clearing the SlotX Different Trip or SlotX Not Fitted Trip” instructions in the diagnostics section of this manual.

1M.002 – Option Software Version

This parameter indicates the PTi210 module firmware revision that is currently stored in the module. It is possible for the user to flash upgrade the firmware using PowerTools Studio software.

The format for this parameter is WW.XX.YY.ZZ, where WW is the major revision, XX is the minor revision, YY is the sub-version number, and YY is the build number.

1M.003 – Hardware Version

This parameter shows the PTi210 module hardware version. This parameter is read-only and is written to on initialization. The format for the hardware version is XX.YY, where XX is the major hardware version and YY is the minor hardware version.

1M.004 – Serial Number LS

The PTi210 module serial number is a 10-digit decimal number loaded during manufacture and cannot be changed. This parameter contains the last 8 digits of the serial number (see also Pr **1M.005**).

1M.005 – Serial Number MS

The PTi210 module serial number is a 10-digit decimal number loaded during manufacture and cannot be changed. This parameter contains the first 2 digits of the serial number (see also Pr **1M.004**).

1M.006 – Module Status

This parameter shows the PTi210 module status according to the table below. This parameter is read-only.

Value	Text	Description
-2	Bootldr - Update	The bootloader is performing a flash update
-1	Bootldr - Idle	The bootloader is idle
0	Initializing	The module is initializing
1	OK	Module is initialized with no errors
2	Config	A configuration error has been detected
3	Error	An error has occurred preventing the module from running correctly

1M.007 – Module Reset

If this parameter is set to a value of 1 (On) then the PTi210 module will be reset and will re-initialize using the existing option module parameter configuration. When the PTi210 module is re-initialized, this parameter will revert to Off (0).

1M.008 – Module Default

If this parameter is set to a value of 1 (On) and the PTi210 module is reset (Pr **1M.007** = On), the default configuration parameter values will be loaded into the PTi210 module. When the parameters have been loaded, this parameter will revert to Off (0).

1M.013 – ModuleOutput.1.Status

This parameter shows the status of digital Output #1 on the PTi210 module. A value of 1 indicates that the Output is active. A value of 0 indicates that the Output is inactive. This equates Pin # 4 on the PTi210 module I/O Connector.

1M.014 – ModuleOutput.2.Status

This parameter shows the status of digital Output #2 on the PTi210 module. A value of 1 indicates that the Output is active. A value of 0 indicates that the Output is inactive. This equates Pin # 5 on the PTi210 module I/O Connector.

1M.017 – ModuleInput.1.Status

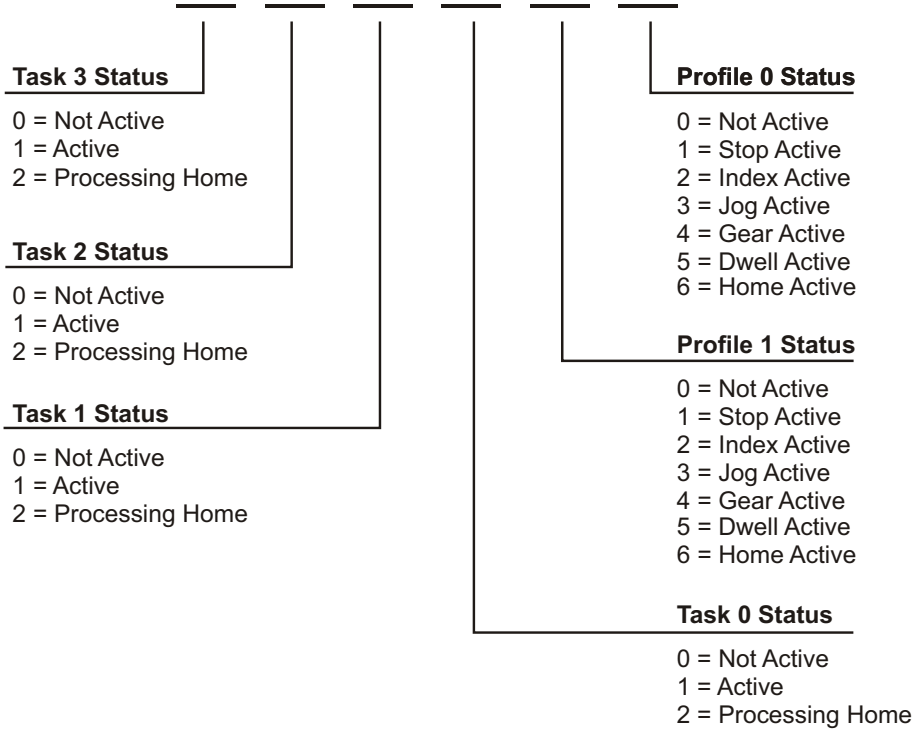
This parameter shows the status of digital Input #1 on the PTi210 module. A value of 1 indicates that the Input is active. A value of 0 indicates that the Input is inactive. This equates Pin # 1 on the PTi210 module I/O Connector.

1M.018 – ModuleInput.2.Status
This parameter shows the status of digital Input #2 on the PTi210 module. A value of 1 indicates that the Input is active. A value of 0 indicates that the Input is inactive. This equates Pin # 2 on the PTi210 module I/O Connector.

1M.019 – ModuleInput.3.Status
This parameter shows the status of digital Input #3 on the PTi210 module. A value of 1 indicates that the Input is active. A value of 0 indicates that the Input is inactive. This equates Pin # 3 on the PTi210 module I/O Connector.

1M.048 – System Status
The System Status parameter is used to indicate the status of each of the User Program Tasks and Motion Profiles. This parameter can give the user some idea of what the PTi210 module is commanding without being online using PowerTools Studio.

There are six individual digits that indicate the status of different processes. This parameter will display a six digit number that can be used in conjunction with the graphic below to determine the status of each process. Leading zeros in the value are not displayed.



Examples:

1M.048 = 11023 would signify that user programs are currently running on Task 2 and Task 1, and that a Index is running on Profile 1 while a Jog is running on Profile 0.

1M.048 = 104 would signify that a user program is running on Task 0 and Gearing motion is running on Profile 0.

1M.049 – Virtual Master Source
If a virtual master is enabled in the configuration, this parameter displays the simulated position count. The range of this parameter is -32768 to 32767.

1M.050 – Option Module Error Status
The error status is provided so that only one option module error trip is required for each option module slot. If an error occurs, the error code is written to this parameter, and the drive may produce a Slotx Error trip (where x is the slot number). A value of zero indicates that the module has not detected an error. A non-zero value indicates that an error has been detected (See Errors and Error Codes in Diagnostics section of this manual). When the drive is reset, this parameter is cleared for the relevant option module.

1M.051 – Registry Revision Number
This parameter displays the minimum registry interface version supported by the current PTi210 module firmware.

If an existing application configuration file is loaded into PowerTools Studio, then PowerTools Studio will convert the application configuration to this version.

1M.055 – Reserved
This parameter is reserved for use by the PTi210 module.

1M.056 – Reserved
This parameter is reserved for use by the PTi210 module.

Parameters not listed above are not used by the PTi210 module.

14 Diagnostics

There are many different tools available to the user to help diagnose problems or errors in the Unidrive M/Digitax HD and the PTi210 module. The most common tools used are the Error Codes, Analog Outputs, and PowerTools Studio utilities such as the Watch Window, Fault View, Status Bar, and Online View Tabs. Any or all of these can be used to figure out why an application may not be running properly.

The following section gives detailed information about each of the tools listed above.

14.1 Errors and Error Codes

The Unidrive M/Digitax HD and PTi210 module fault handling system is made of a series of Trips, Trip Codes, Errors, and Error Codes. Following is a definition of each.

A Trip is an action that happens in the Unidrive M/Digitax HD that causes the drive bridge to be inhibited therefore stopping all motion. When the drive inhibits, the drive does not have control of the motor/load.

An Error is an action that happens in the PTi210 module or Unidrive M/Digitax HD that may or may not cause the drive to Trip. Most of the Errors that occur in the PTi210 module will cause a Trip on the drive.

Following is a description of the Errors Handling System used by the PTi210 module.

When the PTi210 module has an error (i.e. Program - Divide By Zero Error), it sends a signal to the drive that the error occurred. The drive then recognizes the error in the module by causing the drive to trip. The specific trip that will be displayed on the drive depends on the slot number that the PTi210 module is located in. The drive trip will be a Slot# Error trip where # indicates the slot # that the module error occurred in.

The Unidrive M/Digitax HD Trip Log will be updated with the Slot# Error trip. To find out what the specific error in the module was, the user needs to query parameter Pr **10.070** and refer to the relevant option module user guide for information on the trip code.

Following is a list of Errors that can occur in the PTi210 module along with Error Codes, potential causes, and reset methods for each fault.

Error Code (1M.050)	Cause Trip?	Error	Possible Reason	Possible Solution
40	Y	Drive Parameter Access Error	An error occurred during a parameter access	Confirm the parameter is available
41	Y	Drive Parameter Access Error – Parameter Doesn't Exist	The drive menu parameter does not exist	Check the drive menu parameter is correct
42	Y	Drive Parameter Access Error – Parameter is Read Only	The drive menu parameter being written to is a Read Only parameter	Avoid writing to Read Only parameters. Use the Drive Menu Watch in Power Tools Studio or the relevant drive user guide to verify accessibility
43	Y	Drive Parameter Access Error – Parameter is Write Only	The drive menu parameter being read from is a Write Only parameter	Avoid reading from Write Only parameters. Use the Drive Menu Watch in Power Tools Studio or the relevant drive user guide to verify accessibility
44	Y	Drive Parameter Access Error - Written Value Out of Range	The drive menu parameter value being written is outside the accepted range	Ensure the value being written is within the accepted drive menu parameter range. Refer to the relevant drive user guide to verify the parameter range
73	Y	Drive Database Setup Error	An error has occurred setting up the drive database	Replace module if problem persists
74	Y	Module Overheat Error	PTi210 module temperature has exceeded 88 °C	Check for adequate air-flow, or reduce drive switching frequency, or reduce motion performance (increase accel/decel rates) or lengthen dwell times. If problem persists replace module
101	Y	Invalid Configuration Error	The configuration is invalid	Check the application configuration for possible problems (scaling factors, motion timings, etc.)
102	Y	NVM Invalid Error	PTi210 module has detected an invalid NVM configuration	Power cycle the drive and replace the module if problem persists
103	Y	Power Up Test Failure Error	PTi210 module Power Up Test failed	Power cycle the drive and replace the module if problem persists
104	Y	Following Error	Amount of following error exceeded the following error limit set in PowerTools Studio.	Increase Following Error Limit and/or Velocity Loop Bandwidth and/or Position Loop bandwidth in PowerTools Studio configuration. Lower the acceleration and/or deceleration ramp values. Make sure that programmed velocity is within maximum operating speed of the given motor.
105	N	Travel Limit Plus	Hardware Travel Limit Plus switch has activated, or Software Travel Limit Plus position has been exceeded	Verify motion profiles/programs to make sure that motion is not configured to exceed desired travel positions

Error Code (1M.050)	Cause Trip?	Error	Possible Reason	Possible Solution
106	N	Travel Limit Minus	Hardware Travel Limit Minus switch has activated, or Software Travel Limit Minus position has been exceeded	Verify motion profiles/programs to make sure that motion is not configured to exceed desired travel positions
107	Y	No Program Error	PTi210 module has no configuration loaded in it	Download a configuration to the PTi210 module using PowerTools Studio software
108	Y	Motion Trajectory Error	Maximum allowable position change within one control loop update has been exceeded	If using the "Using Capture.#" instruction after an Index.#.Initiate instruction in your user program, make sure that the captured data is recent enough so that the motor can actually achieve the necessary acceleration
109	Y	Trajectory Update Overrun Error	Control Loop processing time has taken longer than the user selected Trajectory Update Rate	Select a longer Trajectory Update Rate in the PowerTools Studio configuration. A longer Trajectory Update Rate gives the control loop more time to process. Be sure not to enable any PLSs, Captures, or Queues that are not being used
120	Y	Program Error - Break Wire Input Event Invalid	An output event is wired to an invalid input event	Check Assignments for invalid connections. If this issue persists contact the supplier
121	Y	Program Error - Break Wire To Non Input Event	Attempt to wire an output event to an output event	Check Assignments for invalid connections. If this issue persists contact the supplier
122	Y	Program Error - Break Wire To Non Output Event	An attempt to wire an input event to a non output event	Check Assignments for invalid connections. If this issue persists contact the supplier
124	Y	Program Error - Call Stack Overflow	Too many "Call Program" instructions have been processed without returning to original "calling" program	Do not nest more than four "Call Program" operations. To avoid this, return to the original calling programs before calling another program. (See Call Program instruction explanation in this manual for more information).
125	Y	Program Error - Create Table Child Not Found	Specified CAM table child could not be created	Re-download the PowerTools Studio configuration file. If the problem persists, please contact the supplier
127	Y	Flash Error	Loading from Flash Memory has failed	Re-download original PowerTools Studio configuration file. If the problem persists, please replace module
129	Y	Program Error - Illegal Command	The User program has encountered an illegal command	Check user programs and re-download original PowerTools Studio configuration file. If problem persists, please replace module
130	Y	Program Error - Index Doesn't Exist	The User program has encountered an invalid Index	Check user programs and re-download original PowerTools Studio configuration file. If the problem persists, please replace module
132	Y	Program Error - Invalid Profile	The User program has encountered an invalid profile	Check motion profiles and user programs for invalid (non existent) profile use
133	Y	Program Error - Invalid Wait For OpCode	User program has encountered an invalid Wait instruction	Check user programs
134	Y	Program Error - Jog Doesn't Exist	The specified Jog motion does not exist	Check user programs and Jog motion settings
135	Y	Program Error - Math Addition Overflow	Math addition operation in user program has resulted in an overflow of the resultant parameter	Verify that the sum of all the operands in addition formulas will not result in a value in the following range: -231 < SUM < 231-1
136	Y	Program Error - Math Divide By Zero	Formula in user program causes a divide by zero error	Make sure that the denominator in all division formulas is not equal to zero
137	Y	Program Error - Math Divide Operand Too Large	Formula in user program uses an operand which is outside the allowed values	Check user programs for invalid operand values
138	Y	Program Error - Math Multiplication Normalization Failed	Normalization of multiplication parameters in user program has failed	Check user programs for invalid operand values
139	Y	Program Error - Math Multiplication Operand Too Large	Multiplication formula uses an operand which is too large	Check user programs for invalid operand values
140	Y	Program Error - Overflow	The user program has encountered an overflow error	Check user programs
141	Y	Program Error - Math Subtraction Overflow	Subtraction formula results in an overflow error	Check user programs
142	Y	Program Error - Math Stack Overflow	User Program math process stack has overflown	Check user programs
143	Y	File Corruption Error - Not Enough Create Data Area	During module startup the module was expecting more data than was found	Verify that there is not a discrepancy between the PTi210 module and PowerTools Studio configuration registry revisions. If the issue persists contact the supplier

Error Code (1M.050)	Cause Trip?	Error	Possible Reason	Possible Solution
144	Y	File Corruption Error - Not Enough Write Data Area	During module startup there was not enough write data in the downloaded configuration	Verify that there is not a discrepancy between the PTi210 module and PowerTools Studio configuration registry revisions. If the issue persists contact the supplier
145	Y	File Corruption Error - Object Not Found	During module startup the downloaded configuration referenced an object that does not exist in the module	Verify that there is not a discrepancy between the PTi210 module and PowerTools Studio configuration registry revisions. If the issue persists contact the supplier
147	Y	Program Error - Flash Memory Size Exceeded	User configuration exceeds available flash memory size	Eliminate any unused Indexes or Programs and re-download the PowerTools Studio configuration
148	Y	Program Error - RAM Memory Size Exceeded	User configuration exceeds available RAM memory size	Eliminate any unused Indexes or Programs and re-download the PowerTools Studio configuration
153	Y	Program Error - Program Doesn't Exist	User program does not exist	Check configuration and user programs
154	Y	Program Error - Program Executing	User program already running	Check configuration and user programs
155	Y	Program Error - Read Table Child Not Found	Specified CAM table child could not be read	Check user programs
156	Y	Program Error - Too Many Wait For Instructions	No more than nine arguments in Wait for Instruction	Limit # of arguments and re-download
157	Y	Program Error - Too Many Wires To Input Event	Too many triggers assigned to a single input	Check assignments for nodes with too many connections. (32 is the limit for a single node)
158	Y	Program Error - Too Much Create Data	During startup the configuration contained too much create data for a target module object	Verify that there is not a discrepancy between the PTi210 module and PowerTools Studio configuration registry revisions. If the issue persists contact the supplier
159	Y	Program Error - Too Much Write Data	During startup the configuration contained too much write data for a target module onject	Verify that there is not a discrepancy between the PTi210 module and PowerTools Studio configuration registry revisions. If the issue persists contact the supplier
161	Y	Program Error - Wire Input Event Invalid	An output event is wired to an invalid input event	Check assignments and user programs for invalid connections. If this issue persists please contact the supplier.
162	Y	Program Error - Wire To Non Input Event	Attempt to wire an output event to an output event	Check assignments and user programs for invalid connections. If this issue persists please contact the supplier.
163	Y	Program Error - Wire To Non Output Event	Attempt to wire an input event to a non output event	Check assignments and user programs for invalid connections. If this issue persists please contact the supplier.
165	Y	Program Error - Write Table Child Not Found	The object being written to does not exist	Check user programs for invalid module variable access
166	Y	Program Error - PTi210 Parameter Write Out of Range	Value written to PTi210 parameter in user program is out of range	Check user program for parameter range
171	Y	Invalid Slot 1 Selection	Option module selected for Slot 1 in PowerTools Studio file does not match actual module type fitted	Switch module located in Slot 1 to match module type selected in PowerTools Studio file. Alternatively, update the PowerTools Studio configuration to match the module type actually fitted, and then redownload the configuration
172	Y	Invalid Slot 2 Selection	Option module selected for Slot 2 in PowerTools Studio file does not match actual module type fitted	Switch module located in Slot 2 to match module type selected in PowerTools Studio file. Alternatively, update the PowerTools Studio configuration to match the module type actually fitted, and then redownload the configuration
173	Y	Invalid Slot 3 Selection	Option module selected for Slot 3 in PowerTools Studio file does not match actual module type fitted	Switch module located in Slot 3 to match module type selected in PowerTools Studio file. Alternatively, update the PowerTools Studio configuration to match the module type actually fitted, and then redownload the configuration
175	Y	Output Overload	Digital output current demand is too high	Check digital output current does not exceed 20 mA in total
176	Y	CAM initiate	Invalid CAM configuration, this error can occur if the CAM table doesn't exist	Check references to which CAM tables are being initialized in programs
177	Y	CAM invalid interpolation	Interpolation type is invalid	Check configuration and replace module if problem persists
178	Y	CAM invalid master	Master Delta of zero in the CAM table	Check the CAM table for any master coefficients with a delta of zero

Error Code (1M.050)	Cause Trip?	Error	Possible Reason	Possible Solution
179	Y	CAM profile limited	The final and starting velocities between two time based index CAM tables is too great	Check the final and initial velocities of the CAMs match
180	Y	Cycle program timeout	Cyclic program timed out	Reduce the number of loops or run time of the Cyclic program
181	Y	Cycle program too big	The Cyclic program contains too many instructions	Reduce the number of instructions in the cyclic program
182	Y	Real time program timeout	Real time program exceeded 50 % of the trajectory update rate	Reduce the code size/complexity of the program.
183	Y	Real time program too big	The real time program contains too many instructions	Reduce the number of instructions in the real time program
184	Y	Programmed CAM element does not exist	The specified CAM segment within a table does not exist	Check references to specific CAM segments in programs
185	Y	Programmed CAM not writable	An attempt to modify the CAMs table has been made whilst the CAM is not writable	If the CAM does need to be writable, ensure it is enabled in the CAM table view. If it is not supposed to be writable, check usage in programs
186	Y	Programmed CAM table does not exist	The specified CAM table does not exist.	Check references to which CAM tables are being initialised in programs
187	Y	CAM forward backward error	Invalid CAM table jump	The specified CAM table to jump to does not exist
201	Y	Invalid RTMoE Easy Mode Configuration	The onboard Ethernet interface has reported an invalid link configuration for the RTMoE Easy Mode link	Check the RTMoE Easy Mode link configuration
208	Y	Database init failed	Failed to initialise the database	Replace module if problem persists
210	Y	Memory Allocation	Failed to allocate memory	Replace module if problem persists
221	Y	Missing Factory Setting	The module serial number has not been set	Return to supplier
226	Y	Processor Exception	The processor is in the exception state	Return to supplier if problem persists
227	Y	Task Starvation	The system task does not have sufficient time to run	Return to supplier if problem persists
228	Y	Runtime except	A runtime exception has occurred	Return to supplier if problem persists

14.2 Analog Outputs

The Unidrive M700/M701 has two Analog Outputs that can be configured to represent any base drive parameter (00.000 - 59.999). The default configuration for the two drive analog outputs are Speed Feedback (Pr **03.002**) and Torque Producing Current / Iq (Pr **04.002**). Using this method will automatically update the analog output value.

To configure the drive Analog Output to represent a PTi210 module parameter, the Analog Output Source of the drive (Pr **07.019** or Pr **07.022**) must be set to a 32-bit drive application parameter. Once the Analog Output Source has been set to a 32-bit parameter, the PTi210 user program must be coded to write the value of the desired PTi210 parameter to the 32-bit menu parameter. The drive analog outputs are scaled -100 % to +100 % if the source parameter is bipolar, or 0 % to 100 % if the source parameter is unipolar. So in the example below, Pr **20.021** range is -2147483648 to 2147483647 equating to an output voltage of between approximately -10 V to +10 V.

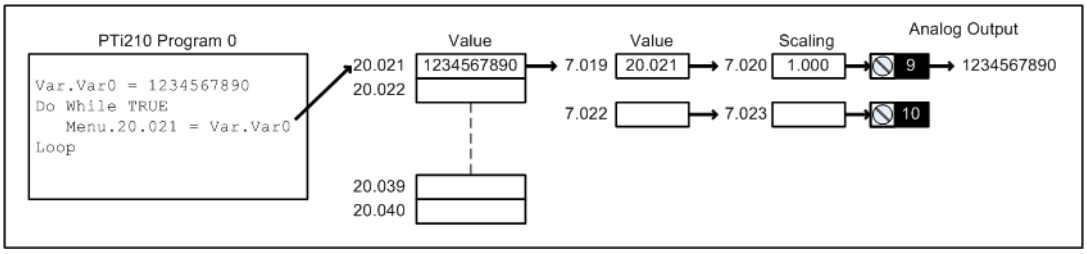


Figure 14-1: Analog Output Example

For more details on the Analog Output of the drive, refer to the Unidrive M *User Guide* or Unidrive M *Parameter Reference Guide*.

14.3 PowerTools Studio

14.3.1 Watch Window

PowerTools Studio contains a diagnostic utility called the Watch Window. The Watch Window can be used while PowerTools Studio is online with the PTi210 module. The Watch Window allows the user to monitor the status of all of their desired system parameters in one location. An example of the Watch Window is found in Figure 14-2.

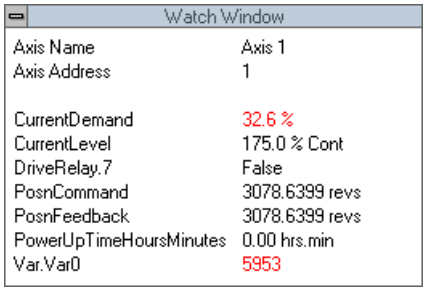


Figure 14-2: Watch Window Example

To setup the Watch Window, select **Tools > Watch Window** from the PowerTools Studio menu bar. If not online with the module, the Watch Window will be a unavailable on the menu. Upon selecting Watch Window, the Select Drive Parameters window will open.

The Select Drive Parameters window, as seen in Figure 14-3, allows the user to specify which parameters are to be viewed in the Watch Window. To select a parameter for the Watch Window, simply click once on the parameter and it will be added to the Watch Window. The parameters already in the Watch Window will be highlighted in the Select Drive Parameters window.

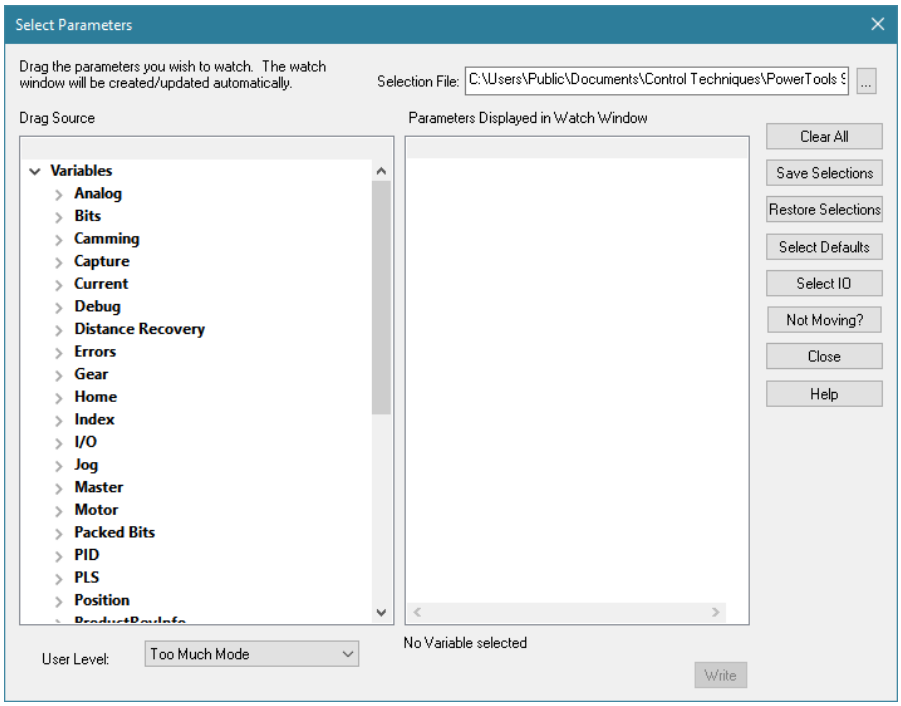


Figure 14-3: Select Drive Parameters Window

Once a parameter is added to the Watch Window, its current value or state is constantly monitored. If a parameter in the window changes value or state, it will change to a red color.

It will remain red until it hasn't changed for a period of 4 seconds. After 4 seconds, the parameter will turn back to black in color. This allows the user to see what has changed recently without looking directly at every parameter.

The following are descriptions of the buttons and controls on the Select Drive Parameters window:

Clear All - By clicking on the ***Clear All*** button, all of the parameters in the Watch Window will be erased.

Save Selections - By clicking on ***Save Selections***, the user can save the specific parameters that have been added to the Watch Window are saved in the file specified in the Selection File box. Once the selections have been saved, the ***Restore Selections*** button can be used to monitor all the same parameters the next time the user opens the Watch Window. Therefore, if there is a list of helpful diagnostic parameters the user wishes to see when online, those specific parameters can be saved and recalled in the Watch Window at any time. The default Watch Window file name is "ptiwatch.wch" but can be changed using the Selection File box.

Restore Selections - By clicking on the ***Restore Selections*** button, the Watch Window will be filled with the list of parameters that were last saved in the file specified in the ***Selection File*** box.

Select Defaults - The ***Select Defaults*** button adds the most commonly used parameters to the Watch Window.

Select I/O - The ***Select I/O*** button will add the Drive and Module digital inputs and outputs to the Watch Window.

Not Moving -The ***Not Moving?*** button will load the watch window with a list of predefined parameters (shown in the following table) that will give an indication why the motor is not moving.

Parameters	Expected Value to get Motion	Comments
DriveEnableStatus	True	Drive must be enabled.
Error.Errored	False	Motion stops on a fault.
DriveOK	True	The drive status must be ok to enable motion.
Stop	False	Motion is stopped if stop is active.
MotionStop	False	Motion is stopped on motion stop.
SoftDriveDisable	False	Motion is prevented by SoftDriveDisable.
AbsolutePosnValid	True	Absolute indexes can not executed until the Absolute position is defined.
InPosn	True	If using "In Position" the last index is not considered complete until this status is met. So motion will stop if you can not achieve in position. While running this is expected to be false as it is only true at the end of a move and prior to starting a new move.
Feedhold	False	Feed Hold stops motion.
FeedRateOverride	100 %	Feed rate scales the speed of the Motion. 100 % scaling indicates one to one scaling.
FeedRateDeactivate	False / True	Activate to disable FeedRate. If feedrate is 100 % it does not matter.
SoftwareTravelLimitMinusActive	False	Motion stops when a travel limit is active.
SoftwareTravelLimitPlusActive	False	Motion stops when a travel limit is active.
CurrentLimitActive	False	Current limits may effect motor motion as it reduces motor generated torque.
Profile.0.MotionStop	False	Each profile has its own Stop. To get motion out of this profile this parameter must be false.
Profile.0.FeedHold	False	Each profile has its own Feedhold. To get motion out of this profile this parameter must be false.
Profile.1.MotionStop	False	Each profile has its own Stop. To get motion out of this profile this parameter must be false.
Profile.1.FeedHold	False	Each profile has its own Feedhold. To get motion out of this profile this parameter must be false.

Close - The ***Close*** button will close the Select Drive Parameters window, while the Watch Window will remain open.

Help - The ***Help*** button will give associated help on the Watch Window setup.

User Level - The User Level setting is a filter for the parameters that are seen in the Select Drive Parameters list. If set to Easy, the parameters used in most basic applications will be seen while the more advanced parameters are hidden. If User Level is set to Detailed, the parameters used in more advanced applications will be added to the list. If set to Too Much, then all parameters available in the system will be seen in the list. This allows the user to select the User Level they are most comfortable with to avoid confusion. If a parameter has been selected and the User Level is changed, then the selected parameter will remain selected.

Within the Select Parameters window the user can change the value of parameters in the drive's RAM memory. Click on a parameter in the Parameters Displayed in Watch Window pane. Just below the pane the parameter that was selected will be shown along with either a check box, text box or list box. The type of box is depended on the parameter selected. Enter the new parameter value and press the ***Write*** button.The data should change in the watch window to the new value.

14.3.2 Errors View

The Errors View in PowerTools Studio helps the user diagnose problems by displaying any active errors or trips. If online when an error activates, an error pop-up window will appear that shows the active error. There are three buttons on the pop-up window that allow the user to clear (reset) an error trip, ignore it, or display the online Help information, both the Reset and Ignore buttons will close and remove the pop-up window. An example of the pop-up window is shown in Figure 14-4.

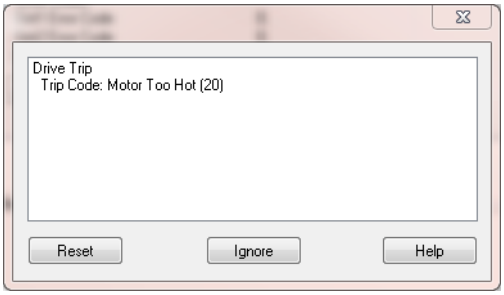


Figure 14-4: PTi210 Error Pop-Up Window

The Errors View also contains a Trip Log that lists the 10 most recent drive trips. Trip 0 is listed as the most recent fault with a Trip Time in Years.Days and Hours.Minutes format. Along with the Trip Code, the Trip Log will also store the time before the most recent trip that each trip occurred.

14.3.3 Status Bar

The Status Bar in PowerTools Studio gives a wide variety of information to the user such as Drive Status, Motion Status, Position and Velocity Feedback, and Communications Status. The Status Bar is found along the bottom of the PowerTools Studio screen. Figure 14-5 shows the Status Bar location

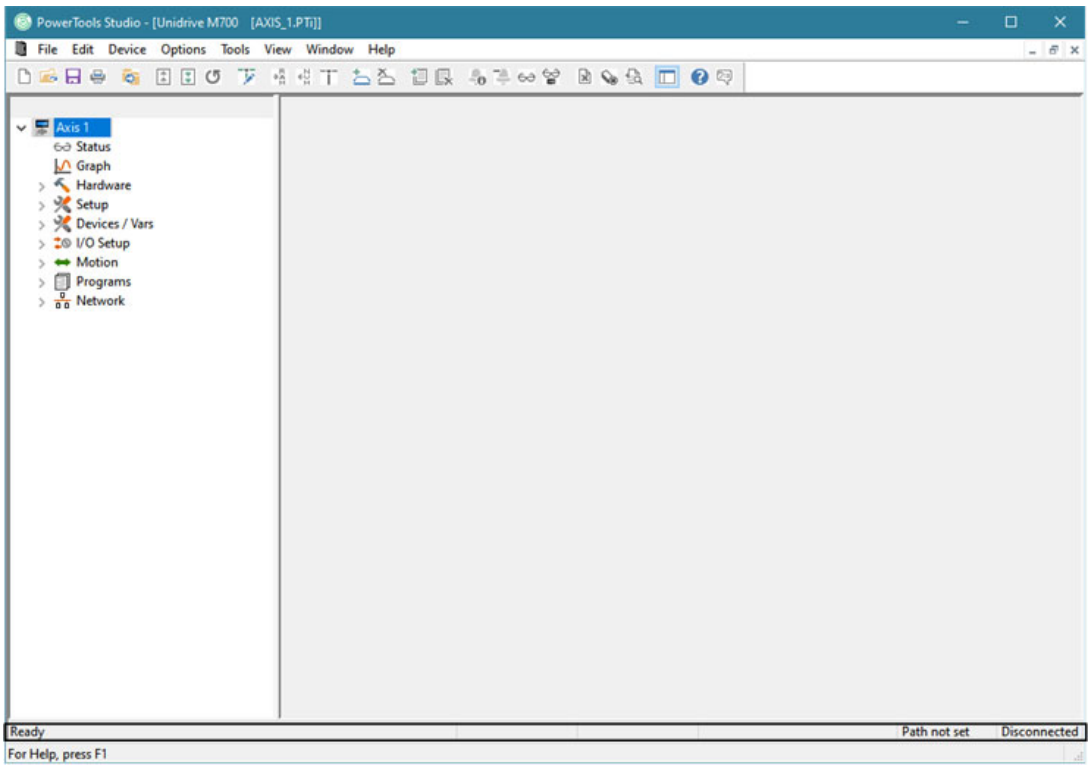


Figure 14-5: PowerTools Studio Status Bar Location

The Status Bar is broken into 6 segments to clearly display the necessary information.

Figure 14-6 shows a more detailed view of the Status Bar.

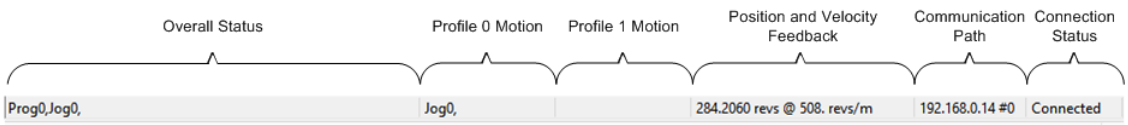


Figure 14-6: Status Bar Description

The first segment of the status bar gives an overall drive status. It will indicate whether the drive is enabled or not, and will also indicate which programs and motion profiles are active.

The second segment shows what motion is currently running on Profile 0.

The third segment shows what motion is currently running on Profile 1.

The fourth segment shows the current Feedback Position and Feedback Velocity in user units.

The fifth segment shows the current communication path. Modbus communications will display the Com port and Modbus Node Address. Ethernet communications displays the IP Address and Modbus Node Address.

The last segment shows the communication status. If PowerTools Studio is online with the system, it will show Connected. If offline, the segment will show Disconnected. If Disconnected, none of the other segments of the status bar will be functional.

14.3.4 Where Am I?

The Where Am I utility found on the PowerTools Studio Toolbar allows the user to find what line of a user program is currently being processed. To activate the Where Am I, simply click on the button in the toolbar. A blue arrow will point to the line of the program that is being processed.

For more information on the Where Am I, refer to section 3 of this manual (Menu Bar or Toolbar descriptions).

14.3.5 Online View Tabs

Many of the Views in PowerTools Studio have feedback information displayed on an Online Tab. The Online Tab is only visible when PowerTools Studio is communicating with the PTi210 module.

The main Status Online Tab is found on the Status View, and it displays feedback and diagnostic parameters for the overall system. Figure 14-7 is an example of the Status View - Online Tab.

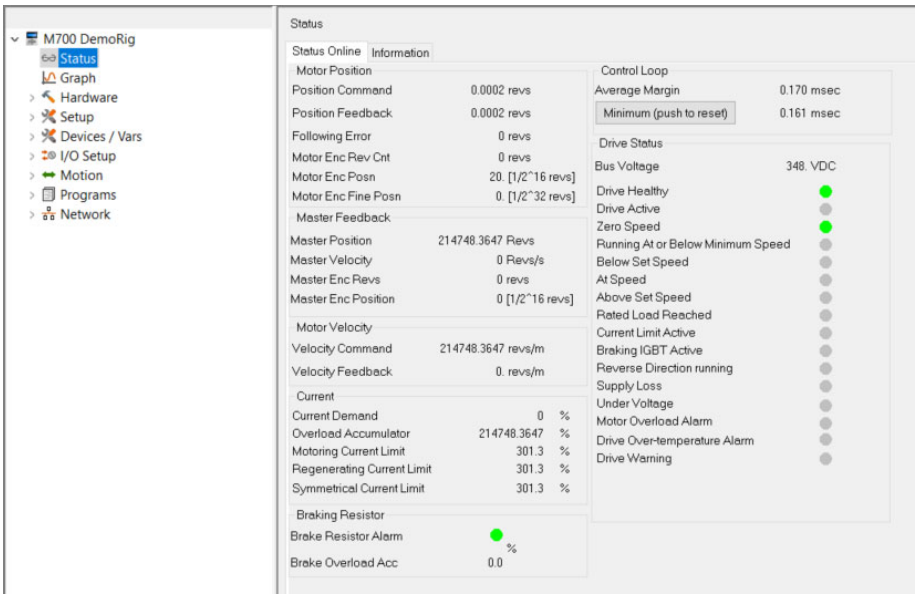


Figure 14-7: Status View-Online Tab

Many of the views in PowerTools Studio have Online tab that contain parameters related to that view when online with the drive.

The Status View also provides an Information tab which shows the configuration file location, drive firmware version, PTi210 firmware version and registry interface revision number as shown in Figure 14-8.

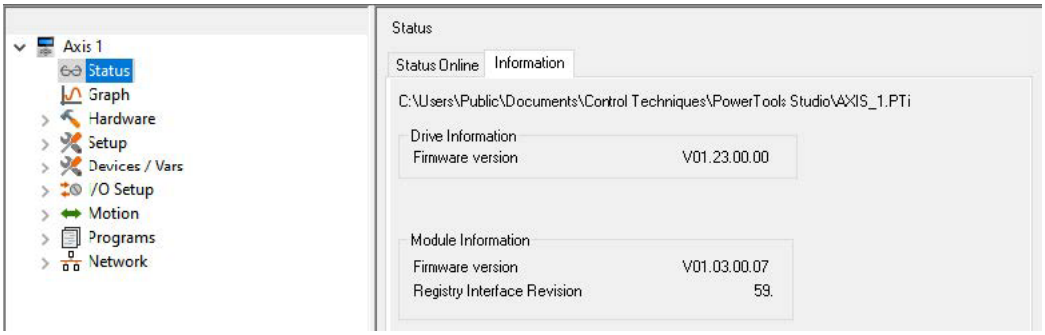


Figure 14-8: Status View – Information Tab

14.4 Clearing SlotX Difference trip after installing the PTi210 module (using the keypad)

After installing a PTi210 module for the first time, the Unidrive M/Digitax HD will display a SlotX Difference (if the drive is fitted with a local or remote keypad) trip (where X is the slot number that the PTi210 module is in). This trip occurs because the drive detects the module type in the slot has changed. This trip will occur on each successive power up until the drive parameter database has been saved. To save the drive parameter database, follow the instructions below.

Saving the Drive Parameters

When changing a parameter in Menu 0, the new value is saved automatically when exiting from the edit mode by pressing the Enter button  to return to parameter view mode from parameter edit mode.

If parameters have been changed in the advanced menus, then the change will not be saved automatically. A save function must be carried out.

The procedure for saving the parameters is as follows:

1. Enter the value 1001 in **mm.000** where **mm** is any menu number (not including the PLC registers menus 70-79 in any SI-Applications/MCi2x0 option module).
2. Press the red Reset button. 

14.5 Clearing the PTi210 module memory using the keypad

There are two ways to default the PTi210 module using the keypad, both will clear the PTi210 memory and delete any user programs. The preferred way is to use the parameters **1M.008** (*Default Module*) and **1M.007** (*Reset Module*).

If, for any reason this is not possible due to the PTi210 module parameters being inaccessible, the following procedure can be used:

1. Navigate to Pr **18.001** on the keypad.
2. Press the Enter button  to enter parameter edit mode.
3. Enter the value 19237.
4. Press the Enter button  again to return to parameter view mode.
5. Remove AC power (and 24V DC backup power if used) from the drive, and wait for 'Under Voltage' message to clear on the keypad display.
6. Restore power to the drive.

After performing the clear memory sequence, the PTi210 module will cause a SlotX Error trip (where X indicates the slot number that the PTi210 module is fitted in). The Error Code for the PTi210 module will be “107 – No Program Error” indicating that the module has no configuration stored in it. Download a new configuration using PowerTools Studio to clear the trip.

15 Glossary

µs
Microsecond, which is 0.000001 seconds.

A
Amps.

Amplifier
Servo Drive.

ARMS
Amps Root Mean Squared (RMS).

Axis
The full system to control in a single motor shaft. A single PTi210 module with Unidrive M/Digitax HD Drive can denote an axis.

AWG
American Wire Gauge.

Baud Rate
The number of binary bits transmitted per second on a serial communications link such as RS-232. (1 character is usually 10 bits.)

Check Box
In a dialog box, a check box is a small box that the user can turn “On” or “Off” with the mouse. When “On” it displays an X in a square; when “Off” the square is blank. Unlike option (radio) buttons, check boxes do not affect each other; any check box can be “On” or “Off” independently of all the others.

Complex Motion
A string of multiple motion commands and logical instructions that form a repeatable operation. For the PTi210 module, the configuration file defines complex motion by setups, functional assignments and programs.

Compound Motion
The combination of indexes in a row in which the deceleration ramp of the first index goes to the velocity of the secondary index. The first index must be initiated within a program (Index.#.CompoundInitiate).

Configuration
The user-created application. It can be saved as a disk file or downloaded to configure the PTi210 module. It includes all the user-defined setup, assignments and programs.

CRC
Cyclical Redundancy Check, the data transfer error checking mechanism.

Destination
A function (i.e., Stop, Preset) that may be assigned to an input line. In PTi210, the input function is connected to the action through click and drag operations in PowerTools Studio Software on the Assignment View.

Dialog Box
A dialog box is a window that appears in order to collect information from the user. When the user has filled in the necessary information, the dialog box disappears.

DIN Rail
Deutsche Industrie Norm Rail

DLL
In Microsoft® Windows®, a Dynamic Link Library contains a library of machine-language procedures that can be linked to programs as needed at run time.

Downloading
The transfer of a complete set of parameters from an FM to a drive.

Drive
Servo drive or amplifier.

EEPROM
An EEPROM chip is an Electrically Erasable Programmable Read-Only Memory; that is, its contents can be both recorded and erased by electrical signals, but they do not go blank when power is removed.

EMC
Electromagnetic Compatibility. The relative immunity of a drive to the effects of electromagnetic fields.

EMI - Electro-Magnetic Interference
EMI is noise which, when coupled into sensitive electronic circuits, may cause problems.

Firmware
The term firmware refers to software (i.e., computer programs) that are stored in some fixed form, such as read-only memory (ROM).

Flash
Another type of EEPROM.

Flash File
In the PTi210 module, this file loads the firmware into the drive and function module. Flash files can field upgrade the firmware.

FM
Function Module - device which is attached to the front of the drive to provide additional functionality.

Global Where Am I
PowerTools Studio feature that indicates which line of which user program is executing.

Home Routine
The home provides motion in applications in which the axis must precisely align with some part of a machine.

Hysteresis
For a system with an analog input, the output tends to maintain its current value until the input level changes past the point that set the current output value. The difference in response of a system to an increasing input signal versus a decreasing input signal.

I/O
Input/Output. The reception and transmission of information between control devices. In modern control systems, I/O has two distinct forms: switches, relays, etc., which are in either an on or off state, or analog signals that are continuous in nature generally depicting values for speed, temperature, flow, etc.

Index
An index is a complete motion sequence (defined motion profile) that moves the motor a specific incremental distance or to an absolute position.

Inertia
The property of an object to resist changes in rotary velocity unless acted upon by an outside force. Higher inertia objects require larger torque to accelerate and decelerate. Inertia is dependent upon the mass and shape of the object.

Input Function
See destination. A function (i.e., Stop, Preset) that may be assigned to an input line. In PowerTools Studio, the input function is connected to the action through click and drag operations in PowerTools Studio Software on the Assignment View.

Input Line
The terminals of a device or circuit to which energy is applied.

Jog
A jog produces rotation of the motor at controlled velocities in a positive or negative direction.

Least Significant Bit
The bit in a binary number that is the least important or having the least weight.

LED
Light Emitting Diode used on the front display of drives and function modules.

List Box
In a dialog box, a list box is an area in which the user can choose among a list of items, such as files, directories, printers or the like.

mA
Milliamp, which is 1/1000th of an Ampere.

MB
Mega-byte.

MODBUS
Communication Protocol by Modicon. The drives follows the Modbus specification outlined in the Modicon Modbus Protocol Reference Manual, PI-MBNS-300 Revision G, November 1994.

Module
PTi210 Module

Most Significant Bit
The bit in a binary number that is the most important or that has the most weight.

Motion Made Easy
Motion Made Easy refers to the package consisting of the PTi210 motion controller option module and the associated programming software utility PowerTools Studio.

ms
Millisecond, which is 1/1000th of a second.

NVM
Non-Volatile Memory. NVM stores specifically defined variables as the variables dynamically change. It is used to store changes through a power loss.

NTC
Negative Temperature Resistor

Option Button
See Radio Button.

Opto-isolated
A method of sending a signal from one piece of equipment to another without the usual requirement of common ground potentials. The signal is transmitted optically with a light source (usually a Light Emitting Diode) and a light sensor (usually a photosensitive transistor). These optical components provide electrical isolation.

Output Function
See source. The terminals at which energy is taken from a circuit or device.

Output Line
The actual transistor or relay controlled output signal.

Parameters
User read only or read/write parameters that indicate and control the drive operation. These variables generally hold numerical data defined in the Setup Views.

PC
Personal Computer.

PE
Protective Earth.

PID
Proportional-Integral-Derivative. An acronym that describes the compensation structure that can be used in many closed-loop systems.

PLC
Programmable Logic Controller. Also known as a programmable controller, these devices are used for machine control and sequencing.

PowerTools Studio
PowerTools Studio is a Windows® based software to interface with the PTi210 module.

Radio Button
Also known as the Option Button. In a dialog box, radio buttons are small circles only one of which can be chosen at a time. The chosen button is black and the others are white. Choosing any button with the mouse causes all the other buttons in the set to be cleared.

RAM
RAM is an acronym for Random-Access Memory, which is a memory device whereby any location in memory can be found, on average, as quickly as any other location. Commonly refers to Read-Write memory, as opposed to Read-Only Memory (ROM, EPROM, EEPROM, Flash). RAM is considered volatile, because its contents are lost during a power loss.

RMS
Root Mean Squared. For an intermittent duty cycle application, the RMS is equal to the value of steady state current which would produce the equivalent heating over a long period of time.

ROM
ROM is an acronym for Read-Only Memory. A ROM contains computer instructions that do not need to be changed, such as permanent parts of the operating system.

RPM
Revolutions Per Minute.

Serial Port
A digital data communications port configured with a minimum number of signal lines. This is achieved by passing binary information signals as a time series of 1's and Ø's on a single line.

Source
The terminals at which energy is taken from a circuit or device.

Travel Limit
The distance that is limited by either a travel limit switch or the software.

Torque
The moment of force, a measure of its tendency to produce torsion and rotation about an axis.

Uploading
The transfer of a complete set of parameters from a drive to an FM.

User Units
Ability of program to allow user to specify which type of units will measure and specify motion and time.

Vac
Volts, Alternating Current.

Variable
A labeled value that encompasses numeric boolean, input function, and output functions.

VDC
Volts, Direct Current.

Velocity
The rate of change in position in a given direction during a certain time interval.

View
Portion of screen within frame.

Windows®, Microsoft®
Microsoft Windows is an operating system that provides a graphical user interface, extended memory and multi-tasking. The screen is divided into windows and the user uses a mouse to start programs and make menu choices.

16 Index

A

B

C

D

E

F

G

H

I

J

M

N

O

P

Q

R

S

T

U

V

Nidec

All for dreams

Connect with us

www.controltechniques.com

www.kbelectronics.com

Nidec Control Techniques Limited. Registered Office: The Gro, Newtown, Powys SY16 3BE.

Registered in England and Wales. Company Reg. No. 01236886.

CONTROL TECHNIQUES

KB

0478-0616-07