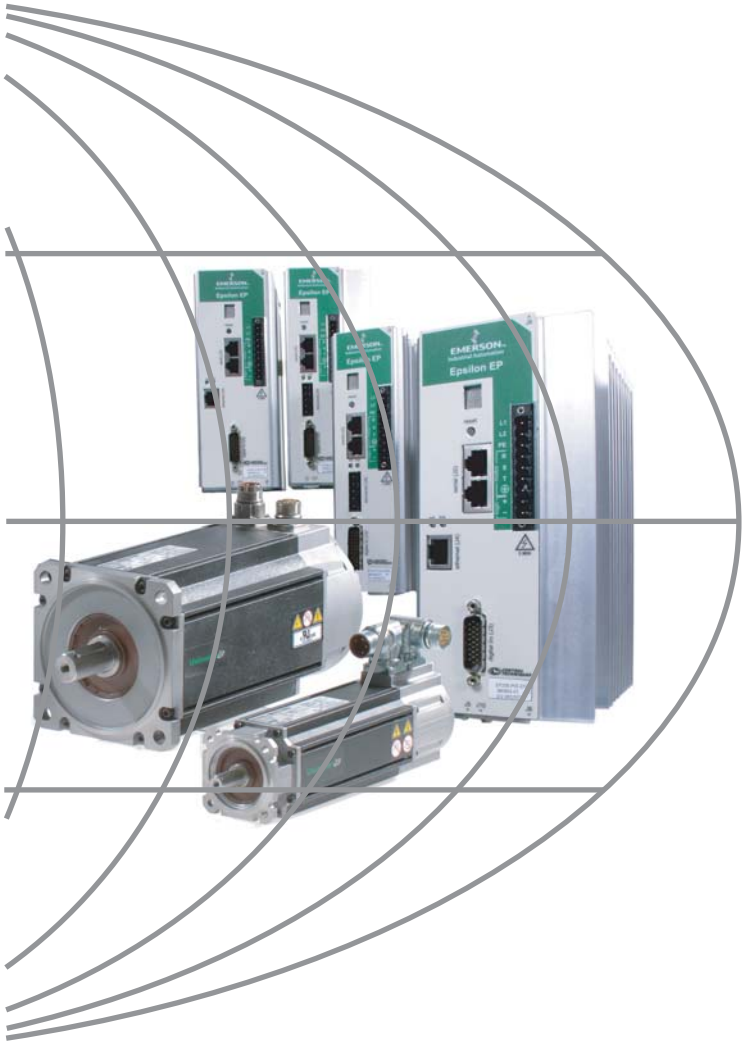
EMERSON
Industrial Automation



Reference Manual

Epsilon EP-P Servo Drive

"MOTION MADE EASY" ®

Part Number: 400518-04
Revision: A3
Date: April 30, 2010

DRIVE CENTRE
Industrial Automation Systems Integrators

57 Galaxy Blvd., Units 1 & 2, Toronto, ON M9W 5P1
TEL: (416) 231-6767
www.drivecentre.ca

CONTROL TECHNIQUES

Epsilon EP-P Drive Reference Manual

P/N 400518-04
Revision: A3
Date: April 30, 2010

Part Number: 400518-04

Revision: A3

Date: April 2010

Printed in United States of America

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.

Customer Support

Control Techniques Americas LLC
12005 Technology Drive
Eden Prairie, Minnesota 55344-3620
U.S.A.

Telephone: (952) 995-8000 or (800) 893-2321

It is Control Techniques' goal to ensure your greatest possible satisfaction with the operation of our products. We are dedicated to providing fast, friendly, and accurate assistance. That is why we offer you so many ways to get the support you need. Whether it's by phone, fax or modem, you can access Control Techniques support information 24 hours a day, seven days a week. Our wide range of services include:

FAX (952) 995-8099

You can FAX questions and comments to Control Techniques. Just send a FAX to the number listed above.

Website and Email www.emersonct.com

Website: www.emersonct.com

Email: info@emersonct.com

If you have Internet capabilities, you also have access to technical support using our website. The website includes technical notes, frequently asked questions, release notes and other technical documentation. This direct technical support connection lets you request assistance and exchange software files electronically.

Technical Support (952) 995-8033 or (800) 893-2321

Email: service@emersonct.com

Control Techniques' "Motion Made Easy" products are backed by a team of professionals who will service your installation. Our technical support center in Eden Prairie, Minnesota is ready to help you solve those occasional problems over the telephone. Our technical support center is available 24 hours a day for emergency service to help speed any problem solving. Also, all hardware replacement parts, if needed, are available through our customer service organization.

When you call, please be at your computer, with your documentation easily available, and be prepared to provide the following information:

- Product version number, found by choosing About from the **Help** menu
- The type of controller or product you are using
- Exact wording of any messages that appear on your screen
- What you were doing when the problem occurred
- How you tried to solve the problem

Need on-site help? Control Techniques provides service, in most cases, the next day. Just call Control Techniques' technical support center when on-site service or maintenance is required.

Training Services (952) 995-8000 or (800) 893-2321

Email: training@emersonct.com

Control Techniques maintains a highly trained staff of instructors to familiarize customers with Control Techniques' "Motion Made Easy" products and their applications. A number of courses are offered, many of which can be taught in your plant upon request.

Application Engineering (952) 995-8000 or (800) 893-2321

Email: service@emersonct.com

An experienced staff of factory application engineers provides complete customer support for tough or complex applications. Our engineers offer you a broad base of experience and knowledge of electronic motion control applications.

Customer Service (Sales) (952) 995-8000 or (800) 893-2321

Email: customer.service@emersonct.com

Authorized Control Techniques distributors may place orders directly with our Customer Service department. Contact the Customer Service department at this number for the distributor nearest you.

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 options are printed in bold type: the **File** menu.

Dialog box names begin with uppercase letters: the Axis Limits dialog box.

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

Button names are in 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.

Reference Materials

The following related reference and installation manuals may be useful with your particular system.

- *Epsilon EP Installation Manual* (P/N 400518-01)
- *Epsilon EP Connectivity Manual* (P/N 400518-05)

Safety Information

Safety Precautions

This product is intended for professional incorporation into a complete system by qualified persons. If you install the product incorrectly, it may present a safety hazard. The product and system may use high voltages and currents, carry a high level of stored electrical energy, or are used to control mechanical equipment that can cause injury.

You must give close attention to the electrical installation and system design to avoid hazards either in normal operation or in the event of equipment malfunction. System design, installation, commissioning and maintenance must be carried out by personnel who have the necessary training and experience. Read and follow this safety information and this instruction manual carefully.

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.

Enclosure

This product is intended to be mounted in an enclosure that prevents access except by qualified persons and that prevents the ingress of contamination. This product is designed for use in an environment classified as pollution degree 2 in accordance with IEC664-1. This means that only dry, non-conducting contamination is acceptable.

Setup, Commissioning and Maintenance

It is essential that you give careful consideration to changes to drive settings. Depending on the application, a change could have an impact on safety. You must take appropriate precautions against inadvertent changes or tampering. Restoring default parameters in certain applications may cause unpredictable or hazardous operation.

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.

Identification of Safety Information

Safety related information through out this manual is identified with the following markings.

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

CAUTION	“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 an Epsilon EP-P drive.

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.

Supply isolation device
The AC supply or high voltage DC 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 6 minutes for Epsilon EP206/209/216 and 3 minutes for Epsilon EP202/204 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) - High Leakage Current
The drive must be grounded by a conductor sufficient to carry all possible fault current in the event of a fault. This equipment has high earth leakage current. You must comply with local safety regulations with respect to minimum size and special installation requirements on the protective earth conductor for high leakage current equipment. The ground connections shown in the manual must be followed.

Fuses
Fuses or over-current protection must be provided at the input in accordance with the instructions in the manual.
The drive alone does not provide branch circuit protection. Branch circuit protection must be provided in accordance with the National Electrical Code and any additional local codes.

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. External control circuits identified as PELV circuits do not need this isolation when they are completely within a zone of equipotential bonding, generally within a single enclosure or group of enclosures bonded together.

1 Introduction

1.1 Epsilon EP Drive

The Epsilon EP drive is a stand-alone, fully digital brushless servo drive designed and built to reliably provide high performance and flexibility without sacrificing ease of use.

The use of State-Space algorithms make tuning very simple and forgiving. The drives are designed to operate with up to a 10:1 inertia mismatch right out of the box. Higher (50:1 and more) inertial mismatches are possible with two simple parameter settings.

The Epsilon EP drive can be quickly configured to many applications in less than 5 minutes with Emerson Control Techniques - PowerTools Pro software on a PC running Windows® 98, 2000, XP (32-bit) or Vista (32-bit).

Complete diagnostics are provided for quick troubleshooting. A diagnostic display on the front of the drive informs the user of the operational or fault status. The last 10 faults are stored in non-volatile memory along with a time stamp for easy recall.

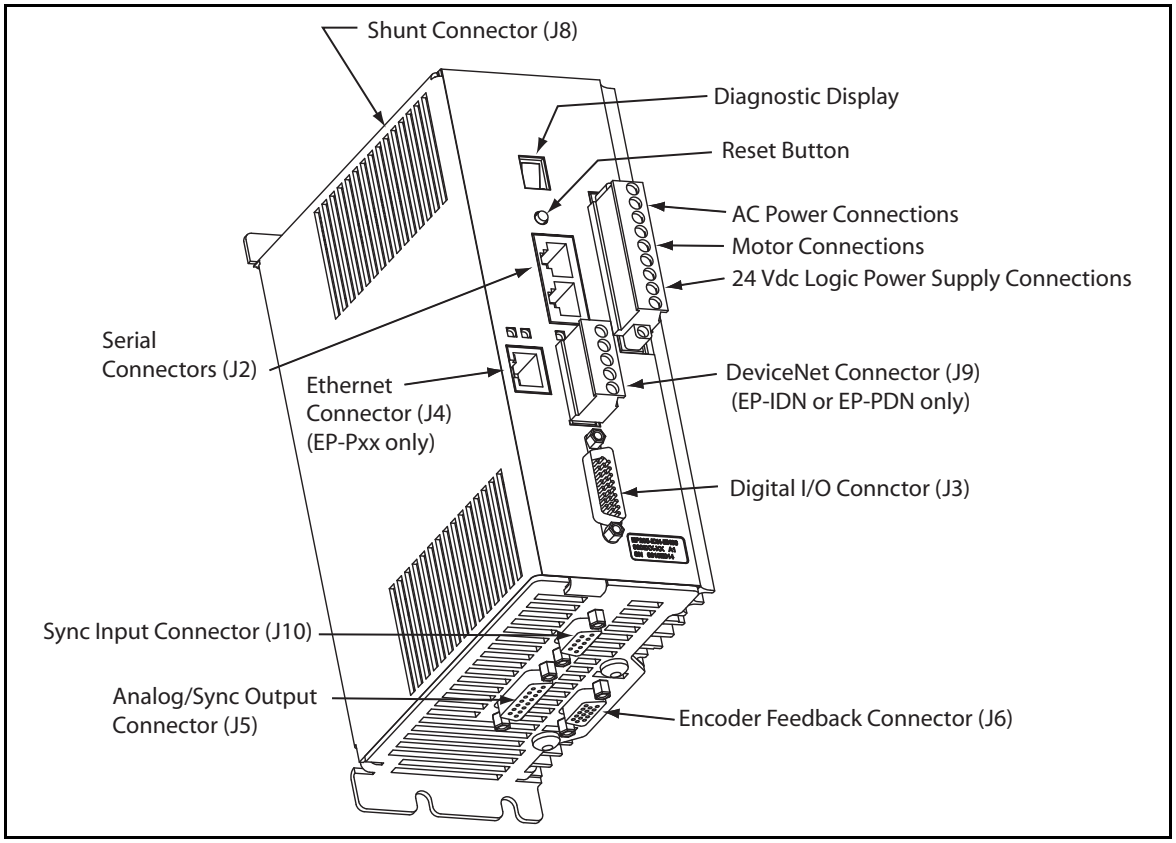


Figure 1: Epsilon EP-PDN (EP202-206) Drive Feature Location

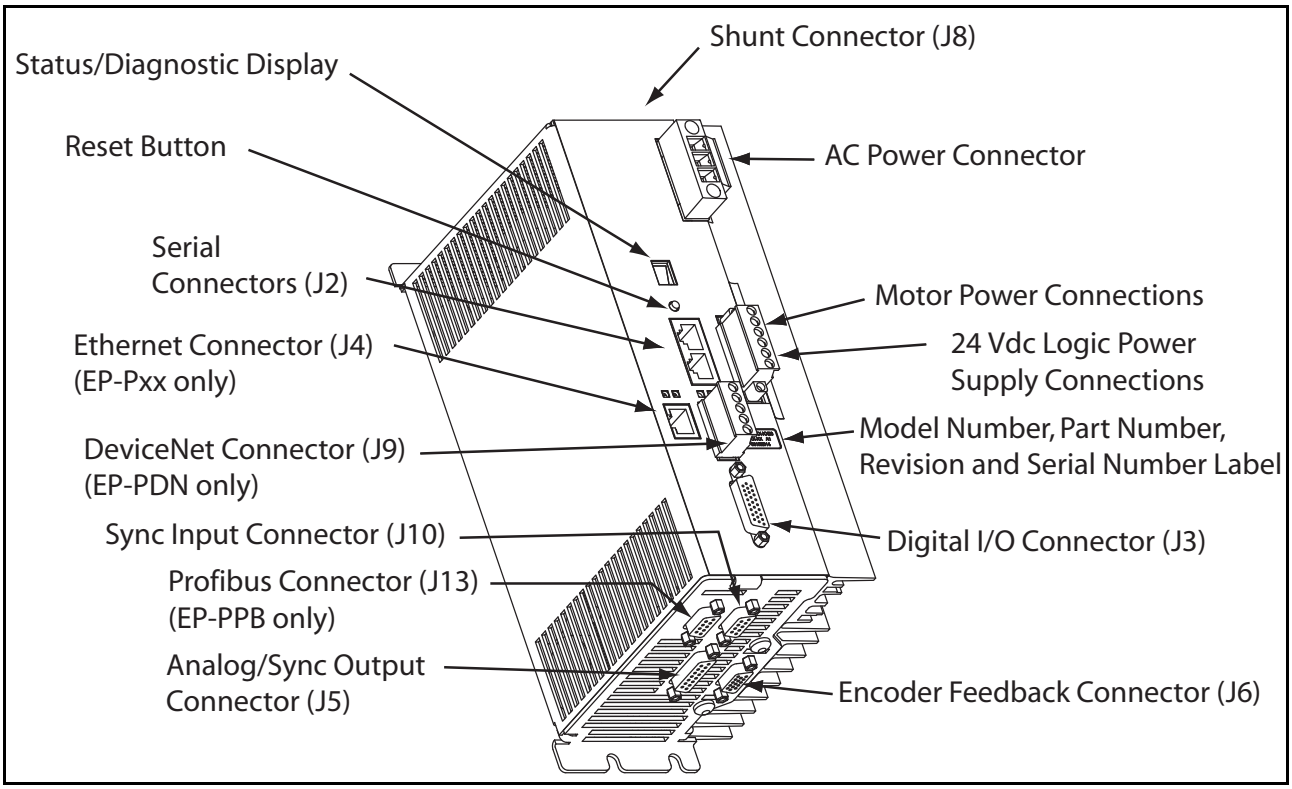


Figure 2: Epsilon EP-209 Drive Feature Location

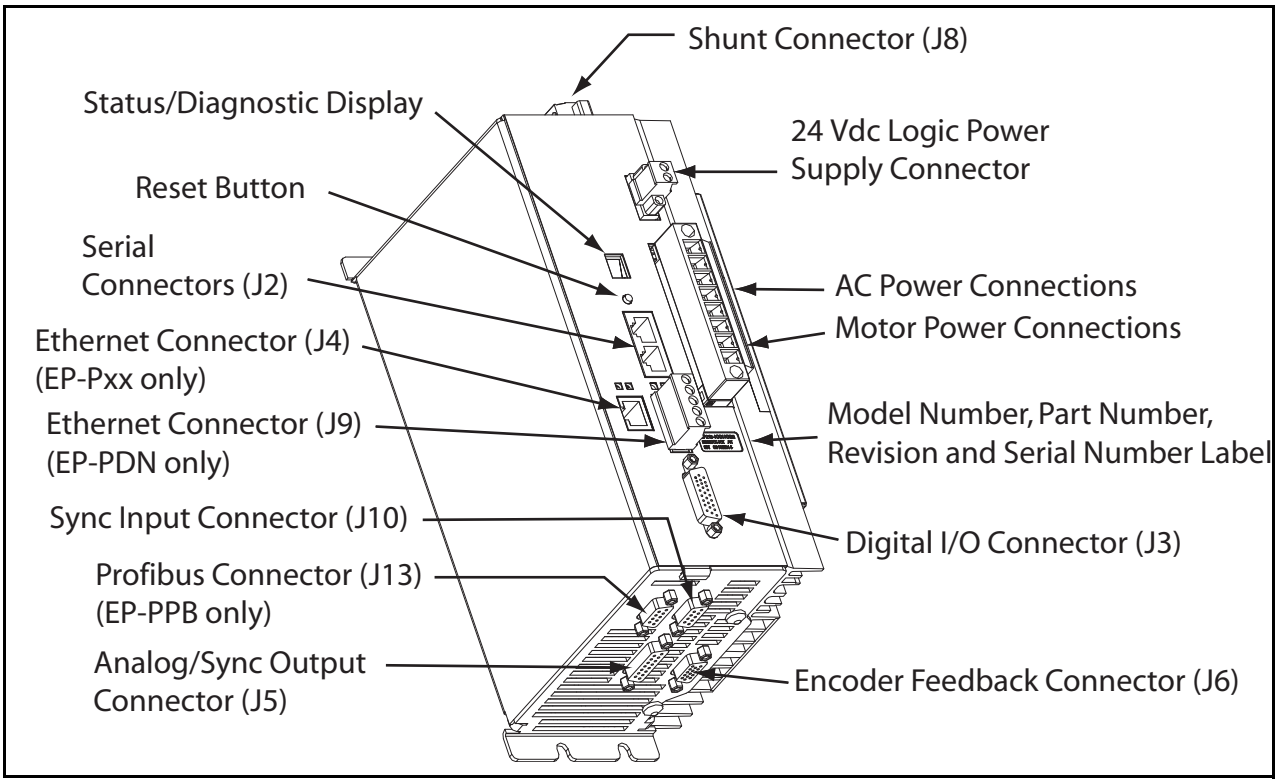


Figure 3: Epsilon EP-216 Drive Feature Location

2 Operational Overview

This section provides a complete functional description of the Epsilon EP-P drive. It is intended to provide you, the user, with a thorough understanding of all operations. The description includes references to many Epsilon EP-P drive parameters which can be displayed and/or edited using PowerTools Pro software, or through any Modbus interface.

The Epsilon EP-P drive allows the user to set up 100 different Indexes, Jog functions and a Home. The Epsilon EP-P drive provides fifteen digital inputs and eight digital outputs.

2.1 Software Interface

The Epsilon EP-P drive is set up using PowerTools Pro software. PowerTools Pro is an easy-to-use Windows® based setup and diagnostics tool. It provides the user with the ability to create, edit and maintain the drive's setup. You can download or upload the setup data to or from a drive. The setup data can also be saved to a file on the PC or printed for review or permanent storage.

2.2 PowerTools Pro Setup Software

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

Features

- "Hierarchy Tree" for quick navigation to any setup view
- Simple I/O function assignments
- Powerful online diagnostic capabilities
- Programming

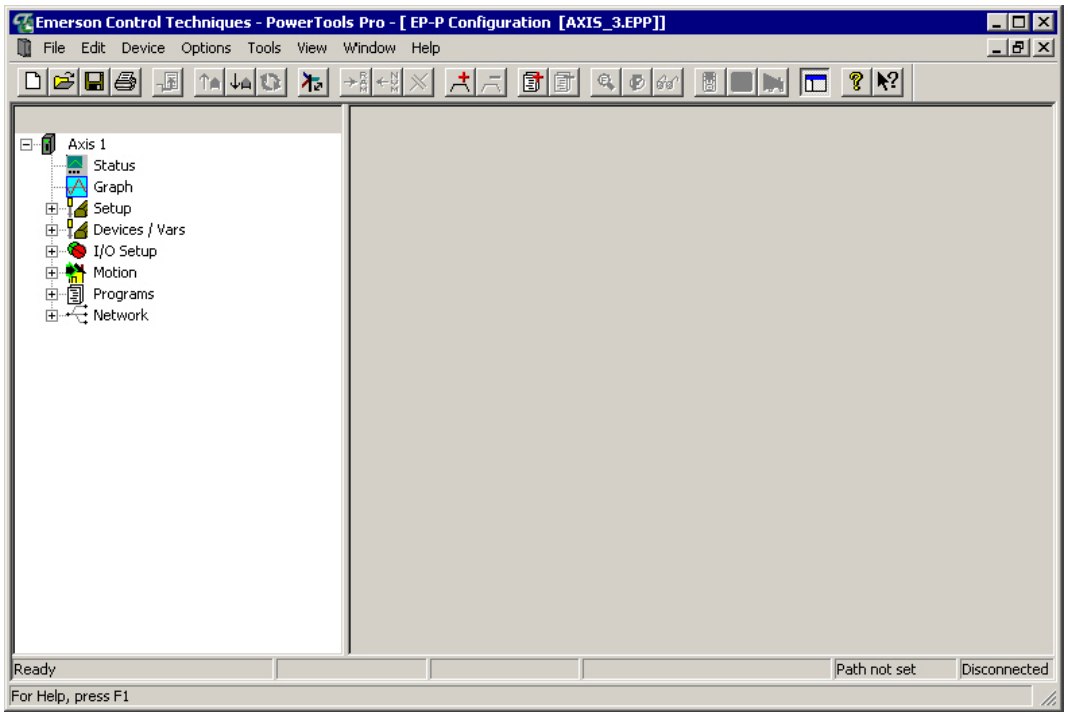


Figure 4: Hierarchy Tree

The "Hierarchy Tree" (shown above in the left side) contains expandable groups of parameters. The groups can be expanded and contracted just like folders in Windows® Explorer. Left click on a view name in the Hierarchy Tree will display that view on the right side of the computer screen.

To setup a drive the user simply steps through the Hierarchy Tree from top to bottom starting with the Setup view. Simple applications can be setup in a matter of minutes.

2.3 How Motion Works

The Epsilon EP-P drive provides six types of motion: jogging, homing, indexing, gearing, camming, and torque mode. The drive will be either in velocity mode or torque mode. What this means is that while the drive is in torque mode, the other types of motion cannot run. If the other types of motion are active on either profile, the torque mode may not run. It is possible to run two different velocity mode types of motion at the same time. For instance, gearing could be running on profile zero while an index is running on profile one. Note that the same instance of a motion type may not run on both profiles at the same time. This means that while index 0 is running on profile zero, it may not run on profile one. But while index 0 is running on profile

zero, index 1 may run on profile one.

Please note that Indexes and Jogs can be run simultaneously by using the two Profiles, however, Gearing or Camming can only run on one profile at a time. This means that two indexes or two jogs can run at the same time, but gearing or camming can not be run on multiple profiles simultaneously.

The Positive direction parameter affects all motion types by specifying which direction of motor revolution (CW or CCW) is considered motion in the “+” direction.

2.4 How Jogging Works

Jogging produces rotation of the motor at controlled velocities in a positive or negative direction.

Assignments to jogs are level sensitive such that when the jog input is turned on, jogging begins and continues jogging until the jog input is removed.

Each jog has its own acceleration and deceleration ramp along with a specified velocity. Jogging has no distance parameter associated with it. If trying to move a specific distance or to a known position, then an index is used.

Jog velocity can be changed on the fly with a negative value reversing the direction. The velocity transition will use the deceleration or acceleration parameters.

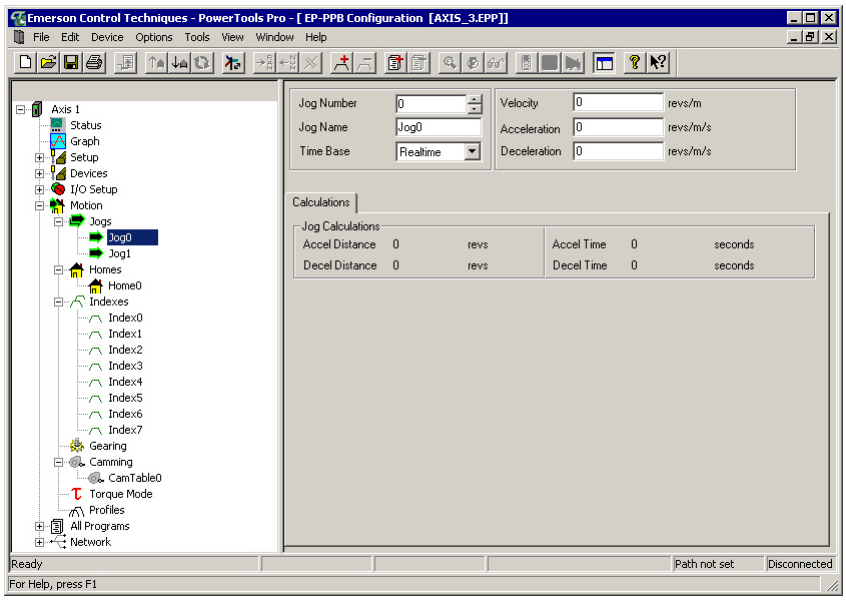


Figure 5: Jog View

2.5 How Home Works

The Home is used in applications in which the axis must be precisely aligned with some part of the machine. The Home is initiated in one of three ways: with the Initiate Destination function found in the Assignments view, through a program, or with the Online tab. A Home or Define Home is required to set the Absolute Position Valid so that any index to absolute position can work.

The Epsilon EP-P drive can home the motor to an external sensor, the motor’s encoder marker pulse, or to a sensor and then to the encoder marker pulse.

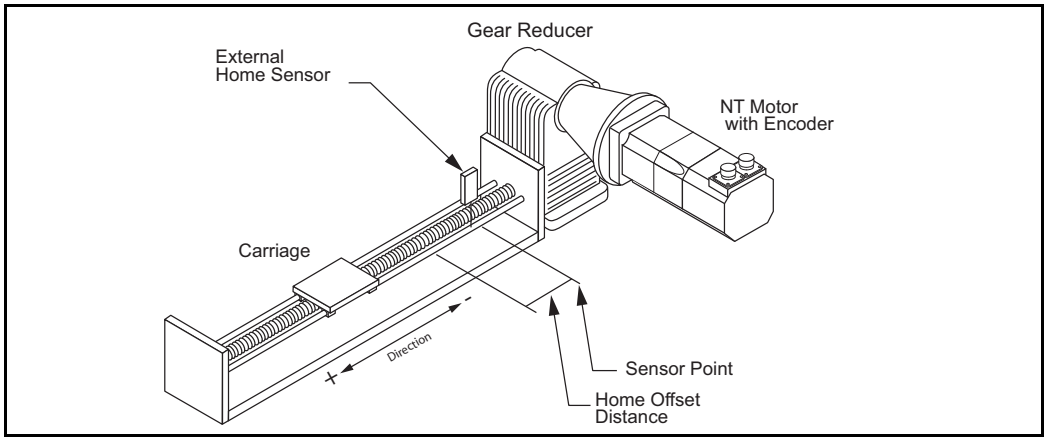


Figure 6: Basic Home Function, Example

The figure above show a basic home function using a ball screw. This example uses most of the setup features in the PowerTools Pro Home

view.

2.5.1 Home Sequence

1. Back off the sensor, if on the sensor. (This step is optional).
2. Move to the external home sensor to establish a home reference point.
3. Next it will move to the Offset position.
4. Then the command and feedback positions are set to the value entered into the End of Home Position.

Homing to the motor's encoder marker will establish the most accurate and repeatable home position. This method will position the motor relative to the location of the rising edge of the encoder marker pulse. Most applications will use a sensor and marker to find an accurate home position in the vicinity of the home sensor.

Several parameters affect how the Home function operates. Each of these parameters are explained in detail on the following pages.

NOTE

The Home function will NOT be initiated when any other motion command is in progress.

2.5.2 Establishing a Home Reference Position

The first step in setting up a home is to select the desired home reference type. The Home Reference type selected determines how the Home Reference Position is established. PowerTools Pro allows selection of one of three different Home Reference types: Sensor, Marker, or Sensor then Marker.

Sensor

Selecting Sensor means the rising edge of the Home Sensor input function is used to establish the home reference position.

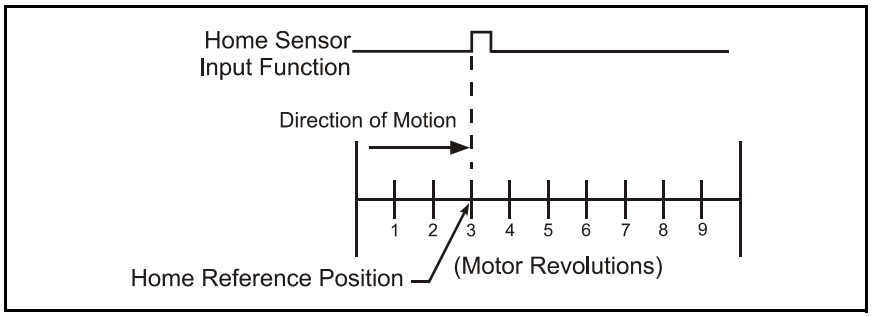


Figure 7: Sensor Home Reference Position

Marker

Selecting Marker means the rising edge of the motor's encoder marker channel is used to establish the reference position.

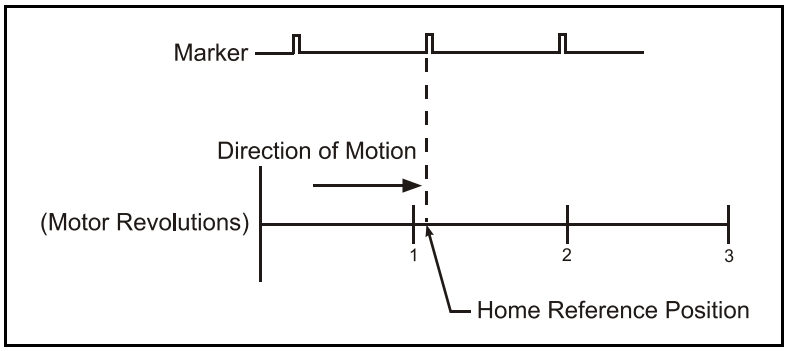


Figure 8: Marker Home Reference Position

Sensor then Marker

Selecting Sensor then Marker means the reference position is established using the first marker rising edge after the drive sees the rising edge of the Home Sensor input function.

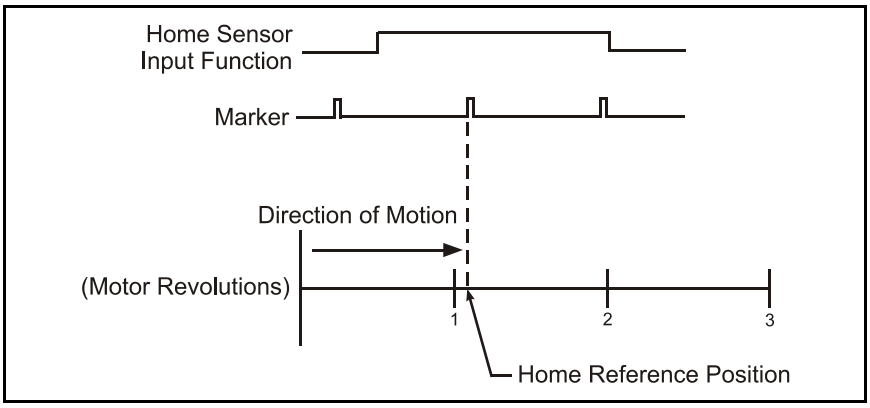


Figure 9: Sensor then Marker Home Reference Position Example 1

2.5.3 Accuracy and Repeatability

The accuracy is one trajectory update rate. For example - if the trajectory update rate is set to 800 µs then the accuracy will be 800 µs, if the trajectory update rate is set to 1.6 ms then the accuracy will be 1.6 ms.

The amount of accuracy the application requires will determine the Home Reference type selected. Homing to an external sensor will only establish a repeatable home position within 0.04 revolutions at 3000 RPMs (800 µsec sensor capture interval).

NOTE

The data above assumes the use of a perfectly repeatable home sensor.

In Sensor then Marker applications, the marker must be at least 800 µsec after the rising edge of the sensor input to be considered a valid marker pulse, see Figure 10.

NOTE

At 1000 RPM, the motor will travel 0.0133 revolutions (or 4.8°) in 800 µsec.

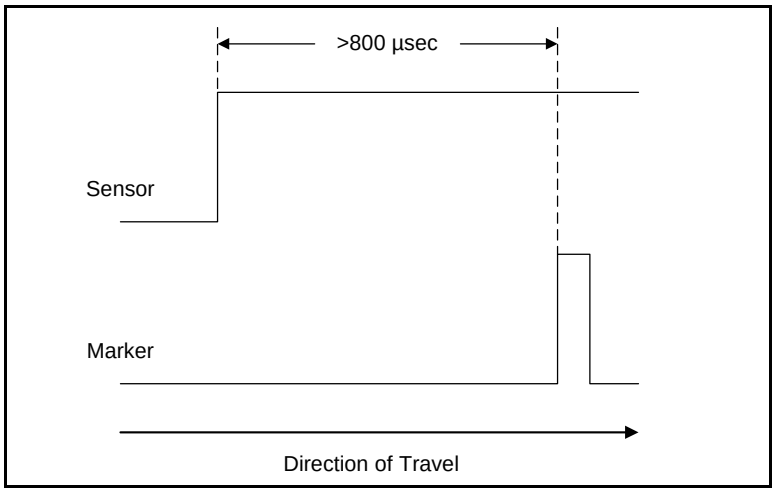


Figure 10: Sensor then Marker Home Reference Position Example 2

The Home Sensor must be “On” for at least 800 µsec to guarantee that it will be recognized.

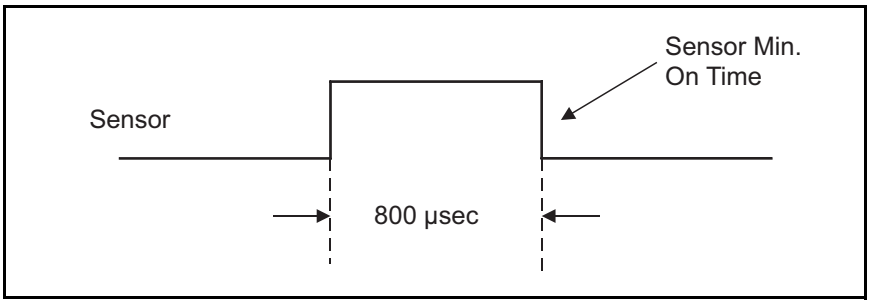


Figure 11: Sensor then Marker Home Reference Position Example 3

2.5.4 Home Offset

The Home Offset is the distance from the home reference position to the final stopping point at the end of the homing sequence. Regardless of

the value you enter for the Offset or which Home Reference type you choose, there is always an offset inherent in the homing process.

The user may either specify a desired offset or allow the drive to calculate an offset automatically. The drive calculates an offset that guarantees that the motor will not have to backup to get to the offset position. This is very convenient for unidirectional applications.

The Calculated offset is the distance travelled during deceleration ramp from the home velocity to a stop plus the distance travelled at the home velocity for 800 usec. This extra distance is used to guarantee that the motor will not need to backup after the deceleration ramp.

The Specified offset allows the user to choose an exact offset from the home reference. Once the home reference is detected, the drive will do whatever is necessary to reach the offset position. This may be as simple as a deceleration to a stop, a continuation at speed followed by a deceleration to a stop, or a deceleration followed by a move in the opposite direction.

To enter a Specified offset, select the Specified offset radio button. PowerTools Pro always displays the calculated offset value as a reference.

Offset Examples

With a Calculated offset if the home reference is detected before the axis has reached its peak velocity, the axis will still continue to the precise offset position, see Figure 12.

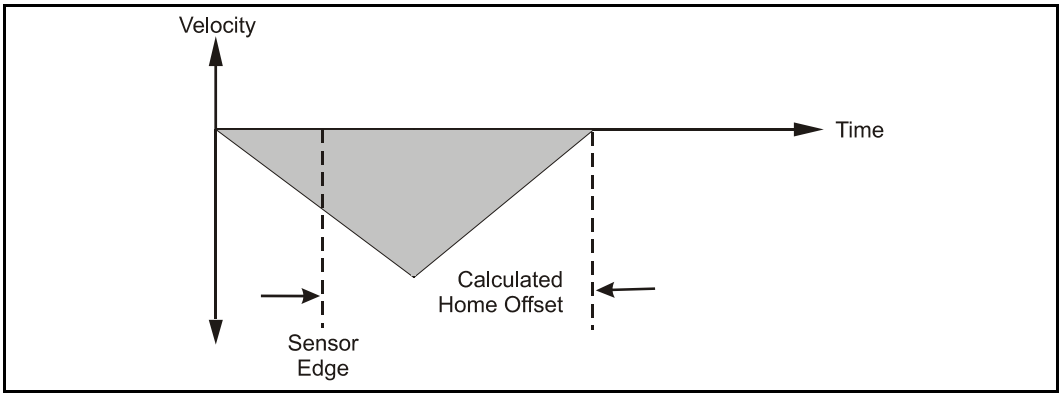


Figure 12: Calculated Home Offset, Peak Velocity Not Reached

If the home reference is detected after the axis has reached its peak velocity, the axis will decelerate to the precise offset position.

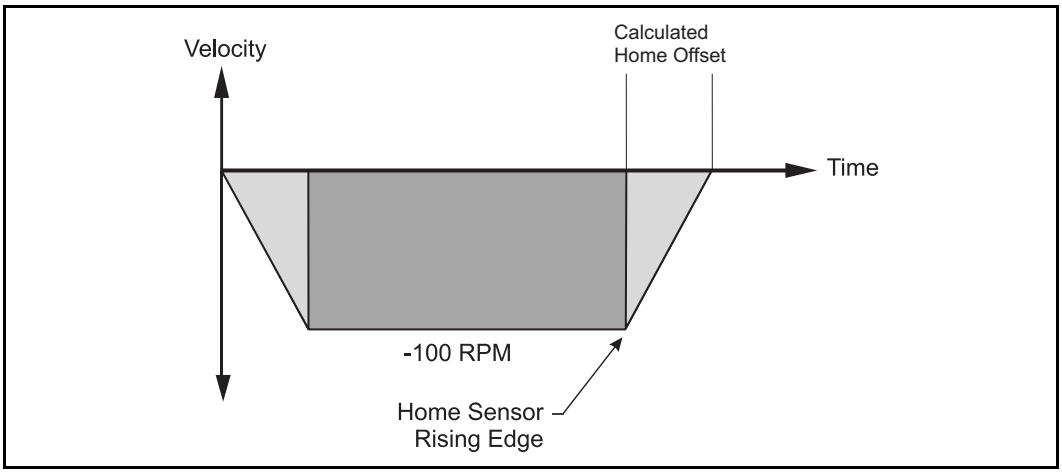


Figure 13: Calculated Home Offset, Peak Velocity Reached

Two examples below show operation when the specified offset is greater or lesser than the calculated offset. This causes the axis to continue on at speed before decelerating and stopping at the offset position, or backing up after the home sensor.

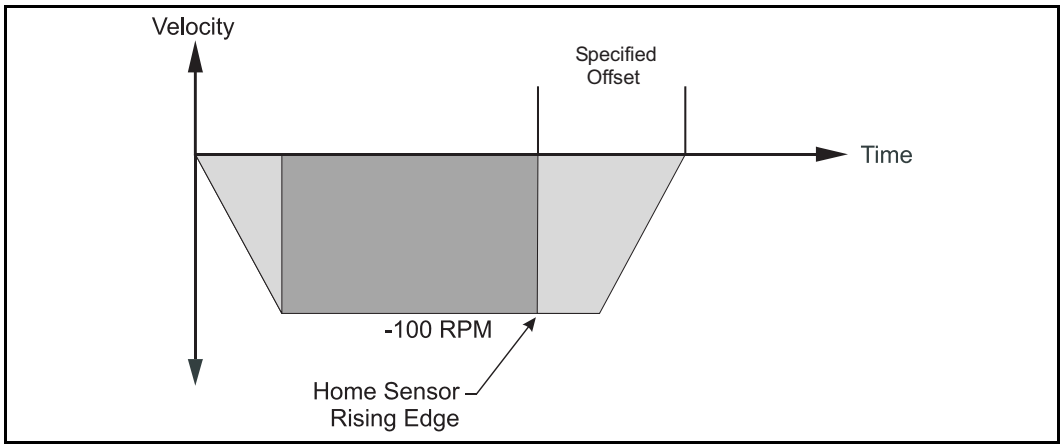


Figure 14: Specified Home Offset, Greater than Calculated Offset

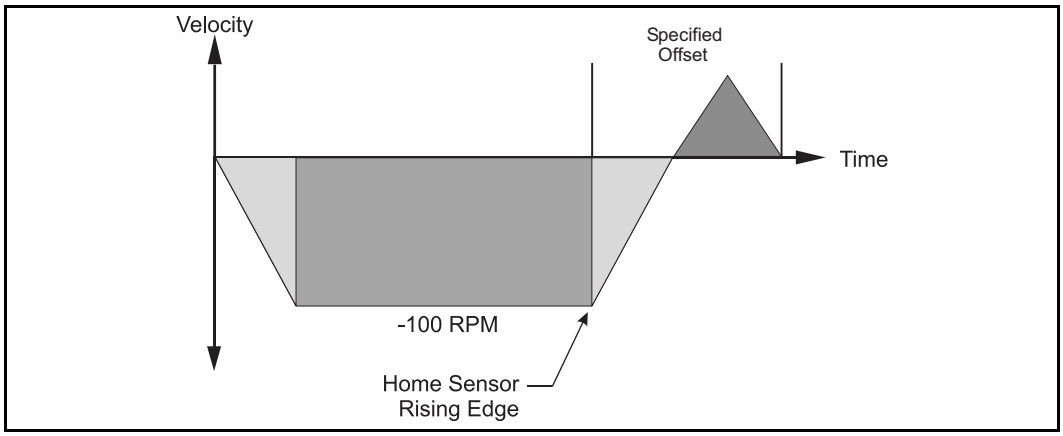


Figure 15: Specified Home Offset, Backup Required

2.5.5 End of Home Position

The End of Home Position (End Posn) defines the home position in relation to the machine's coordinate system. At the completion of the home, the value of the End of Home Position is put into the command position.

2.5.6 Home Limit Distance

This parameter places an upper limit on the incremental distance the motor will travel during the home.

If no reference is found, the system will decelerate and stop at the limit distance. The Home Limit Distance Hit function will be activated if the home stops at the limit distance without finding the reference. Additionally, the Home.CommandComplete function will not turn "On" if the limit distance is hit.

2.5.7 Home Examples

Linear Application

In this example, the system uses an external sensor and the motor's encoder marker channel to establish a Home Reference Position. This is the most accurate and most common way to home.

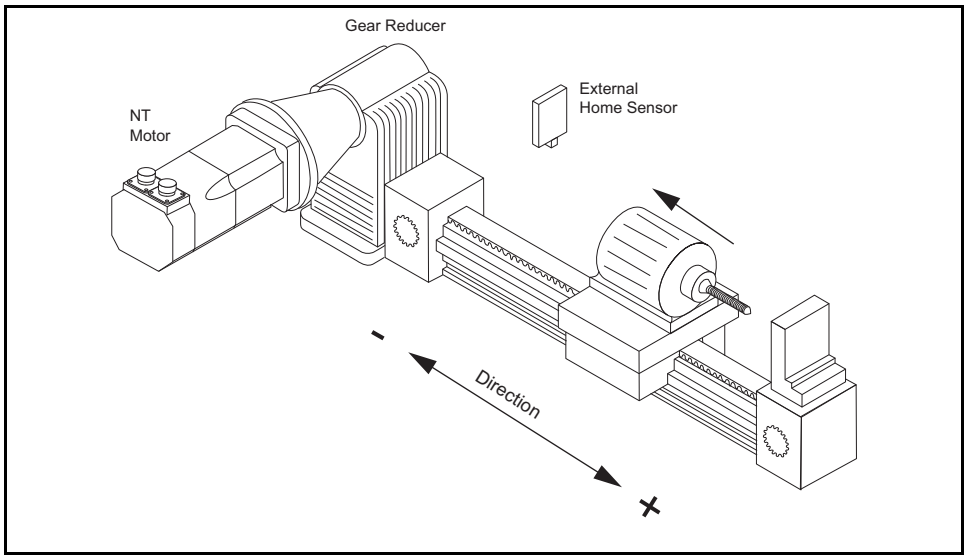


Figure 16: Home to Sensor and Marker, Example

When the drive sees the Home Initiate, it accelerates the motor to the Home Velocity.

The motor continues at that velocity until it first senses the Home Sensor input. It continues at the same velocity until the motor's encoder marker channel is sensed. The rising edge of the motor's encoder marker channel is used to establish the reference position. Once the home reference is detected, the motor decelerates to a stop and moves to the offset position.

Home Sequence

1. If on sensor then back off (if enabled)
2. Search for sensor
3. Search for marker
4. Go to offset (2.0 Revs)
5. Set feedback position equal to End of Home Position

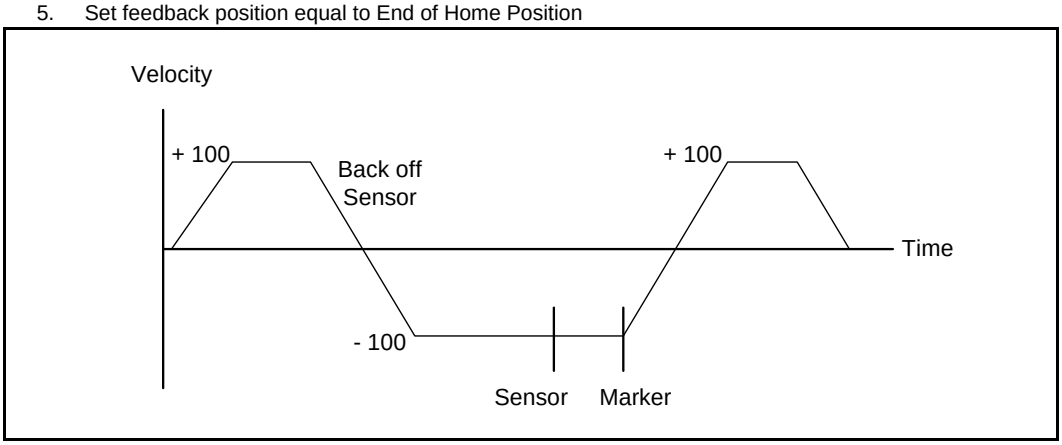


Figure 17: Home Velocity Profile

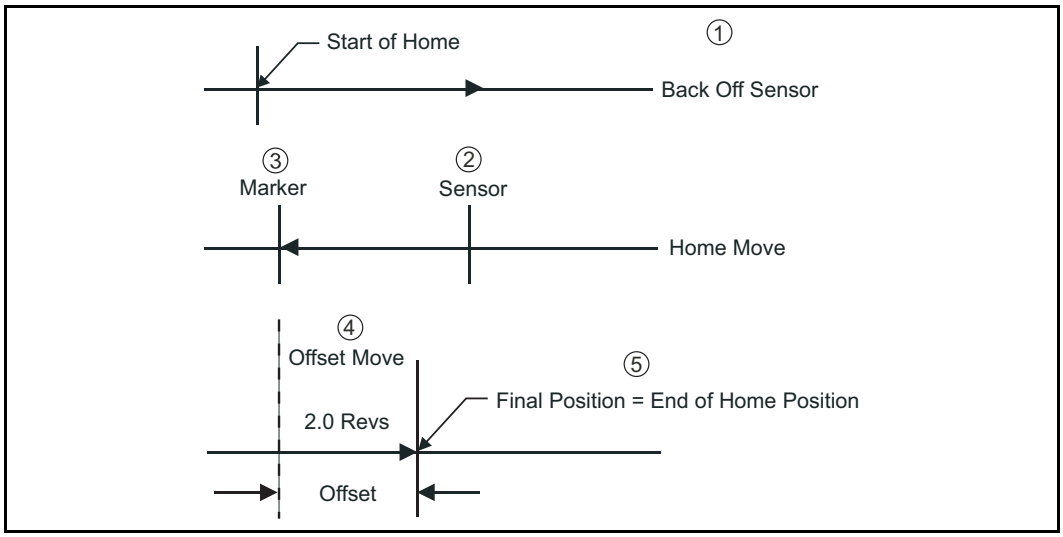


Figure 18: Home Move Sequence

Rotary Application

This example uses an external sensor and the motor's encoder marker pulse to establish a home reference position.

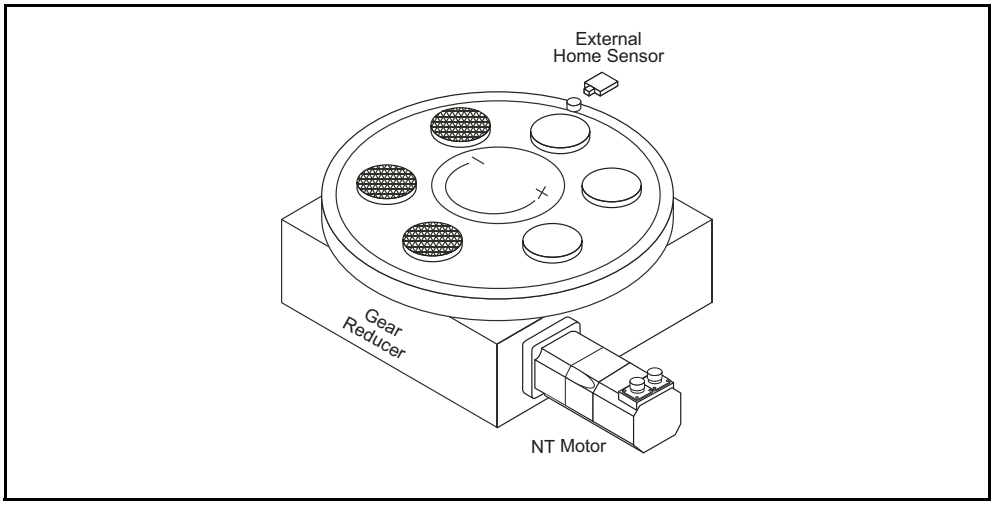


Figure 19: Home Sensor and Marker then Offset, Example

When the drive sees the rising edge of the Home Initiate function, it accelerates the motor to the Home Velocity. The motor continues at that velocity until it first senses the Home Sensor input. The motor continues on at the home velocity until the marker is activated.

The rising edge of the motor's encoder marker channel is used to establish the reference position.

After sensing the rising edge of the motor's marker channel, the drive will continue moving and will decelerate to a stop at the specified offset position.

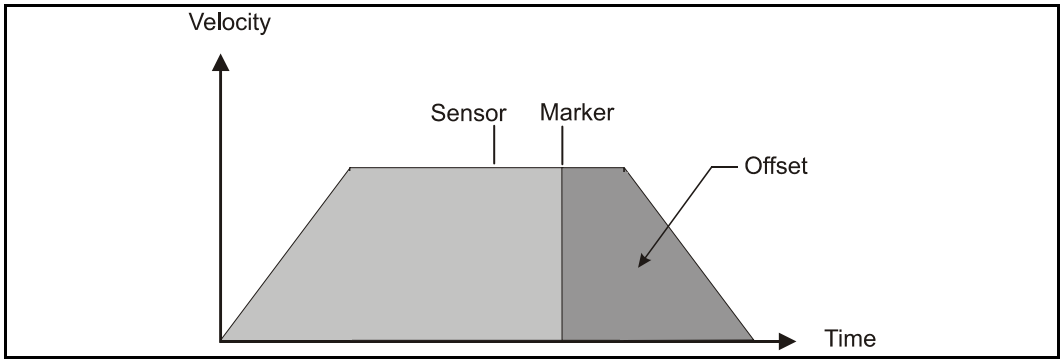


Figure 20: Home Velocity Profile

2.6 How Indexes Work

An index is a complete motion sequence that moves the motor a specific incremental distance or to an absolute position. This motion sequence includes an acceleration ramp to a programmed velocity, a run at velocity, and a deceleration ramp to a stop.

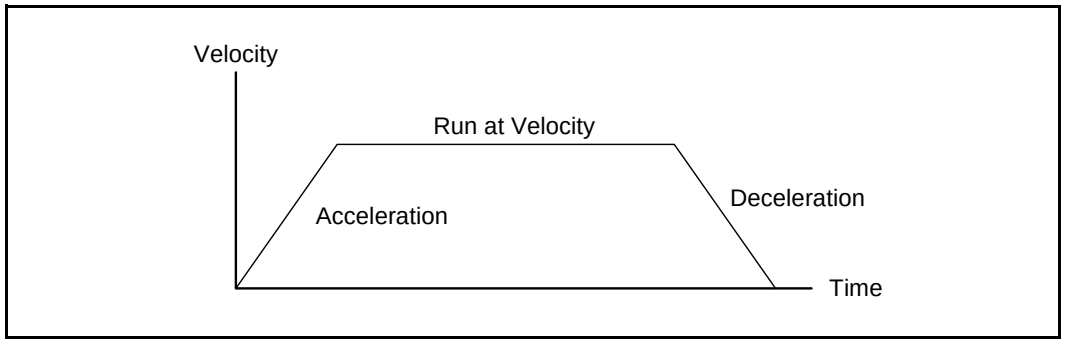


Figure 21: Index Motion Sequence

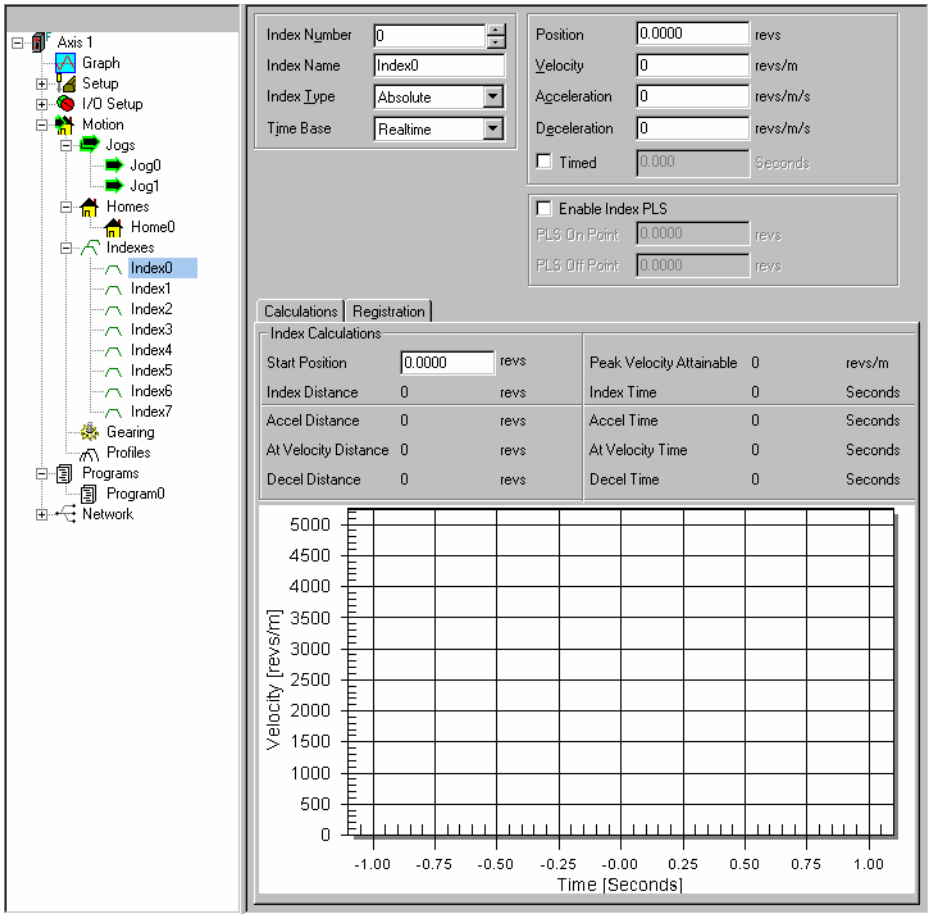


Figure 22: Indexes View

Indexes use acceleration and deceleration ramps which may or may not reach the specified velocity depending on the total distance and the ramp values. For example, a short move with long acceleration and deceleration ramps may not reach the target velocity entered.

Indexes cannot be initiated when any other motion (jogging, homing, or program) is in progress. Indexes can be aborted with the Stop destination found in the Ramps group on the Assignments View.

The Epsilon EP-P supports eight types of indexes: absolute, correction, incremental, posn track cont., posn track once, registration, rotary plus and rotary minus.

2.6.1 Absolute vs. Incremental

The difference between absolute and incremental indexes is that absolute indexes move to a specific absolute position and incremental indexes move the motor a specific distance. The following figures and explanations demonstrate this concept.

Absolute Indexes

Absolute indexes are used in applications where the motor must travel to a specific position, regardless of where the motor is when the index is initiated.

The drive calculates the distance required to move to the specified position from the current position.

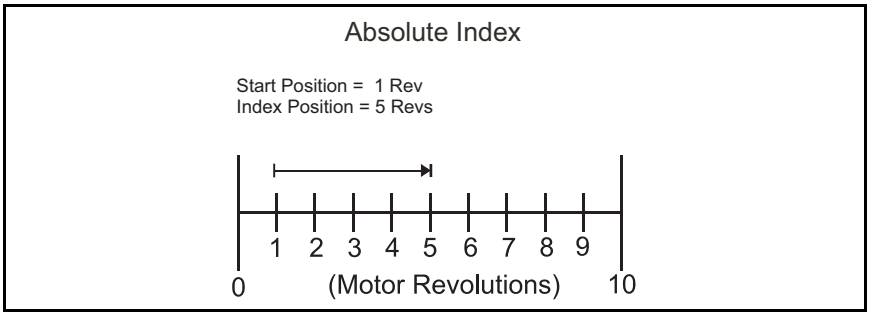


Figure 23: Absolute Index Example 1

In the example above, the current position is 1 rev. If this index is initiated, the motor will travel to a position of 5 revs no matter where it is sitting before the move. From 3 revs, it will travel 2 revs to finish at 5 revs. If the absolute index to 5 revs is initiated a second time immediately after the index, no motion will occur because the motor will already be at a position of 5 revs.

The direction of an Absolute Index is determined by the starting position and the absolute index position. If the starting position for the above index is 9 revs, then the motor will rotate in the negative direction to end up at 5 revs. The figure below shows this.

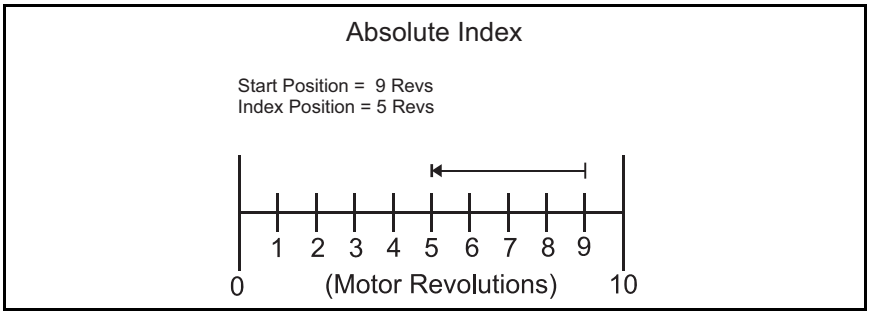


Figure 24: Absolute Index Example 2

Absolute indexes with Rotary Rollover enabled will take the shortest path to the position entered in the index position parameter.

NOTE

Absolute indexes move to positions relative to where the machine was homed using the Home, or the DefineHome destination.

Incremental Indexes

An incremental index will move the motor a specified distance in the + or - direction regardless of the starting position. The direction of the incremental index motion is determined by the sign (+ or -) of the Index Distance parameter.

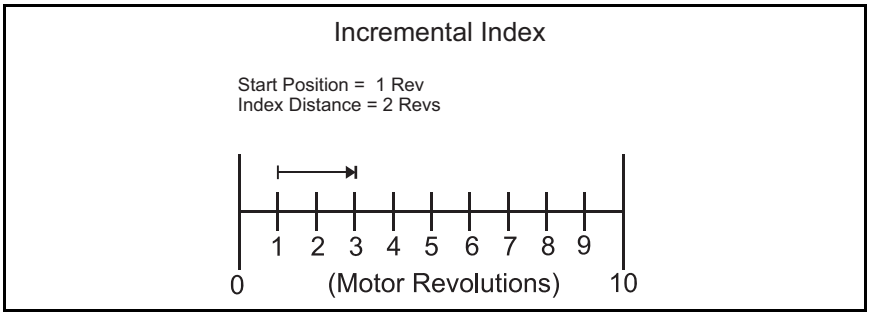


Figure 25: Incremental Index Example

In the example above, the motor starts at 1 rev, travels a distance of 2 revs and stops at 3 revs. If the same index is initiated a second time, the drive would move the motor another 2 revs to a position of 5 revs. If initiated a third time, the motor would travel another 2 revs to a final position of 7 revs. Figure 26 shows this operation.

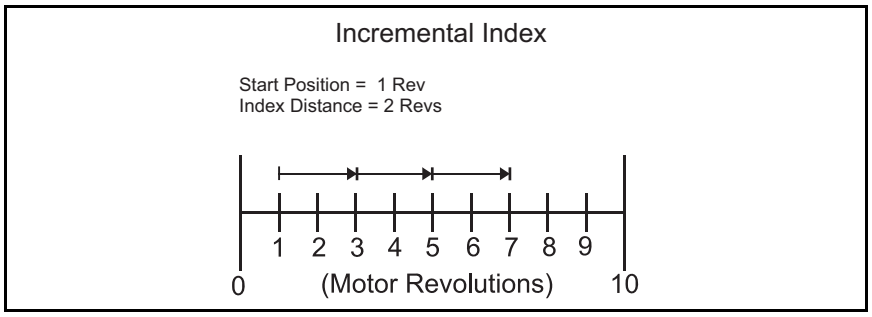


Figure 26: Incremental Index Example 2

Correction Indexes

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. Another 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

Posn Tracker Cont and Posn Tracker Once Indexes

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.

Registration Indexes

A Registration Index is used in applications where the motor must move until an object is detected and then move a specific distance from the point of detection, such as finding a registration mark and moving a distance beyond.

The Registration Index consists of two parts. The first part accelerates the motor to the target velocity and continues at this velocity until it receives a registration trigger (sensor or analog). Upon receipt of a registration trigger, the registration offset will be executed at the target velocity. The Sensor Limit Distance Hit source can be used to turn on an output, if a sensor input or analog limit is not received within the Limit Distance. A registration window can also be used to determine the validity of a registration trigger. If a registration trigger is received outside of the registration window, it will be ignored.

Rotary Plus and Rotary Minus Indexes

Rotary Plus and Rotary Minus Indexes provide forced directional control of moves to absolute positions. The position entered for a Rotary Plus or Minus type index must be within the rotary range (i.e. 0 ≤ Position < Rotary Rollover Point). All other parameters function the same as they do with absolute indexes. An Absolute Index is a direct move to a specific position, regardless of the starting point. A Rotary Plus Index moves to the specified position, but is forced in a positive direction. Similarly, a Rotary Minus index moves to the specific position, but is forced in a negative direction.

Rotary Plus and Minus Indexes are usually used in rotary applications, therefore the rotary rollover feature on the Setup - Position view in the PowerTools Pro software must be enabled to use them.

1. In the following examples the term “D” = (absolute position specified) - (current position). If “D” is negative, motion in the negative direction is implied.
2. In the following examples the Rotary Rollover parameter on the Setup - Position view is set to 360.00°. This means that with each revolution of the motor (or rotary table), feedback will count up to 359.99°, then roll over to Ø°.

Indexes with Rotary Rollover Enabled

Incremental move distances can be outside of the rotary rollover range. See the "Setting Up Parameters" chapter for an explanation of Rotary Rollover.

Example 1: If the starting position is at Ø° and 720° is the specified distance, an Incremental index would move 2 revolutions in the positive direction. At the completion of this index the motor position would be Ø°.

Absolute indexes will take the shortest path to the specified position. Absolute index positions must be within the rotary

rollover range.

Example 2: If the starting position is at 90° and 80° is the specified position, an Absolute index would travel 10° in the negative direction. At the completion of this index the motor position would be 80°.

Example 3: If the starting position is 45° and 315° is the specified position, an Absolute index would travel 90° in the negative direction because that is the shortest path between 45° and 315°.

Rotary Plus indexes will move to the specified position and are forced in a positive (or plus) direction. Rotary Plus index distances must be within the rotary rollover range.

Example 4: As in example 2 above, the starting position is at 90° and 80° is the specified position. A Rotary Plus index would travel 350° in the positive direction. At the completion of this index the motor position would be 80°.

Example 5: If the starting position is 10° and the specified position is 350°, a Rotary Plus index will travel 340° in the positive direction.

Rotary Minus indexes move to the specified position, but are forced to travel in the negative (or minus) direction. Rotary Minus index positions must be within the rotary rollover range.

Example 6: As in examples 2 and 4 above, the starting position is at 90° and 80° is the specified position. A Rotary Minus index would travel 10° in the negative direction. At the completion of this index the motor position would be 80°.

Example 7: If the starting position is 15° and the specified position is 270°, a Rotary Minus index would travel 105° in the negative direction.

2.7 How Communications Work

2.7.1 Configuring Communication

Before attempting to upload or download a configuration file using PowerTool Pro, the software must be configured to the correct communication settings for the intended communication connection. The Epsilon EP-P drive supports both RS-485 serial communication connections (J2) and Ethernet communication (J4) connection, on the front of the drive.

When downloading or uploading a selected configuration file that is open in PowerTools the communication scanner will only scan the configuration file's Modbus ID and Ethernet address (unless Ethernet change address is clear). Once a connection is established, that connection (Comm port used, port number, modbus ID and Ethernet address) is saved, a second download, upload or reconnect will bypass the scanner and use this previously defined communication path. To change the communication path use the Change Path toolbar button.

The communication connection may be selected in the Upload Drive Configuration, Download to Device IDx or the Change Path dialog boxes. From the **Device** menu, choose **Upload Drive**, **Download** or **Path Change** to open the dialog box or the toolbar buttons can also be used, see below.

2.7.2 Upload Drive

Uploading is the process of reading information back from the drive to the PowerTools Pro configuration file views.

To upload information from a drive, click on the ***Upload All*** button, on the PowerTools Pro toolbar or from the **Device** menu, choose **Upload All** or **Upload Drive**. The Upload Drive Configuration dialog box will open, all communication connections are scanned and the results appear. In Figure 27, it shows that one drive on COM 1 was found, an Epsilon EP204-P drive. The Upload Drive Configuration dialog box contains the following information for every drive found:

- Ip Address/COM
- Modbus Address ID
- Drive Type
- Module Type
- Communication Options
- Base/Drive FW Revision
- Module FW Revision
- Module Serial Number
- Drive Serial Number

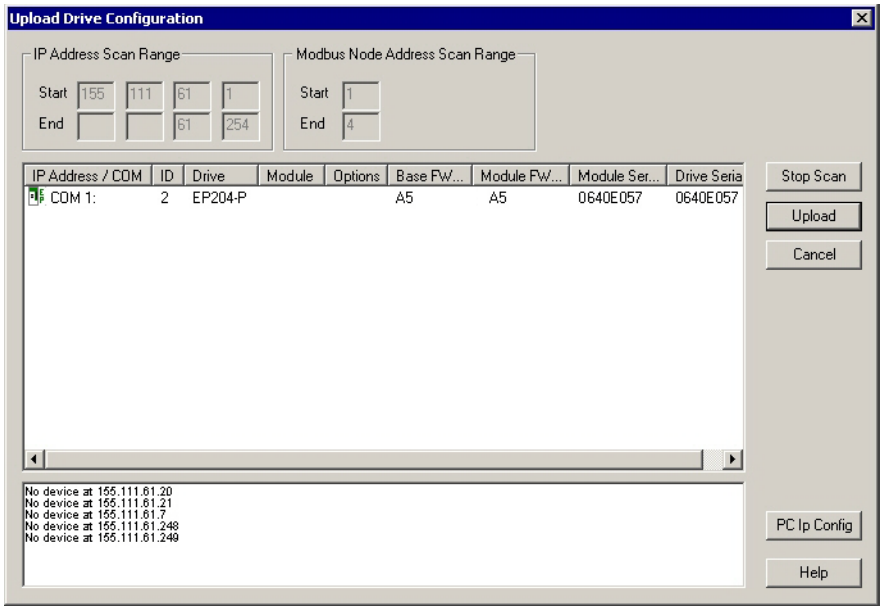


Figure 27: Upload Drive Configuration Dialog Box

Select the drive to upload and click ***Upload***.

2.7.3 Downloading

Downloading is the process of sending the configuration created with PowerTools Pro from the PC to the drive. Changes made in PowerTools Pro will not take effect until the information has been downloaded or the Update to RAM button has been clicked.

To download information to a drive, click the ***Download*** button on the PowerTools Pro toolbar or from the **Device** menu, choose **Download**. The Download to Device IDx (x equals the modbus address) dialog box will open, all communication connections are scanned and the results appear. In Figure 28, one drive on COM port 1 was found, it's a Epsilon EP204-P drive. The Download Drive Configuration dialog box contains the following information for every drive found:

- Ip Address/COM
- Modbus Address ID
- Drive Type
- Module Type
- Communication Options
- Base/Drive FW Revision
- Module FW Revision
- Module Serial Number
- Drive Serial Number

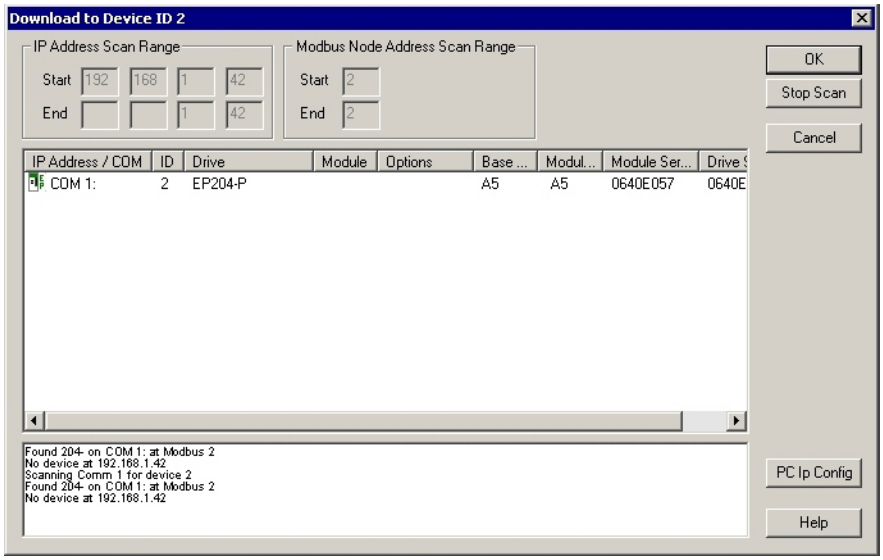


Figure 28: Download to Device ID 1 Dialog box

Select the drive to download to and click ***OK***.

2.7.4 Change Path Connection

This function allows the user to change the drive and Ip address/Com port used for download and upload. It is used when the user has already selected one Ip Address/Com port and wishes to change to another.

The dialog box provides the user with communication information available on the Modbus and Ethernet network. This information contains:

- Ip Address/COM
- Modbus Address ID
- Drive Type
- Module Type
- Communication Options
- Base/Drive FW Revision
- Module FW Revision
- Module Serial Number
- Drive Serial Number

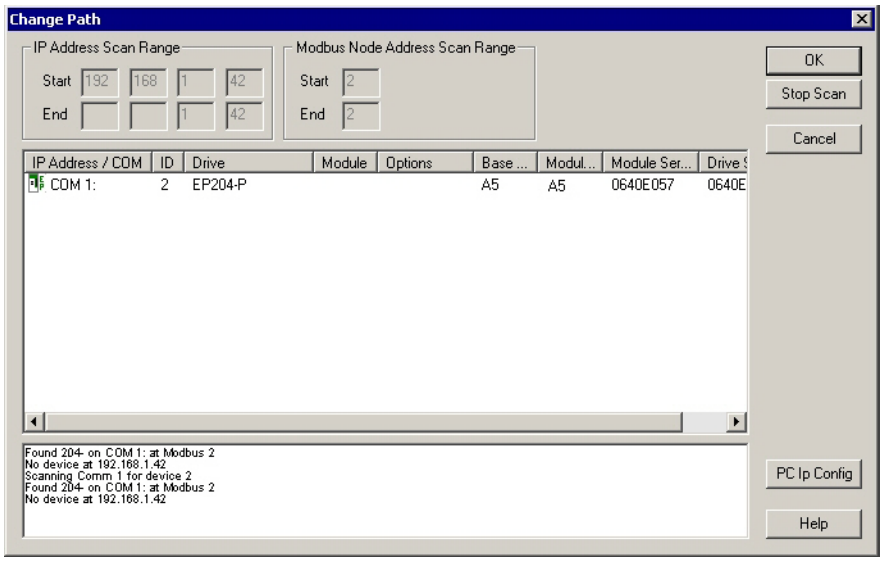


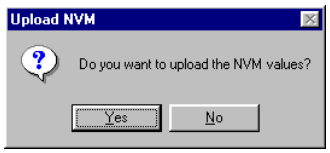
Figure 29: Change Path Dialog Box

Select the drive in the list and then click ***OK***. The communication connection path will then be displayed in the status bar at the bottom of PowerTools Pro window.

2.7.5 NVM Options for Uploading and Downloading

Uploading

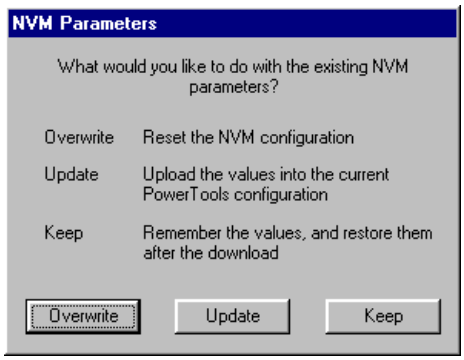
When uploading from a drive, the values that were last downloaded are uploaded and put into a PowerTools Pro configuration file. At the completion of the upload, the user will be asked if they wish to upload the NVM values. This dialog box is shown below.



By selecting Yes, the values of all parameters stored in NVM will be uploaded and entered into the PowerTools Pro file values. If No is selected, the values entered into the PowerTools Pro file will remain the same as those that were last downloaded to the drive.

Downloading

When downloading to the drive the user will be required to select how to handle the NVM parameters upon downloading. The dialog box asking the user to select one of three options for the download is shown below.



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 (PowerTools Pro file). 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 drive and enter them into the user configuration (PowerTools Pro file). Once the NVM values have been stored in the file, the file is fully downloaded.

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 (PowerTools Pro file).

The following table shows an example of how these options work:

	Before Download	After Download		
		Overwrite Option	Update Option	Keep Option
PT Pro file value for Index.0.Vel	150	150	500	150
NVM value for Index.0.Vel	500	150	500	500

2.7.6 Updating to RAM

The ***Update to RAM*** button can be used to send changes to the drive without performing a complete download. The ***Update to RAM*** button is found in the PowerTools Pro toolbar. This operation will send only those changes that have been made since the last ***Update to RAM*** or a **Device>Download** to the drive was done. The changes will take effect immediately upon clicking on the button.

CAUTION	The parameters will be sent to the drive without stopping motion or disabling the drives. 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 below describes a typical process using the Update to RAM to make changes, and then downloading when complete to save changes to NVM.

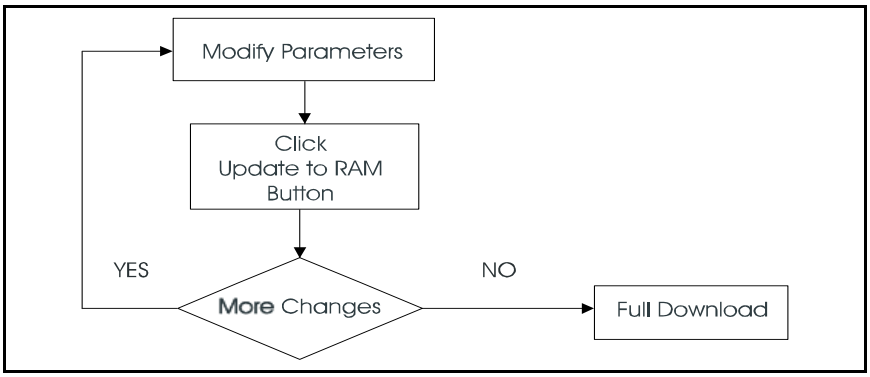


Figure 30: Update to RAM Flow Chart

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

- If no parameters have been modified by the user, 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 be 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 drive. 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, when the download is complete, the ***Update to RAM*** button will be disabled
- If the user modifies parameters, and disconnects, the ***Update to RAM*** button will be disabled, and the changes will not be sent.

2.7.7 Options/Preferences/Ptools Operation

Communications Tab

This tab allows the user to set-up the serial communication baud rate. The drive baud rate and PowerTools Pro baud rate must match, default drive baud rate = 19200. The Enable Scan group allows the user to select which communications connections are scanned when doing any communication operations. Windows allows up to 256 COM ports. PowerTools Pro scans the Windows registry to find all the defined COM ports. The scanner sends commands down each of the COM ports to find the drives. Default = All ports are scanned. The user also has the control of which Modbus node address are scanned when doing any communication operations The range is 1 to 247 but since it's not practical from a time stand point to scan all nodes in the range the default is 1 to 4 node addresses. This scan range can be changed by the user.

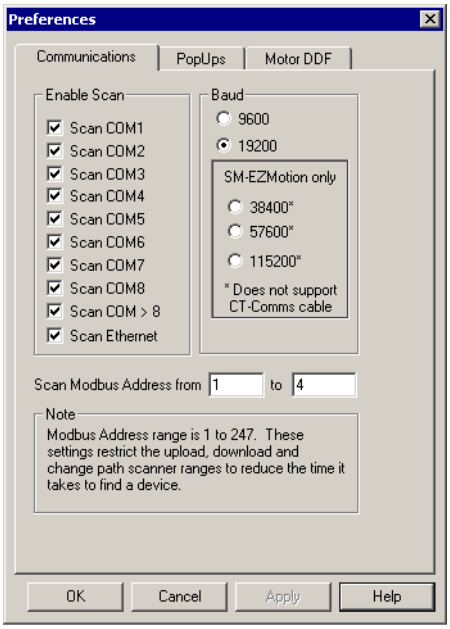


Figure 31: Preferences-Communications Tab

PopUps Tab

The options in this dialog box controls the dialog boxes that the user encounters when uploading and downloading the configuration file.

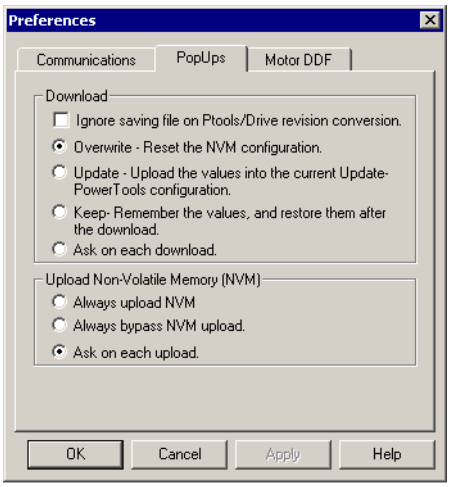


Figure 32: Preferences-PopUps Tab

Download Section:

Ignore saving file on Ptools/Drive revision conversion.

On a download PowerTools Pro checks the firmware revision of the drive that it is about to be downloaded to and is required to make changes to files that are to be downloaded to older firmware revisions. This check box allows the user to avoid saving the newer file before converting it to a previous revision.

Overwrite – Reset the NVM configuration.

When this option is selected the “Overwrite” function will default on every download to the drive. This function will overwrite the entire configuration including user defined NVM parameters as set in the NVM setup area of PowerTools Pro.

NOTE

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

Update – Upload the values into the current Update PowerTools configuration.

When this option is selected the “Update” function will Update the NVM on every download to the drive. Upon download the Update function uploads the configured NVM from the drive and places the data into the PowerTools Pro configuration file. The software then downloads this newly updated file to the drive.

Keep – Remember the values, and restore them after the download.

This option was created to allow users to save the values that have been changed via HMI, PLC or internally in a program so long as they have been added to the NVM list. When this option is selected PowerTools Pro will poll the drive on download for all of the values that have been added to the NVM list. PowerTools Pro then stores these values into a temporary memory location and after the program download is complete PowerTools Pro reinstates these values to the parameters before the drive can be enabled.

Ask on each download.

This option was created for users who want control of whether they will overwrite or keep the NVM on download. When this option is selected, PowerTools Pro will display a pop-up window that gives the user the option to Overwrite, Update, or Keep as described above.

Upload Non-Volatile Memory (NVM) Section:

Always upload NVM

When this option is selected, PowerTools Pro will default on an upload to uploading all of the parameters that have been mapped to the NVM and updating the display of these parameters in PowerTools Pro.

Always bypass NVM upload

When this option is selected, PowerTools Pro will not upload the NVM and the values that were originally downloaded to the drive will be displayed in the PowerTools Pro configuration.

Ask on each upload

When this option is selected, PowerTools Pro will default to asking the user via a dialog box whether to upload the NVM or to bypass the NVM upload.

Motor DDF Tab

The Motor DDF tab controls which .ddf files PowerTools Pro will use when working in the configuration file. The standard ddf files contain a list of motors that can be purchased from Control Techniques and their parameters. This tab gives the advanced user the capability to create their own standard motor ddf file. Never save a newly created user standard ddf file as StdMotor.ddf because it will be overwritten with the next release of PowerTools Pro. The user’s ddf file name would be enter

into the Standard Motor DDF Name text box for the correct drive type, see below.

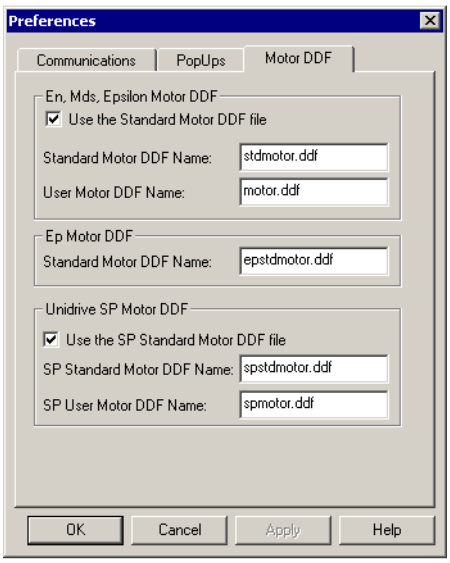


Figure 33: Preferences-Motor DDF Tab

En, MDS, Epsilon Motor DDF Section:

Use the standard Motor DDF file

Select this check box to use the standard motor ddf file.

Standard Motor DDF Name

The name of the standard motor ddf file is entered in the text box, the information in this file will be used to setup the configuration file, the default is stdmotor.ddf.

User Motor DDF Name

When a custom motor is created this is the name of the motor ddf file where the information is stored, default is motor.ddf.

Ep Motor DDF Section

Standard Motor DDF Name

The name of the standard motor ddf file for Epsilon EP drives.

Unidrive SP Motor DDF Section

Use the SP Standard Motor DDF file

Select this check box to use the standard motor ddf file with the Unidrive SP.

SP Standard Motor DDF Name

The name of the standard motor ddf file is entered in the text box, default is spstdmotor.ddf.

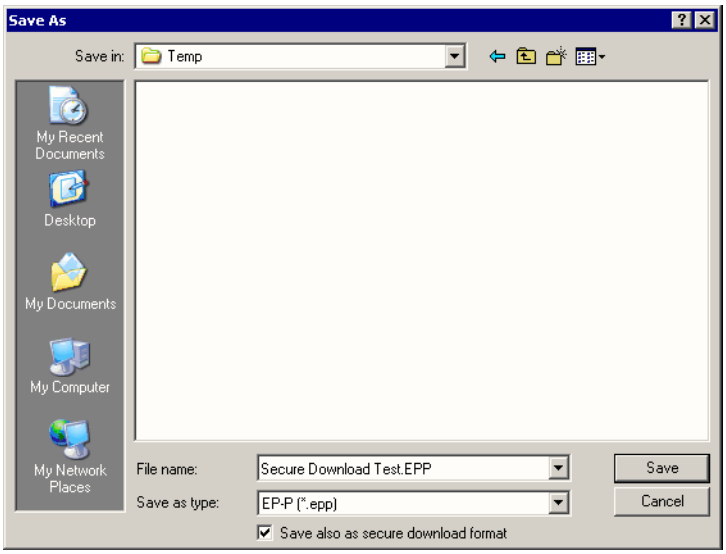
SP User Motor DDF Name

When a custom motor is created this is the name of the motor ddf file where the information is stored, default is spmotor.ddf.

2.7.8 Secure Downloading

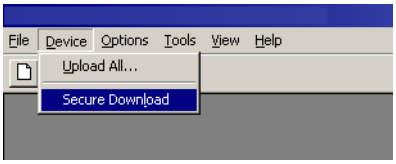
The Secure Download feature allows the user to download a configuration that prevents anyone from uploading the file, or going online with the system. This is used to protect a file from being accessed by unauthorized personnel. If a secure file is downloaded to the system, all diagnostics capabilities 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 Pro. Then on the **File** menu, click **SaveAs**. The following SaveAs dialog box should appear when saving an EP-P configuration file.



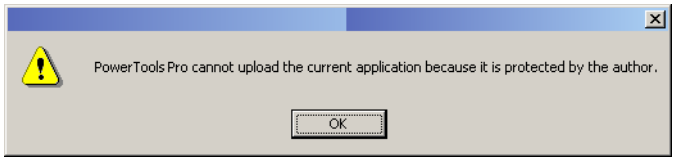
In this dialog box, select the “Save also as secure download format” check box located at the bottom of the dialog box, then click ***Save***. Doing this will save the file in BOTH the standard file format (.EPP), as well as in the secure file format (EPPs). The “s” at the end of the file extension stands for “secure”. 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 Pro. Then on the **Device** menu, click **Secure Download**, as shown below.



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

A secure file (EPPs) 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 residing in the drive has been protected by the user. An example of this is shown below.



2.8 Brake Operation

The brake is operational when the Brake.Disengaged parameter is wired to an output. It is expected that this output is then physically wired to the Motor's brake. The standard operation for brake is to engage the brake when the drive is disabled. The LED shows the little "b" when the brake is engaged. The Brake can further be controlled by the Brake controls Brake.Activate and Brake.Release. The Brake.Release overrides all conditions (drive disable and Brake.Activate) to insure the brake is disengaged. The Brake Activate parameter allows the user to engage the brake while the drive is enabled. However, you need to be careful because Brake.Activate does not stop motion commands

The table below shows the relationship between the Brake sources and destinations.

NOTE

No motion should be commanded while the brake is engaged.

Brake.Activate	Brake.Release	Drive Enable		Brake.Disengaged	Brake Status	Drive Diagnostic Display
Off	Off	Off		Off	Enabled	b
Off	Off	On		On	Disabled	
On	Off	Off		Off	Enabled	b
On	Off	On		Off	Enabled	
x	On	x		On	Disabled	

Brake Release

The Brake.Release destination function will release the brake under all conditions. When this function is active, the Brake output will be on (that is, release brake). This function overrides all other brake control, thus allowing the brake to be released while a fault is active or the power stage is disabled.

Brake Activate

The Brake.Activate destination function, when active, will engage the brake unless overridden by the Brake Release function. This function lets you externally engage the brake while allowing the drive to also control the brake during fault and disabled conditions.

Brake Disengaged

The Brake.Disengaged source function is used to control the motor holding brake. If the Brake function is off, the brake is mechanically engaged. When the brake is engaged, the diagnostic display on the front of the drive will display a “b”.

The drive outputs are limited to 150 mA capacity, therefore, a suppressed relay is required to control motor coil. Control Techniques offers a relay, model # BRM-1.

2.9 How Data Capture Works

Data Capture is a mechanism to capture data and display that data graphically. The capture mechanism is part of the drive and captures drive data as fast as 100 usec. Data is captured in a circular 8 K byte buffer. The format is fixed at 4 channels of 32 bit words for a total of 512 time samples. The circular buffer is continuously loaded until the trigger condition (or command abort) stops loading data. The capture mechanism follows three buffer states - Filling Buffer, Waiting for trigger, and Triggered.

At the start of the Run command, the buffer starts to fill (filling the whole buffer). The buffer must be completely filled before the trigger is armed. Once the buffer is filled, the buffer state will display - Waiting for Trigger. When the trigger is detected, the data capture is stopped (triggered). The sampling rate is based on the trajectory update rate. The sample rate can be adjusted in multiples of the trajectory update rate. PowerTools displays this in the form of seconds. At the update sampling, a new set of data is overwritten into the circular buffer and the trigger is checked.

The Data Capture update rate for the Epsilon EP-P drive is 100 usec. The drive captures data at the user selectable trajectory update rate of 800, 1200 or 1600 usec for Drive Inputs, Drive Outputs and Custom Variables. This means if the Data Capture rate is faster then the Epsilon EP-P drive trajectory update rate the user will be sampling data faster than it is changing. All other parameters in the Select list box are captured at 100 usec except Bus Voltage and Following Error which are 400 usec.

The trigger detection checks the data level. It does not specially look for an edge. Once the buffer is filled the trigger is armed and the check for trigger level is started. Since the drive is looking back in the buffer at data captured during the fill, the trigger condition may already exist. If that is so, the drive immediately transitions to the trigger state. If not, the drive continues the data capture cycle of sample and trigger check until the trigger is detected at the edge of the data transition. When the Trigger Falling Edge check box is selected the trigger is detected when the data transitions below the trigger level.

When sampling digital inputs and outputs, the data captured is binary bit mapped. The state of all the digital signals in the group selected are encoded into one 32 bit word. When this is graphed it is displayed as an analog signal. To trigger on this bit map data, the trigger mechanism is changed to a mask. The user can select one of the bits to trigger on.

The captured data is uploaded when the ***UploadPlot*** button is pressed. Once uploaded, PowerTools plots the data in graph window. Data is also saved in a data file named, PtProGraphData.csv. This data file can be exported to a spread sheet for data manipulation and graphing.

2.9.1 Navigating the Graph Window

The Graph window display can be altered, double-click anywhere in the Graph Window except on the graph area itself. The Customization dialog box opens and contains tabbed graph options. Many of the graphs attributes, such as colors, line format, etc. can be changed in this box. The graph can also be exported to a file.

Holding the shift key down while moving the mouse allows the user to zoom in on the graph area. Double-click on the graph area and the Graph Coordinate window opens and gives the x/y coordinate of where the mouse point was when double-clicked.

The Graph window overlaps the data into a Y axis if the next channel has the same units. If the units change for the next channel, a new graph on the Y axis is added to the plot. If None is selected for a channel the drive data capture samples zero for that channel an PowerTools ignores plotting that channel. The Reserved channel selection is for internal use only and also captures zero. The title of the graph matches the application’s name defined on the Setup view.

Graph settings are downloaded to the drive when the ***Run*** button is pressed. Only changed values are sent. The graph settings are the same as any application variable. When a variable is changed in a PowerTools view the ***Update to RAM*** button is available, indicating the application and drive are out of sync, (***Update to RAM*** remains unavailable if the user changes a variable that requires a reboot. The user then requires a full download). When the ***Run*** button is pressed, it does a limited Update to RAM by downloading the changed graph settings to RAM.

3 Setting Up Parameters

3.1 Status View

3.1.1 Status Online Tab

The Status Online tab (see Figure 34) is visible when online and consists of the Motor Position group, Motor Velocity group, Control Loop group, Master Feedback group, and the Torque group.

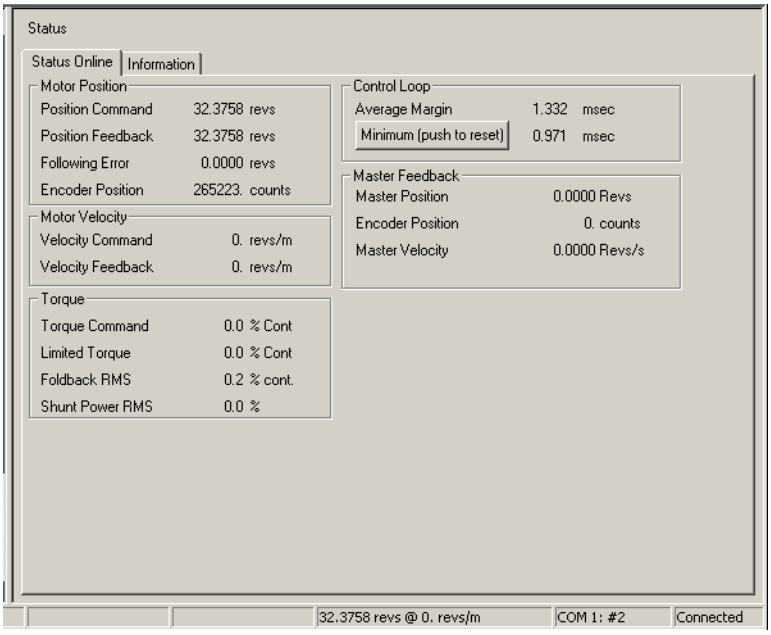


Figure 34: Status View - Status Online Tab

3.1.2 Motor Position Group

Position Command

Position command (PosnCommand) is the commanded position generated by the drive based on the application. This parameter does not take following error into account. See also Position Feedback and Following Error. Units are in user units.

Position Feedback

Feedback position (PosnFeedback) is the actual motor position in user units. PosnCommand minus the PosnFeedback is the FollowingError.

Following Error

Following Error (FollowingError) is the difference between the PosnCommand and the PosnFeedback. It is positive when the PosnCommand is greater than the PosnFeedback.

Encoder Position

Motor encoder position in encoder counts (PosnFeedbackInCounts). This position reflects the feedback position of the motor and is not scaled into user units. This is a signed 32 bit value.

3.1.3 Motor Velocity Group

Velocity Command

The Velocity Command (VelCommand) is the velocity that the drive is commanding the motor to run at. This command is generated by the drive velocity control loop and position loop. It is displayed in user units.

Velocity Feedback

The Velocity Feedback (VelFeedback) is the feedback (or actual) velocity. It is calculated using the change in position of the motor encoder. It will always return the actual motor velocity - even in synchronized applications in which the master axis is halted during a move.

3.1.4 Control Loop Group

Changing the Trajectory Update Rate can have a major effect on the performance of the servo system. A longer trajectory update rate means that more time is available to process user programs. A shorter update rate means that the control loop is updated more often and provides the most accurate performance. Without proper diagnostics, it can be impossible to tell how much time is being consumed by the control loop update, and how much time is available to run user programs.

The Control Loop group of parameters on the Status Online tab shows the user how much time is available to run programs. There are two parameters available to help with this. They are as follows:

Control Loop Limit

This parameter shows the lowest measured time difference (in microseconds) between the Trajectory Update Rate and the time taken to process the control loop since the last reset. Certain features in the Epsilon EP-P require more time to process (i.e. Real Time program execution, PLS, Capture, Compound Indexes), and therefore will cause lower limits. The software records the lowest measured value and displays it as the limit. To reset the limit to the average and continue tracking the lowest value, the user can click on the Limit button. If the Limit reaches 0, a fault will be generated. If a Limit of less then 75 - 100 usec is seen, it is recommended to switch the trajectory update rate to the next higher value.

The cyclic program is run as a high priority user program and is not part of the Control loop time.

Average Margin

This parameter shows a running average of the difference (in microseconds) between the Trajectory Update Rate and the time taken to process the control loop since the Status Online tab was brought up. The higher the value, the more time available to run user programs. For Averages less than 150 usec, it is recommended to switch the trajectory update rate to the next higher value.

3.1.5 Master Feedback Group

Master Position

Used for synchronized motion, this displays the position of the master encoder in the user units name, defined on the Master Units View.

Encoder Position

This displays the position of the master encoder in counts.

Master Velocity

This displays the velocity of the master encoder in master user units/second.

3.1.6 Torque Group

Torque Command

This displays the torque command value before it is limited. The torque command may be limited by the motors rating, the drives rating, the Torque Limit (if the Torque Limit Enable destination is active) or Current Foldback. Units for this parameter are defined in the Torque Group on the User Units View.

Limited Torque

This is the actual torque commanded to the motor. This value is the result after the TorqueCommand is limited by the current foldback or the TorqueLimit value (if enabled).

Foldback RMS

This parameter accurately models the thermal heating and cooling of the drive and motor. When it reaches 100 percent, current foldback will be activated. See the Diagnostics section for an explanation of foldback.

Shunt Power RMS

This parameter models the thermal heating and cooling of the drive internal shunt. This parameter indicates the percent of shunt capacity utilization. When this value reaches 100 percent, the drive will generate an RMS Shunt Power Fault.

3.1.7 Information Tab (Online Only)

PowerTools Pro is designed to work with all releases of drive firmware. PowerTools Pro internally uses interface revisions to negotiate and define the application to match the connected drive. The first line shows the file name and user application for reference.

3.1.8 Drive Information Group

Firmware Revision

Displays the revision of the drive firmware.

Serial Number

Displays the serial number of the drive.

The following interface revisions are PowerTools and the drive firmware internal revision numbers to identify parameter sets, data structures, data ranges and access attributes to match the drives’s data structures to PowerTools. When parameters are added to support new features or parameters are altered, a new interface revision is assigned.

Base Interface Revision

This parameter displays the interface revision of the data parameter structure in the Epp's base drive. This revision has a correlation with the features supported by the Epp drive.

Registry Interface Revision

This parameter displays the interface revision of the "Registry" data parameter structure in the Epp drive. This revision has a correlation with the features supported by the Epp. On download, both the Registry and Base Interface revisions are used to convert the Power Tools data set to match the drive's data structure. In this way Power Tools can support different releases of the EPP Flashes. For the most part the user does not need to be concerned over interface revisions.

FPGA Revision

The Field Programmable Date Array (FPGA) is an electronic component in the drives. The FPGA revision is only displayed because when the Epsilon EP drive was first released the FPGA was not field upgradable. When the revision is <100 the drive is not field upgradable. When the revision is >100 the FPGA can be upgraded by flash upgrading and then Virtual Master Sync Output will work.

3.2 Graph View

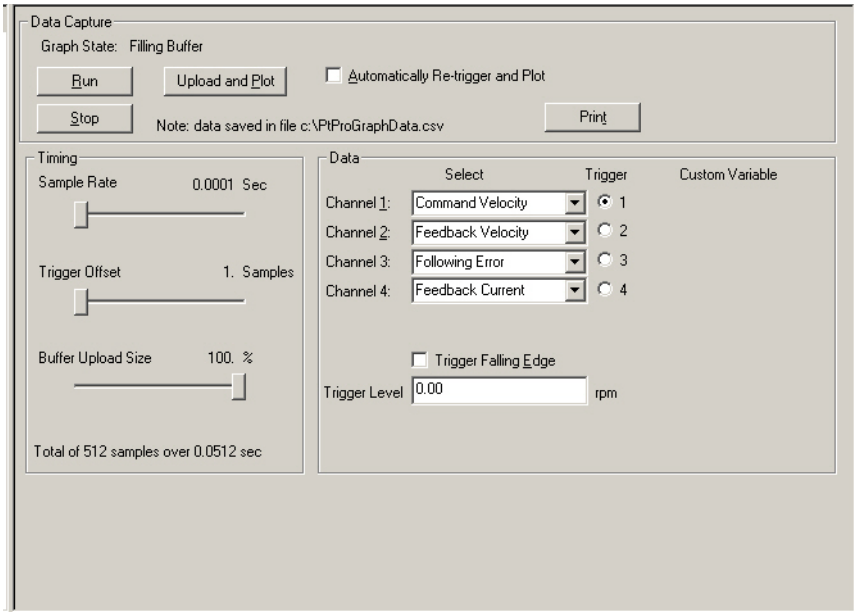


Figure 35: Graph View

3.2.1 Data Capture Group

Graph State

There are three 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.

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 the check box and the Automatically Re-trigger and Plot tells PowerTools Pro to 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. Initially 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 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.

3.2.2 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 desire 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 date. 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 corresponds to the number of samples that will be included on the graph display and data capture prior to the actual trigger. If the Trigger offset slider is completely to the left (min samples), the data capture and graphing will

start at the trigger location. If the slider is completely to the right (max samples) the graph will capture data until the trigger point.

Buffer Upload Size

The buffer upload size slider truncates the drive captured data. If the slider is completely to the right (max) the complete buffer will be uploaded. If the slider is completely to the left, only 1% of the buffer will be uploaded. This parameter does not effect the data capture size, it only defines how much of the buffer will be uploaded.

3.2.3 Data Group

Data Channel 1 - 4 Select List Boxes

The Channel 1 through Channel 4 list boxes give the user options for parameter display. If parameters with the same units are mapped on adjacent channels then the graphical display will show these two parameters overlapped on the same x/y axis. If it is desirable to have two adjacent Channels with the same units mapped to separate axis on the graph then the selection (none) should be used on the channel in between these two parameters.

Trigger Radio Buttons

Selecting the radio button will cause the graphical capture to trigger the capture 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 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 Sample Rate.

Custom Variable

A Custom Variable text box is only available once the user has selected Custom Variable from the Select list box. This field is used to define what parameter will be plotted on that channel. The custom variable parameter can be entered two ways: by just typing any custom variable 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 custom variable text box.

Trigger Mask List Box

This list box is only available when Drive Inputs or Drive Outputs are 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.

3.3 Setup View

The Setup View contains all of the primary system setup parameters. These parameters must be setup prior to using your system.

By selecting Setup in the Hierarchy Tree, the Setup view will appear on the right side of the view (see Figure 36). The Setup view is divided into six groups, with an explanation of each function. The groups are Identification, Configuration, Drive Encoder Output, Positive Direction, Update Rate and Switching Frequency.

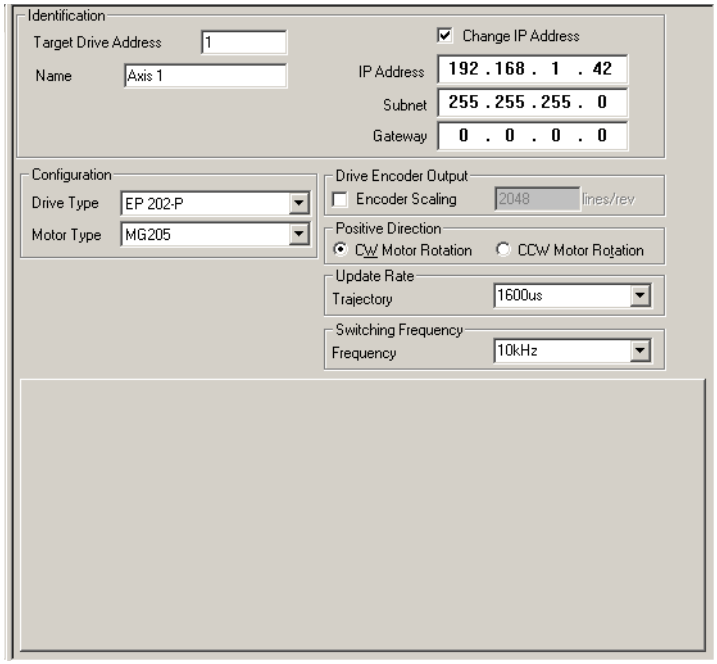


Figure 36: Setup View-Epsilon EP-P drive

3.3.1 Identification Group

The identification group consists of the Name, Target Drive Address, Change IP Address Check box, IP Address, Subnet and Gateway.

Name

This is a 12-character alpha/numeric user-configured name for this axis. Enter this name for the drive currently being set up. Assigning a unique name for each drive in the system allows the user to quickly identify a drive when downloading, editing, and troubleshooting. All keyboard characters are valid. This will default to Axis 1.

Target Drive Address

This parameter (Modbus.ModbusId) is the Modbus address of the target drive to which the user will download the configuration. The default target drive address is 1.

Change IP Address Check box

Change IP Address check box is used to determine if the user wants to use the scanner to determine the application's Ethernet address. If Change IP Address check box is selected then the scanner selected address becomes the new application address (saved with application). This applies for Download, Upload into an existing application and Change Path.

When Change IP Address check box is clear the Ethernet Address scanner range is the last scan range entered using the scanner's "Stop Scan". If Change IP Address is selected, then the scanner's "Ip Address Scan Range" range will be loaded to select only the applications IP address.

The Change IP Address check box must be selected to change the IP Address, Subnet and Gateway.

IP Address

This parameter (Ethernet.EthernetConfiguration.IPAddress) is a 32-bit identification number for each node on an 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.

Subnet

This 32-bit parameter (Ethernet.EthernetConfiguration.Subnet) indicates the subnet mask used for this node. The subnet mask is used to group drives that are connected on the same physical connection. For a detailed description of Subnet mask refer to the Industrial Ethernet Overview section in the *Epsilon EP Connectivity Reference Manual* (P/N 400518-04).

Gateway

This 32-bit parameter (Ethernet.EthernetConfiguration.Gateway) indicates the default Gateway address for the drive. When attempting to communicate with a drive on a different Subnet, the message must go through this gateway to reach it's destination. For a detailed description of the Gateway address refer to the Industrial Ethernet Overview section in the *Epsilon*

EP Connectivity Reference Manual (P/N 400518-04).

3.3.2 Configuration Group

The configuration group consists of list boxes for Drive Type and Motor Type.

Drive Type List Box

Select the drive model for the system you are currently setting up.

Motor Type List Box

Select the motor model for the application from the list of motors.

Motor types are arranged as follows:

- FM motors

  Motor voltage is determined by the first letter (E=230, U=460), brake motors have a 1 as the last digit (095E2A401).
- NT motors

  Standard 230 V motors, brake motors have BR after the model #.
- XV motors

  Standard 230 V motors, brake motors have BR after the model #.
- MG motors

  Standard 230 V motors, brake motors have BR after the model #.
- Unimotor Classic motors

  Model numbers end with a UL in the part number, for example: 115EZB200-UL.

  Motor voltage is determined by the first letter (E=230, U=460).

  For brake motors, use the standard motor model.

NOTE

Selecting the wrong motor type can cause poor performance and may even damage the motor and/or drive.

3.3.3 Drive Encoder Output Group

The drive encoder output group consists of the encoder scaling check box and encoder scaling.

Encoder Scaling Check Box

Select this check box to enable the Encoder Scaling parameter of the Drive Encoder Output.

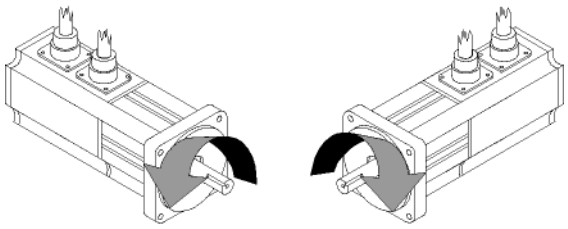
Encoder Scaling

This parameter defines the encoder resolution (lines per revolution) of the drive's encoder output. This feature allows you to change the drive encoder output resolution in increments of 1 line per revolution up to the density of the encoder in the motor. If the Encoder Output Scaling parameter is set to a value higher than the motor encoder density, the drive encoder output density will equal that of the motor encoder.

3.3.4 Positive Direction Group

The Positive Direction group consists of a CW (clockwise) Motor Rotation radio button or a CCW (counter-clockwise) Motor Rotation radio button.

The motion will move in either CW direction or CCW direction. 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).

3.3.5 Update Rate Group

Trajectory

This parameter configures the interrupt interval for the drive 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. After Control Loop is complete, all messages are handled. Messages are Modbus data, DeviceNet data,

Display information, and are only processed if a message is waiting. If no device is querying data from the Epsilon EP-P drive or sending data to the Epsilon EP-P drive, then messages do not take up any time. Once messages have been processed, the remainder of the interrupt is dedicated to running the motion programs of user programs.

Available selections for Trajectory Update are 800, 1200, and 1600 microseconds. The longer the update, the more time is dedicated to the user programs, and the less time dedicated to servo performance. The shorter the update, 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 drive to help select the ideal setting. (See description of Control Loop Group of online parameters on page 23 for further information)

3.3.6 Switching Frequency Group

This parameter defines the switching frequency of the drive. For the Epsilon EP drives, the switching frequency is 10 kHz and cannot be changed.

3.4 Motor View

The Motor view under Setup view is used for many different functions:

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

The primary function of this view is to define the parameters for the given motor that is to be connected to the drive.

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

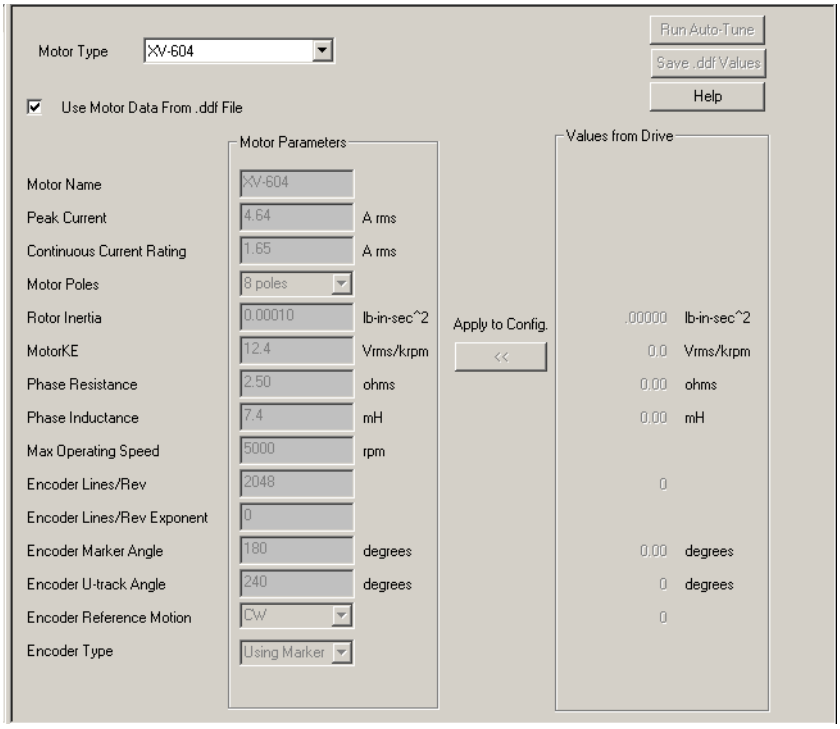


Figure 37: Motor View

3.4.1 Motor Type List Box

Use this list box to select the motor type. PowerTools Pro software will display all the standard motor models and any user defined motors.

CAUTION	Selecting the wrong motor type can cause instability and may cause property damage to the motor and/or drive.

3.4.2 Use Motor Data From .ddf File Check box

When selecting a motor for use with the Epsilon EP-P drive the user has two basic options:

1. Use a motor that already exists in the standard motor definition file (StdMotor.ddf) or custom motor definition file (Motor.ddf).

2. Create a custom motor that has not been used before.

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 Motor 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 view (see Figure 37). 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 “User 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 on the Save .ddf Values button on the right side of the view.

3.4.3 Motor Parameters Column

Motor Parameters column is a column of data displayed on the Motor view under the Setup view (See Figure 37). This column of data contains the values for each of the motor data parameters. The values in this column are unavailable for edit 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 there no longer is a link to the data in the .ddf file.

If the user does edit motor parameter values on this view, 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 view. Below are the motor parameter with a brief description.

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 above.

Peak Current

Specifies the peak current allowed by the motor. 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 Rating

Specifies the continuous current allowed by the motor. It is used to determine the drive continuous current and peak current limits. The drive can also limit the continuous current to the motor based on the drive capacity. 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 pole pairs (N-S) on the motor. The supported values are 2, 4, 6, 8, 10, 12, 14 and 16 poles. The motor manufacturer typically provides the motor pole information.

Rotor Inertia

This parameter specifies the inertia of the motor rotor. The drive uses this parameter to interpret the “Inertia Ratio” parameter. “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 5.0 to 500.0 Vrms/ kRPM. The motor manufacturer will typically provide the Ke data.

Phase Resistance

Specifies the phase-to-phase resistance of the motor. This value is determined by measuring the resistance between any two motor stator terminals with an ohm meter. The range is.1 to 50 ohms.

Phase Inductance

Specifies the phase-to-phase inductance of the motor.

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.

Encoder Lines/Rev

Specifies a coefficient for determining the number of encoder lines per mechanical revolution. The supported values are 1 to 16383. The equation for determining the total number of encoder lines per revolutions is:

$$nLines = n*10x$$

where:

nLines = Total number of Encoder Lines
n = Motor Encoder Lines per Rev Coefficient
x = Motor Encoder Exponent

The total number of encoder lines is used both for commutation and for position/velocity control. To properly commutate the motor, the drive

must know the electrical angle (the angle between the motor magnetic field and stator coils).

Encoder Lines/Rev Exponent

Specifies a coefficient for determining the number of encoder lines per mechanical revolution. The supported values are 1 to 16383. The equation for determining the total number of encoder lines per revolutions is:

nLines = n*10x

where:

nLines = Total number of Encoder Lines
n = Motor Encoder Lines per Rev Coefficient
x = Motor Encoder Exponent

The total number of encoder lines is used both for commutation and for position/velocity control. To properly commutate the motor, the drive must know the electrical angle (the angle between the motor magnetic field and stator coils).

Encoder Marker Angle

Specifies the electrical angle at which the marker (Z) pulse occurs with reference to V_{TS} when the motor is spun in the encoder reference direction. At power-up the drive obtains an initial estimate of the electrical angle from the status of the U, V and W commutation tracks. This estimate can be off by as much as 30 °.

When the drive receives the marker pulse, the drive will, within one second, gradually shift the commutation to the more accurate electrical angle specified by this parameter. The system will then operate more efficiently.

Encoder U-track Angle

Specifies the electrical angle at which the rising edge of the U commutation track will occur with reference to VTS when the motor is spun in the encoder reference direction.

At power-up the drive looks at the status of the U, V and W commutation tracks and, using this parameter, obtains a crude (± 30 °) estimate of the electrical angle.

Encoder Reference Motion

Specifies the direction of motion assumed in phase plots of the encoder's quadrature and summation signals. The supported values are CW(1) and CCW(0). Your encoder may have the same phase plot but is generated from a different direction of rotation. This parameter affects the way the drive interprets the quadrature and commutation signals.

Encoder Type

The supported values for this parameter are 1 and 0. If set to a 1 the drive uses the Encoder Marker angle as well as the Encoder U Angle for commutation. If this parameter is set to a 0, the drive uses only the Encoder U Angle.

3.4.4 Run Auto-Tune Button

The drive has 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 Pro allows the user to initiate the Auto-Tune feature from the Motor view.

The following table shows which parameters must be entered in order to run the Auto-Tune feature, and which parameters are measured by the Auto-Tune.

Motor Parameters	Needed to Run Auto-Tune	Measured by Auto-Tune Mode #
Motor Name		
Peak Current	Yes	
Continuous Current Rating	Yes	
Motor Poles	Yes	
Rotor Inertia		3
Motor Ke		3
Phase Resistance		2,3
Phase Inductance		2,3
Max Operation Speed	Yes	
Encoder Lines/Rev	Yes	1,2,3
Encoder Lines/Rev Exponent	Yes	1,2,3
Encoder Marker Angle		1,2,3
Encoder U-Marker		1,2,3
Encoder Reference Motion		1,2,3
Encoder Type		

All Auto-Tune modes cause motion. It is important to read and understand the warnings and instructions on the Auto-Tune windows. It is strongly recommended to unload the motor before performing an Auto-Tune.

When online with the drive, to initiate an Auto-Tune, click ***RunAuto-Tune*** button. The Auto-Tune dialog box opens and contains warnings and instructions related to the Auto-Tune procedure, as well as selection of the Auto-Tune mode. An example of one of the Auto-Tune windows is shown in Figure 38.

Figure 38: Auto-Tune Dialog Box - Auto-Tune Mode 3

After the Auto-Tune Mode has been selected, click ***Proceed***, to start the Auto-Tune. As the Auto-Tune is executing the gray check boxes will show the progress of the Auto-Tune. When the Auto-Tune is completed the Auto-Tune dialog box closes and the auto-tune results will be in the Values from Drive column on the Motor view.

Auto-Tune Mode 1

Auto-Tune Mode 1 provides the encoder information of the motor. Auto-Tune Mode 1 is the most beneficial tuning by providing parameter values that are difficult to measure or vary the most from common motor specifications. Auto-Tune Mode 1 will rotate the motor for 1 or more revolutions at a slow speed.

With Auto-Mode 1 the test sequence is Moves 1 through 3 consisting of:

Move #1, Vel = Max Motor Vel, Distance < (1 / # of motor poles) motor revs, Either Direction

Move #2, Vel = Max Motor Vel, Distance < (1 / # of motor poles) motor revs, Either Direction

Move #3, Vel = 0.15 Electrical Cycles/sec (approx 1 rev/min), Distance < 1.0 Revolutions of motor, Direction = CW

The parameters measured in Auto-Tune Mode 1 are Encoder Marker Angle, U-track Angle, Encoder Reference Motion and Encoder Resolution.

As Auto-Tune Mode 1 is executed the test is checked off, Phase 1 Encoder Marker Angle, U Angle, Motion.Resolution

Auto-Tune Mode 2

Auto Tune Mode 2 provides and method to determine the motor resistance and inductance. These are used in the current control loop. It is more accurate to make these measurements with external meters. If you do not have measured motor resistance and inductance, you can use auto tune 2.

Auto Mode 2 is a subset of auto mode 3 running Moves 1 through 3:

Move #1, Vel = Max Motor Vel, Distance < (1 / # of motor poles) motor revs, Either Direction

Move #2, Vel = Max Motor Vel, Distance < (1 / # of motor poles) motor revs, Either Direction

Move #3, Vel = 0.15 Electrical Cycles/sec (approx 1 rev/min), Distance < 1.0 Revolutions of motor, Direction = CW

The parameters measured by Auto-Tune Mode 2 are those tested in Auto-Tune Mode 1 plus Phase Resistance and Phase Inductance.

As Auto-Tune Mode 2 is executed, the completed portion of the test is checked off,

Phase 1 Encoder Marker Angle, U Angle, Motion.Resolution

Phase 2 Voltage for Phase 3

Phase 3 Resistance and Inductance

Auto-Tune Mode 3

Auto Tune Mode 3 provides a method to determine the motor Ke and Rotor Inertia. These are used in the velocity control loop. It is more accurate to use the Motor Ke and Rotor Inertia from the manufacturer. If you do not have manufacturer provided motor Ke and Rotor Inertia specifications, you can run an Auto-Tune Mode 3.

WARNING	Auto-Tune Mode 3 runs the motor at 75% max motor velocity. If the load is connected this may cause harm to the system or personnel. It is strongly recommended to unload the motor before performing any Auto-Tune Mode.

Auto-Tune Mode 3 is the full auto-tune running Moves 1 through 5:

Move #1, Vel = Max Motor Vel, Distance < (1 / # of motor poles) motor revs, Either Direction

Move #2, Vel = Max Motor Vel, Distance < (1 / # of motor poles) motor revs, Either Direction

Move #3, Vel = 0.15 Electrical Cycles/sec (approx 1 rev/min), Distance < 1.0 Revolutions of motor, Direction = CW

Move #4, vel < 0.75 max motor velocity, Time = 4 sec, Direction = CW

Move #5 - #12, vel < 0.75 max motor velocity, Distance = 3 Revolutions of the motor, Direction = CW

As Auto-Tune Mode 3 is executing the completed portion of the test is checked off,

Phase 1 Encoder Marker Angle, U Angle, Motion.Resolution

Phase 2 Voltage for Phase 3

Phase 3 Resistance and Inductance

Phase 4 Motor Ke

Phase 5 Rotor Inertia

3.4.5 Values from Drive Column

The Values from Drive column shows the new motor parameters as a result of an Auto-Tune. These parameters are continually being read from the drive. The theory of operation is that the user will perform an Auto-Tune function that reads/measures/calculates the motor parameters within the drive. The results are read from the drive and displayed in the Values from Drive column. Once they are displayed in PowerTools Pro (in the Values From Drive column) the user can apply those values to the Motor Parameters column by clicking on the Apply to Config. button, in the middle of the Motor view (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 drive. When offline, the values in the Values from Drive column will all read zero.

3.4.6 Apply to Config. Button

After any Auto-Tune has been performed, the results are displayed in the Values from Drive column on the Motor view and the measured parameter values are saved in the drives NVM. When the user clicks ***Apply to Config.***, the values in the Values From Drive column are transferred into the Motor Parameters column. The configuration file then can be saved to disk.

Therefore, in order to store the motor parameter values in the drives flash memory the configuration file must be downloaded to the drive using **Device** > **Download**.

3.4.7 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 Motor.ddf file so it can be easily recalled at a later time. If the user does not save the motor data to the 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 Motor.ddf file, click the ***Save .ddf Values*** button. This takes all the parameter values and writes them to the Motor.ddf file.

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

User Defined Motor Name Conflict Dialog Box

The purpose of this dialog box is to resolve conflicts between the application’s motor settings and those defined in the .ddf file.

The User Defined Motor Name Conflict dialog box opens during the following conditions:

1. From the Motor view, click the Save .ddf values button and the motor already exists with the same name but has different motor parameters
2. Opening an application (or uploading a application), where the Use Motor data from the .ddf file check box is select but the data in the application no longer matches the .ddf file.
   This occasionally occurs when a newer version of PowerTools Pro is installed and the parameters for the standard motors has been updated in the .ddf file.

If the motor name does not exist in the .ddf file, it will be written into the file.

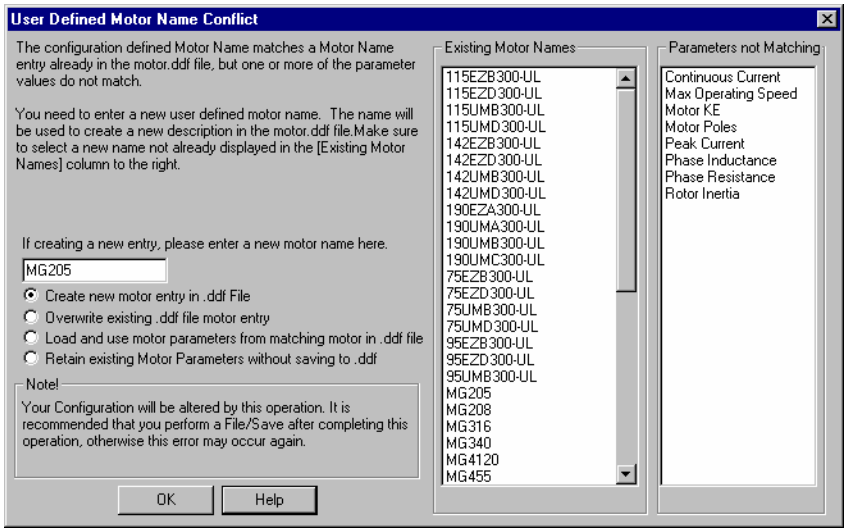


Figure 39: The User Defined Motor Name Conflict Dialog Box

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:

Create new motor entry In .ddf File

The user can select to keep the existing data and create a new entry into the motor.ddf file with a different name. After selecting this option, the user simply enters a new name in the Please enter a new motor name text box. Then click OK, the data will be written to the .ddf file using the new motor name.

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 the overwritten data will be lost forever. The overwritten data cannot be recovered.

If the user attempts to overwrite data for a Standard Motor (in the stdmotor.ddf file), the operation will be canceled and the user will be notified that they cannot proceed. The figure below 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.

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

If this option is selected, the motor data in the Motor.ddf or stdmotor.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.

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 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 Box

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 motor.ddf and stdmotor.ddf files. When selecting a new name, it is important to select a name that is not already displayed in this list box.

Parameters Not Matching List

This list 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 motor.ddf or stdmotor.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.

3.5 User Units View

The User Units View is used to scale the desired application units into known values. All information for distance, velocity, and accel/decel units are set up here and used throughout the system setup.

By selecting User Units in the Hierarchy Tree, the User Units view will appear on the right (see Figure 40).

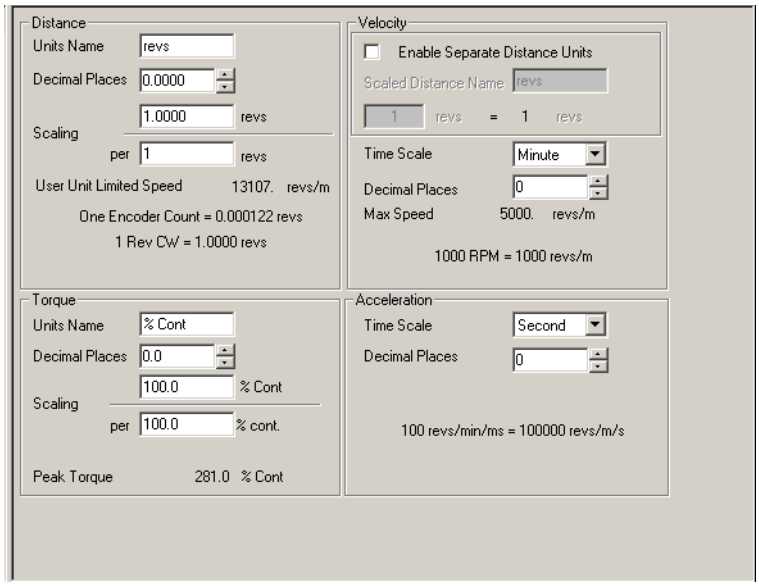


Figure 40: User Units View

3.5.1 Distance Group

Units Name

This is a 10-character name for the distance user units the user wants to use in the 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 configuration. 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.

NOTE

When the number of decimal places are changed in an existing configuration file the Index accel and decel parameters need to be checked.

Scaling

A Characteristic Distance and Length must be established to allow the drive to scale user units back to actual motor revolutions. This scaling factor is as follows:

$$\textbf{Scaling} = \frac{\textbf{Characteristic Distance}}{\textbf{Characteristic Length}}$$

Characteristic Distance

This is the distance the load travels (in user units) when the motor travels the characteristic length (in motor revolutions).

Characteristic Length

This is the distance the motor travels (in whole number of revolutions) to achieve one characteristic distance of load travel.

Distance Scaling Examples:

A 1.5" diameter pulley is used to drive a conveyor belt, and the user wishes to use units of inches instead of revolutions.

Units Name — Set to Inches

Decimal Places — Set to desired accuracy 0.000

In one revolution of the motor (or pulley), the belt will travel a distance of one pulley circumference.

= 1.5" x π

= 1.5 x 3.14...

= 4.712 inches / revolution

$$\textbf{Scaling} = \frac{\textbf{Characteristic Distance = 4.712}}{\textbf{Characteristic Length = 1}}$$

If the user decides to put a 5:1 reducer on the system, the user simply needs to change the Characteristic Length.

Now the belt travels 4.71" in 5 motor revolutions.

$$\textbf{Scaling} = \frac{\textbf{Characteristic Distance = 4.712}}{\textbf{Characteristic Length = 5}}$$

Keep in mind that the characteristic length is always a whole number and the valid range is from 1 to 2000.

NOTE

User Units may affect end motor speed and could cause trajectory faults.

Because of internal math in the Epsilon EP-P drive, some user unit combinations may cause a drive trajectory faults. The maximum motor velocity allowed by the drive is detailed under the distance section of the User Units View and is labeled “User Unit Limited Speed”. When the user unit setup is altered in such a way that the maximum motor speed allowed by the drive is less than the maximum speed allowed by the chosen motor, the readout of maximum motor speed allowed by the drive changes to have a red background. If a configuration is downloaded to the drive with a red background on the “User Unit Limited Speed”, the drive will obtain a trajectory fault at speeds near this velocity. To alleviate this issue, simply remove decimal places from your user units, and/or change the characteristic distance (numerator) of your scaling parameters to be a smaller number that it was. The red background indicating trajectory faults will go away when the user unit setup is scaled for a realistic accuracy based on the encoder counts per revolution.

3.5.2 Velocity Group

Enable Separate Distance Units Check Box

If selected (enabled), separate distance and velocity units, name and scaling will be enabled. If not enabled, the velocity units, name and scaling will be defined by the Distance Group.

Scaled Distance Name

If the user wants the velocity units to have a different distance scaling than the distance units a name can be entered here up to 10 characters. For example, the user distance units name could be inches while the velocity units name is feet per minute.

Velocity Distance Units Scale Factor

This parameter scales the Velocity Distance Units back to actual distance units. To do this, enter the number of distance user units that are equal to one velocity scaled distance unit.

Separate Distance Units Example:

A user has an application using a leadscrew with a 0.5"/turn lead. The user wants to have Distance Units of Inches, but wants Velocity Units of Feet so motion can be programmed in feet/minute.

Distance Units Name — Inches

Enable Separate Distance Units — Select check box (enabled)

Scaled Distance Name — Feet

Velocity Distance Units Scale Factor — # of Distance Units / 1 Scaled Distance Unit

1 Foot = 12 Inches

Velocity Distance Units Scale Factor = 12

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 set in this parameter determines the number of digits after the decimal point used in all real-time velocity parameters throughout the software. Using a high number of decimal places will improve velocity resolution, but will also limit the maximum velocity. You can select from zero to six decimal places of programming resolution.

3.5.3 Acceleration Group

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 set in this parameter determines the number of digits after the decimal point used in all real-time accel/decel parameters throughout the software. Using a high number of decimal places will improve accel/decel resolution, but will also limit the maximum accel/decel rate. You can select from zero to six decimal places of programming resolution.

3.5.4 Torque Group

Units Name

10-character name for the torque user units.

Decimal Places

The number of decimal places set in this parameter determines the number of digits after the decimal point used in all torque parameters throughout the software. Using a high number of decimal places will improve torque resolution, but will also limit the maximum torque. You can select from zero to six decimal places of programming resolution.

Scaling

The amount of torque in user torque units will be set equal to the Percent Continuous Current. This parameter is used to scale the actual

torque back into the user defined units. The units of this parameter are % ContinuousCurrent. This scaling factor is used along with the user torque to establish a relationship between torque user units and actual torque.

3.6 Dual Loop View

Dual Loop View provides the setup parameters to have an axis position controlled by a remote or secondary encoder, while maintaining a velocity encoder feedback to help with loop stability and to prevent the axis “runa way” if the remote encoder becomes disconnected.

An example application would be where a web is pulled through rollers, but there is not a precise gripon the material so it can slip a little and so loose position. This is solved by an external encoder touching the web and providing the accurated feedback of the web position.

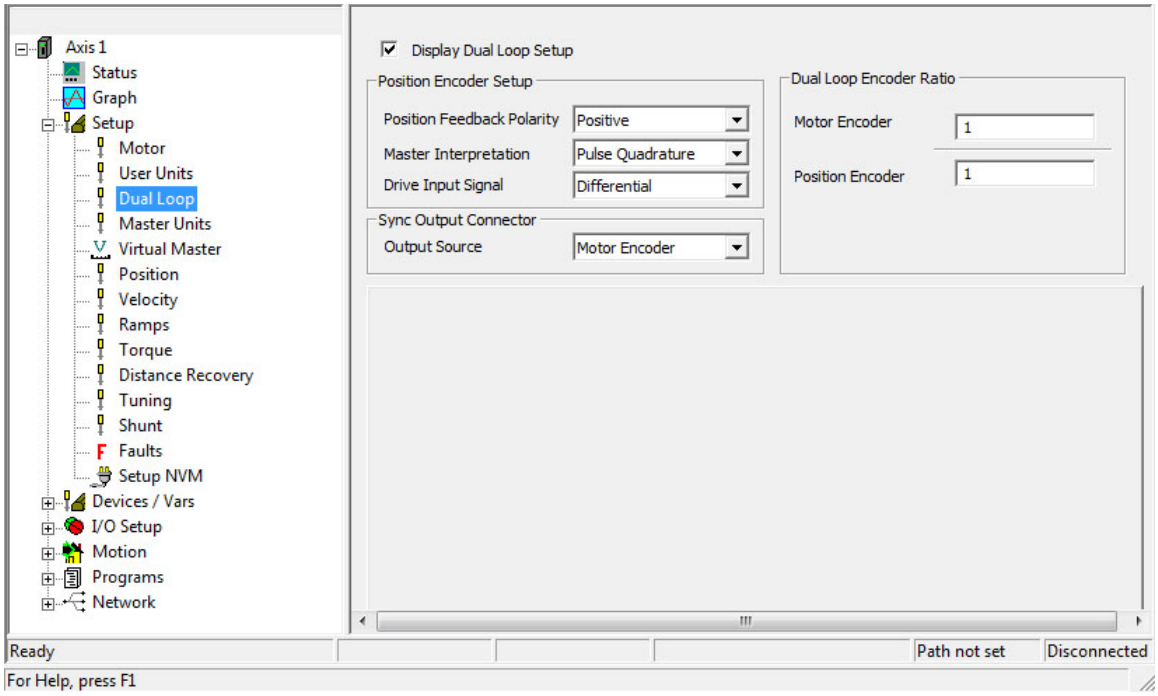


Figure 41: Dual Loop View

3.6.1 Display Dual Loop Setup

Select this check box to enable the Dual Loop Mode feature which accepts a secondaryfeedback device to close the position loop while still using the motor encoder to close the velocity loop.

3.6.2 Position Encoder Setup Group

Position Feedback Polarity

Position Feedback Polarity defines the direction of the position encoder that corresponds to a positive position change. This value is somewhat arbitrary, since the position encoder can be mounted to spin CW or CCW as the material moves "forward". If this value is not set correctly in Dual Loop mode the motor will rotate in the opposite direction as desired.

Master Interpertation

This parameter determines how the incoming pulses are seen to generate the synchronized motion command. This setting allows the user to choose the appropriate signal type to match the device generating the master input pulses.

Pulse Quadrature

The most common type for use with incremental encoders using AB Quadrature signals

Pulse Direction

Used with Step and Direction control signals

Pulse Pulse

Used with Pulse Forward and Pulse Reverse control signals

Available when Sync Encoder Input is selected as the Master Source on the MasterUnits View or on the Dual Loop View when the Dual Loop setup checkbox is selected enabling the feature.

Drive Input Signal

This is selected accordingly based on whether the incoming pulses are Differential (default) or Single Ended. Available when Sync Encoder Input is selected as the Master Source on the Master Units view or when the Dual Loop feature is enabled.

3.6.3 Sync Output Connector Group

Output Source

This determines which signal is sent out the Sync Output connector on the drive. If Motor Encoder (default) is selected, the encoder signals from the motor that the drive is controlling will be sent out the 15-pin Analog/Sync Output connector (J5). If Drive Encoder Input is selected, then the signals coming in the drives 9-pin Sync Input connector on the drive will be sent out the Analog/Sync. Output connector. When Virtual Master is selected the virtual master signal generated by the drive will be sent out the Analog/Sync. Output connector.

3.6.4 Dual Loop Encoder Ration

$$\textbf{Dual Loop Encoder Ratio} = \frac{\textbf{Motor Encoder}}{\textbf{Position Encoder}}$$

Motor Encoder

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.

Position Encoder

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.

3.7 Master Units View

Master Units View provides the setup parameters for use with synchronized motion. This setup window determines how the encoder signals are interpreted and establishes the scaling for all master units (master distance, master velocity, etc.).

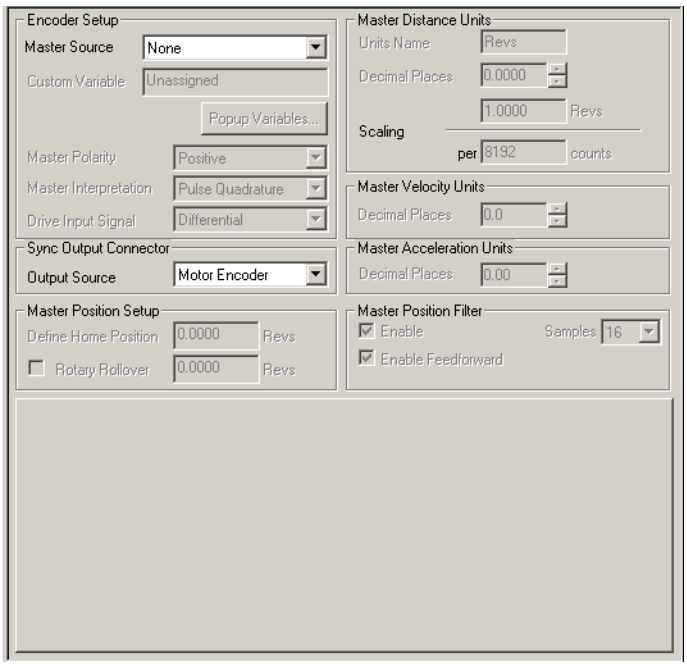


Figure 42: Master Units View

3.7.1 Encoder Setup Group

Master Source

Master Source (MasterAxis.Source) indicates the source of the master encoder input.

Select Custom Variable, the Custom Variable text box will became available to enter a parameter. The custom variable can be entered two ways: by just typing any 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 Custom Variable text box.

Select Sync Encoder Input to use the Sync Input connector (J10) on the Epsilon EP drive.

Select Virtual Master.Counts to use the Virtual Master as the Master Source.

Custom Variable

The Custom Variable (MasterAxis.CustomSourceVariable) text box is only available once the user has selected Custom Variable from the Master Source list box. This field is used to define what parameter will be used as the Master Source. The Custom Variable can be entered two ways: by just typing any parameter 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 Custom Variable text box.

Master Polarity

Master Polarity (MasterAxis.Polarity) defines the direction of the master encoder that corresponds to a positive master position change.

Master Interpretation

Master Interpretation (MasterAxis.Interpetation) determines how the incoming pulses are seen to generate the synchronized motion command. This setting allows the user to choose the appropriate signal type to match the device generating the master input pulses.

Drive Input Signal

Drive Input Signal is selected based on whether the incoming pulses are Differential (default) or Single Ended. Available only when Sync Encoder Input is selected as the Master Source.

3.7.2 Sync Output Connector Group

Output Source

Output Source determines which signal will be sent to the 15-pin Analog/Sync Output connector on the drive. If Motor Encoder (default) is selected, then the encoder signals from the motor that the drive is controlling will be sent out the Analog/Sync Output connector. If Drive Encoder Input is selected, then the synchronization signals sent to the drives 9-pin Sync Input connector will be sent to the drives Analog/Sync Output connector. When Virtual Master is selected the Virtual Master that was setup on the Virtual Master view is sent to the Analog/Sync Output connector.

3.7.3 Master Position Setup Group

Define Home Position

Define Home Position (MasterAxis.DefineHomePosn) 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, (MasterAxis.RotaryRolloverEnable=on)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 (MasterAxis.RotaryRolloverPosn) 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.

3.7.4 Master Distance Units Group

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.

Scaling

When Sync Encoder Input is selected as the Master Source then the scaling factor is defined as MasterAxis.CharacteristicDistance/MasterAxis.Counts. The numerator (top value of the scaling fraction) is the Characteristic Distance. The denominator (bottom value of the scaling fraction) is the # of Counts. The Characteristic Distance is the number of Master Distance Units that will be traveled per number of counts in the bottom of the fraction. The Counts parameter is the number of incoming pulses it takes to travel the characteristic distance.

3.7.5 Master Velocity Units Group

Decimal Places

Decimal Places determines the number of decimal places to be used in the velocity parameter for all synchronized motion.

3.7.6 Master Acceleration Units Group

Decimal Places

The number of decimal places set in this parameter determines the number of digits after the decimal point used in all real-time accel/decel parameters used for synchronized motion throughout the software. Using a high number of decimal places will improve accel/decel resolution, but will also limit the maximum accel/decel rate. You can select from zero to six decimal places of programming resolution.

3.7.7 Master Position Filter Group

The master position filter is designed for applications where the master encoder input requires smoothing due to low resolution or high gain. These applications include low speed masters, low resolution master encoders, and large follower to master gear ratios.

Filters inherently introduce phase shift (or delay) in the followers response to the master position, velocity and acceleration.

The drive provides Feedforward compensation to correct for these delays introduced by the filter.

The user may set the number of filter samples to be used to “smooth” the master encoder velocity. The more samples used by the filter, the smoother the master velocity signal, however, the more positional delay introduced by the filter. This means that more filtering will cause more position error between master and follower. Feedforward is used in conjunction with the filter to provide accurate positioning performance while still maintaining smooth motion.

The following table can be used to best determine the proper filter settings for your application.

		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

Filter parameters cannot be changed using the “Update to RAM” feature. Changes must be fully downloaded before taking effect.

The gray box in the table above denotes the default setting for the master filter parameters.

Enable Check Box

The Enable check box is used to turn on or turn off the Master Position Filter. If selected, the filter is turned on (active) and the user must select the number of samples used by the filter. If clear, the filter is not used.

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 above to select proper setting.

Enable Feedforward Check Box

The Enable Feedforward check box is used to turn on or turn off feedforward. If selected, feedforward is active. If the check box is clear, feedforward is not used.

3.8 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.

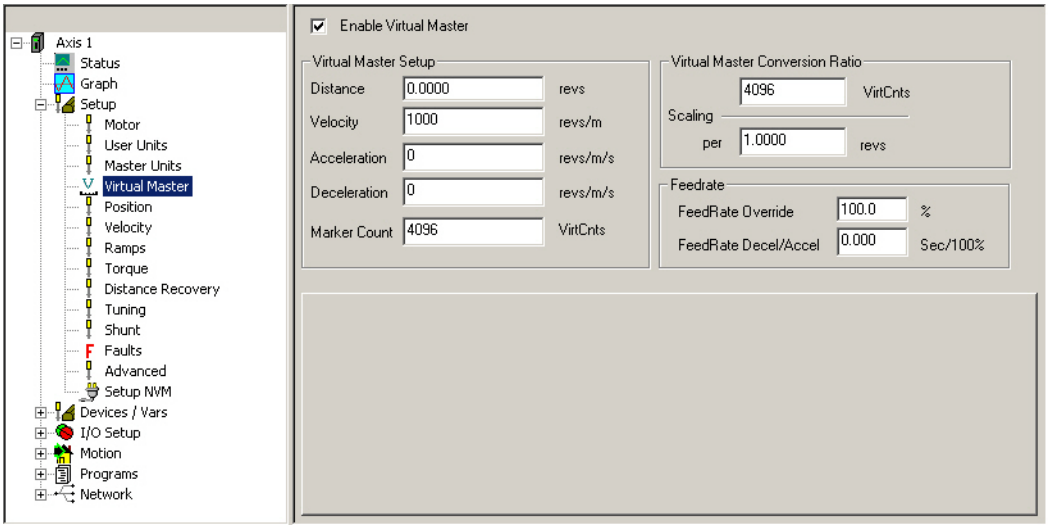


Figure 43: Virtual Master View Enabled

3.8.1 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.

3.8.2 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.

Marker Count

The encoder marker pulses are simulated and this parameter (VirtualMaster.MarkerCount) specifies the number of VirtualMaster.Counts per encoder marker pulse generated.

3.8.3 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.

Example:

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.

3.8.4 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.

3.9 Position View

The Position View allows you to set up and view the parameters related to drive positioning. In Figure 44, Position has been selected in the Hierarchy Tree. The right side of the view is divided into groups. An explanation of the groups and their functions is provided below.

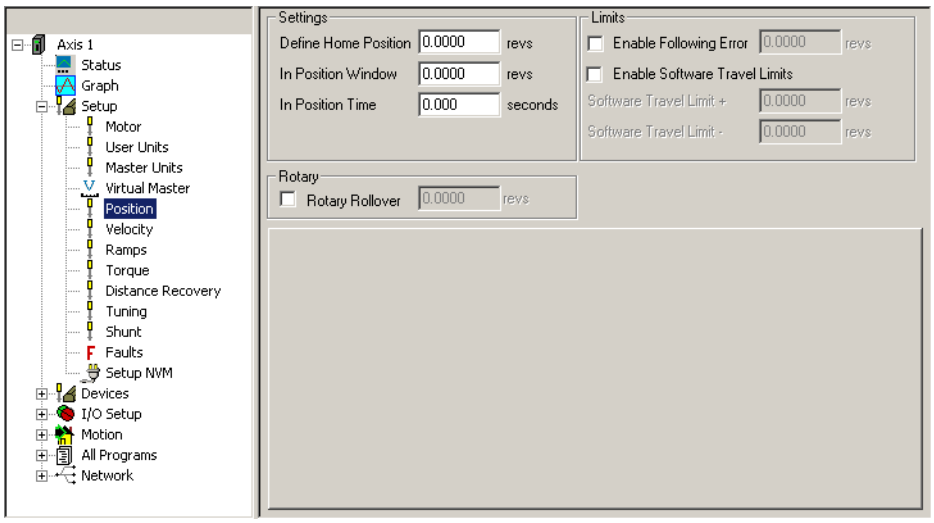


Figure 44: Position View

3.9.1 Settings Group

Define Home Position

Define Home Position (DefineHomePosn) is the value to which the Position Command will be set when the DefineHome 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

The In Position (InPosn) source will activate at the end of a move if the absolute value of following error is less than or equal to the In Position Window (InPosnWindow) for the In Position Time (InPosnTime).

In Position Window (InPosnWindow)

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.

Or the In Position window is set to 0.001 inches. If at the end of an index, the following error is calculated to be.0015 inches, then the InPosn source will not activate.

In Position Time (InPosnTime)

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 value.

3.9.2 Limits Group

Enable Following Error Check Box

Select this check box (FollowingErrorEnable) to enable (or disable if the check box is 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

Following Error (FollowingError) 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 (F). 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 Check Box

Select this check box (SoftwareTravelLimitEnable) to enable (or disable if clear) the software travel limits. If disabled, the software travel limits are not monitored.

Software Travel Limits

Software Travel limits can be used to limit machine travel. They are often setup inside the hardware travel limits to add another level of security or protection from exceeding the machines travel limits. The Epsilon EP-P drive constantly monitor the feedback position, and if this position exceeds the values entered for Software Travel Limit + or -, then the drive will decel to a stop. Software Travel Limits are not functional unless the Absolute PosnValid source is active. AbsolutePosnValid is active upon successful completion of a home or the DefineHome destination is activated.

To recover from a software travel limit, a jog may be commanded in the opposite direction of travel. For example, if a software travel limit - is

hit, then the axis can be jogged in the + direction.

Software Travel Limit + (SoftwareTravelLimitPlusPosn)

If the absolute position is greater than or equal to this value the SoftwareTravelLimitPlusActive source shall activate.

A rising edge occurs when the absolute position is greater than or equal to the parameter SoftwareTravelLimitPlusPosn. A falling edge will be generated as soon as the above is not true.

Software Travel Limit - (SoftwareTravelLimitMinusPosn)

If the absolute position is less than or equal to this value the SoftwareTravelLimitMinusActivate shall activate.

A rising edge occurs when the absolute position is less than or equal to the parameter SoftwareTravelLimitMinusPosn. A falling edge will be generated as soon as the above is not true.

3.9.3 Rotary Group

Rotary Rollover Check Box

Select this check box (RotaryRolloverEnable) to enable (or disable if clear) the rotary rollover feature.

Rotary Rollover

This parameter (RotaryRolloverPosn) 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 travelling 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 Pro software.

If an absolute index is used with a non-zero rotary rollover point, the Epsilon EP-P drive 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.

3.9.4 Online Tab (not shown)

While online, the following real-time data will be displayed.

Motor Position Group

Position Command

This is the commanded position in user units.

Position Feedback

This is the feedback position of the motor in user units.

Following Error

The Following Error is the difference (in user units) between the Position Command and the Position Feedback. It is positive when the Position Command is greater than the Position Feedback.

Encoder Position

The motor position in encoder counts since power up when the value was set to zero. This is a signed 32-bit value.

3.10 Velocity View

The Velocity View allows the setup of feedrate override details.

By selecting Velocity in the Hierarchy Tree, the Velocity View will appear on the right (see Figure 45).

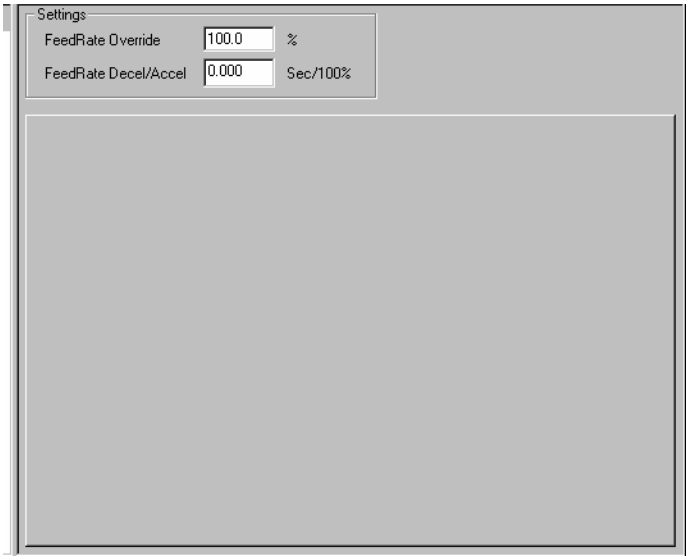


Figure 45: Velocity View

3.10.1 Settings Group

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 Modbus or in a program.

FeedRate Decel/Accel

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.

Example:

Feedrate Override is set to 100% (default). The user wishes to slow down motion to 50% of programmed velocity. If FeedRate Decel/Accel is set to 1 Sec/100%, when the FeedRate Override parameter is changed to 50%, it will take 0.5 seconds to decelerate to 50% velocity.

Decel/Accel Time = FeedRate Decel/Accel * % Change in FeedRate
= (1 Sec/100%) * (100% - 50%)
= 0.5 Seconds

Example:

A user wishes to accelerate from 100% programmed velocity to 175% in 0.5 Seconds. Therefore, the value they need to enter for Feedrate Decel/Accel is calculated as follows:

FeedRate Decel/Accel	=	Decel Time/ % Change in FeedRate
	=	(0.5 Sec) / (175% - 100%)
	=	0.5 Sec / 75%
	=	(0.5 Sec) / (100% * 75%)
	=	0.66 Sec / 100%

3.10.2 Online Tab (not shown)

If online, the following real-time data will be displayed.

Motor Velocity Group

Velocity Command

The Velocity Command is the actual command generated by the drive to the motor in user units.

Velocity Feedback

This parameter is the actual feedback motor velocity in user units.

3.11 Ramps View

The Ramps View allows the user to define various accel/decel ramps used under typical application conditions. By selecting Ramps in the Hierarchy Tree, the Ramps View will appear on the right (see Figure 46).

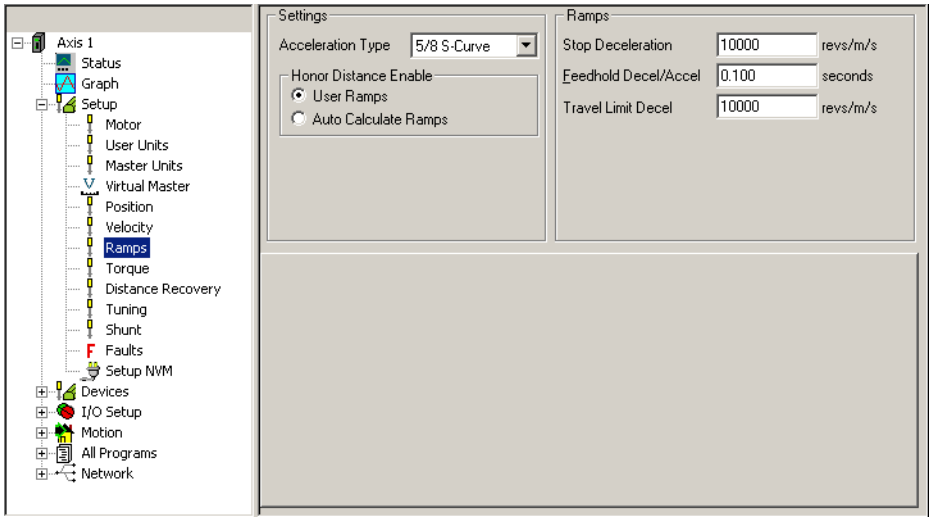


Figure 46: Ramps View

3.11.1 Settings Group

Acceleration Type

The Acceleration Type list box 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 Group

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 Pro 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.

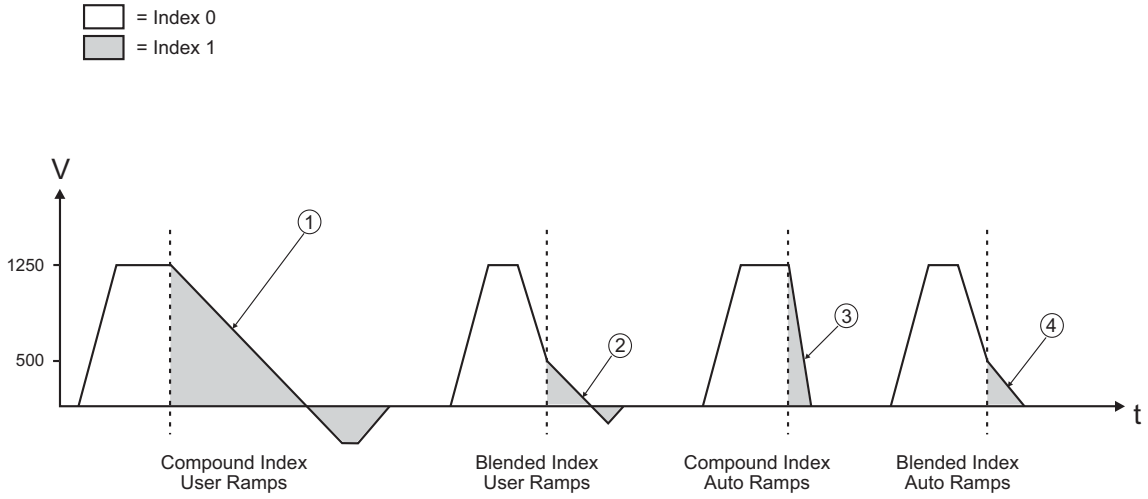
User Ramps Radio Button

Prior to the introduction of this feature, User Ramps was the only ramp control type available. 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 Figure 47 and Figure 48 below for examples of how User Ramps work. For more information on Index.#.CompoundInitiate and/or Index.#.BlendInitiats, see the programming section of this manual.

Auto Calculate Ramps Radio Button

When Auto Calculate Ramps is selected the drive 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 of deceleration rate is ignored, See the figures below for examples of how Auto Calculate Ramps work.

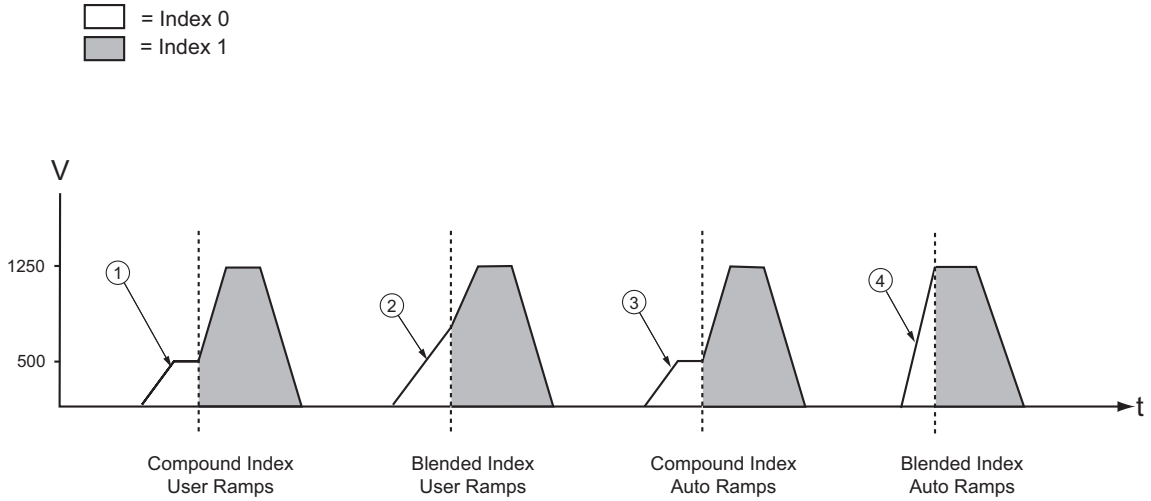
	Distance (Revs)	Velocity (RPM)	Accel (RPM/sec)	Decel (RPM/sec)
Index 0	20	1250	2000	3000
Index 1	3	500	500	500



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.

Figure 47: Ramps Examples of a Fast Index to a Slower Index

	Distance (Revs)	Velocity (RPM)	Accel (RPM/sec)	Decel (RPM/sec)
Index 0	5	500	1000	1000
Index 1	20	1250	2000	2000



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.

Figure 48: Ramps Examples of a Slow Index to a Faster Index

3.11.2 Ramps Group

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 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 Decel

The value entered here is the deceleration ramp that is used when a software or hardware travel limit is hit.

3.12 Torque View

The Torque View allows you to edit torque level and limit parameters as well as view real-time torque values when online.

By selecting Torque in the Hierarchy Tree, the Torque View will appear on the right (see Figure 49). The right part of the window is divided into groups. An explanation of the groups and their functions is provided below.

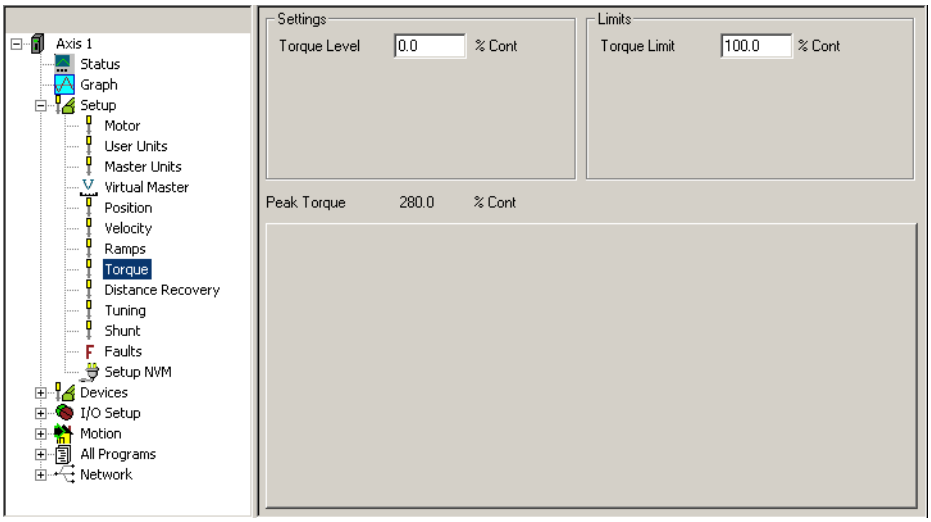


Figure 49: Torque View

3.12.1 Settings Group

Torque Level

This parameter sets the activation point for the Torque Level Active source. If set to 100%, the Torque Level Active source will activate any time the Torque Command reaches or exceeds 100% continuous. This parameter is specified in Torque User Units.

3.12.2 Limits Group

Torque Limit

This parameter sets the value to which the Torque Command will be limited when the Torque Limit Enable destination is active. To make the Torque Limit always active, assign the Torque Limit Enable destination to the Initially Active source on the Assignments view.

3.12.3 Peak Torque

Displays the Peak Torque for the motor drive combination setup in the Setup View.

3.12.4 Online Tab

If online, this view will show the Torque Command, Limited Torque, Foldback RMS, and Shunt Power RMS.

3.13 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.

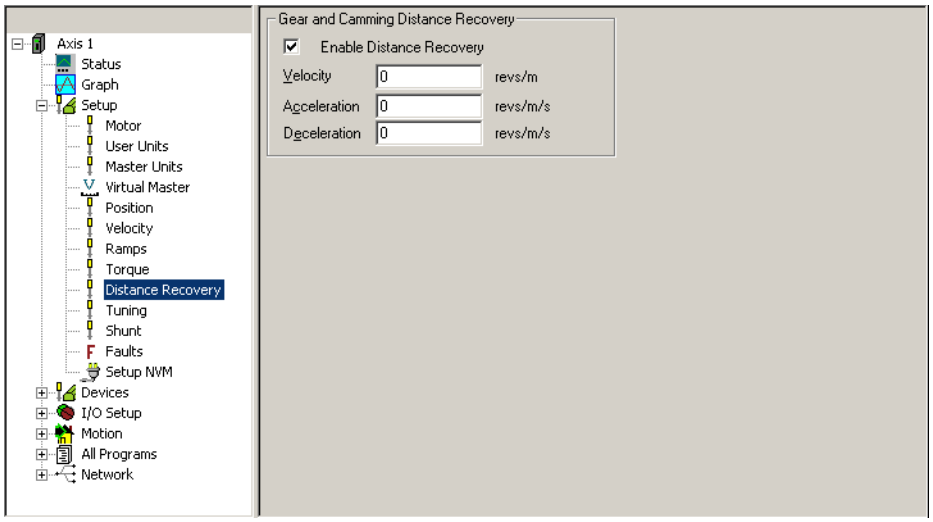


Figure 50: Distance Recovery View

3.13.1 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.

3.14 Tuning View

The Tuning View allows you to modify tuning parameters based on specific application information.

By selecting Tuning in the Hierarchy Tree, the Tuning View will appear on the right (see Figure 51). The right part of the window is divided into groups. An explanation of the groups and their functions is provided below.

For help on calculating tuning parameters and more in-depth tuning information, turn to *Tuning Procedures* on page 187.

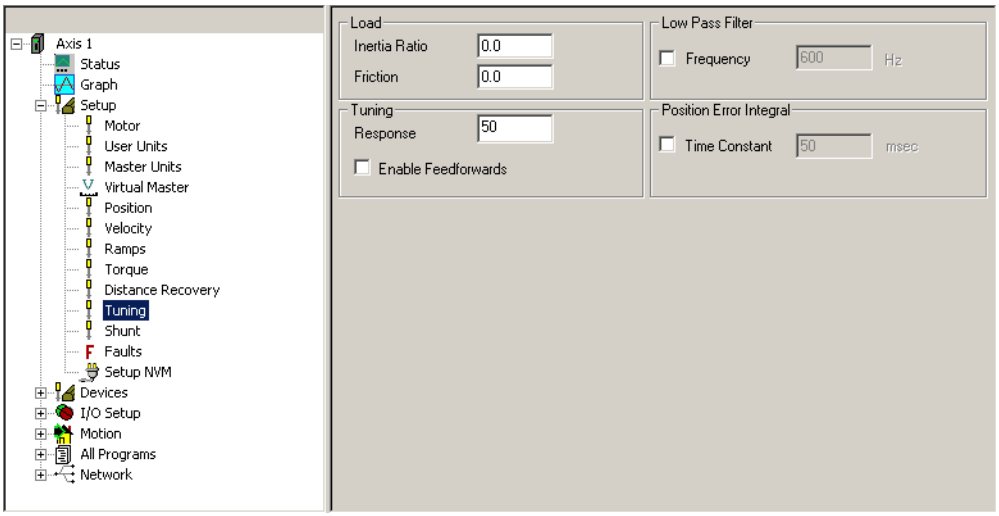


Figure 51: Tuning View

3.14.1 Load Group

Inertia Ratio

Inertia Ratio specifies the load to motor inertia ratio and has a range of 0.0 to 50.0. If the exact inertia is unknown, a

conservative approximate value should be used. If you enter an inertia value higher than the actual inertia, the resultant motor response will tend to be more oscillatory.

Friction

This parameter is characterized in terms of the rate of friction increase per 100 motor RPM. If estimated, always use a conservative (less than or equal to actual) estimate. If the friction is completely unknown, a value of zero should be used. A typical value used here is less than one percent.

3.14.2 Low Pass Filter Group

The Low Pass Filter will reduce machine resonance due to mechanical coupling and other flexible drive/load components by filtering the command generated by the velocity loop.

Low Pass Filter Enable Check Box

When this check box is selected it enables a Low Pass Filter to be applied to the output of the velocity command before the torque compensator.

Low Pass Filter Frequency

This parameter defines the Low Pass Filter cut-off frequency. Signals exceeding this frequency will be filtered at a rate of 40 dB per decade. The default value is 600 Hz.

3.14.3 Tuning Group

Response

The Response adjusts the velocity loop bandwidth with a range of 1 to 500 Hz. In general, it affects how quickly the drive will respond to commands, load disturbances and velocity corrections. A good value to start with (the default) is 50 Hz. The maximum value recommended is 80 Hz.

Enable Feedforwards Check Box

When feedforwards are enabled, the accuracy of the Inertia and Friction parameters is very important. If the Inertia parameter is larger than the actual inertia, the result could be a significant overshoot during ramping. If the Inertia parameter is smaller than the actual inertia, following error during ramping will be reduced but not eliminated. If the Friction parameter is greater than the actual friction, it may result in velocity error or instability. If the Friction parameter is less than the actual friction, velocity error will be reduced but not eliminated.

3.14.4 Position Error Integral Group

Time Constant Check Box

When this check box is selected it enables the Time Constant parameter.

Time Constant

Position Error Integral is a control term, which can be used to compensate for the continuous torque required to hold a vertical load against gravity. It is also useful in low speed applications, which have high friction.

The user configures this control term using the “Position Error Integral Time Constant” parameter. This parameter determines how quickly the drive will correct for in-position following error. The time constant is in milliseconds and defines how long it will take to decrease the following error to 37 percent of the original value. In certain circumstances the value actually used by the drive will be greater than the value specified here.

Min Time Constant = 1000/Response

For example, with “Response” set to 50, the minimum time constant value is 1000/50 = 20 msec.

3.15 Shunt View

The Shunt view is used to configure the external shunt for the EP204, EP206, EP209 and EP216 drives.

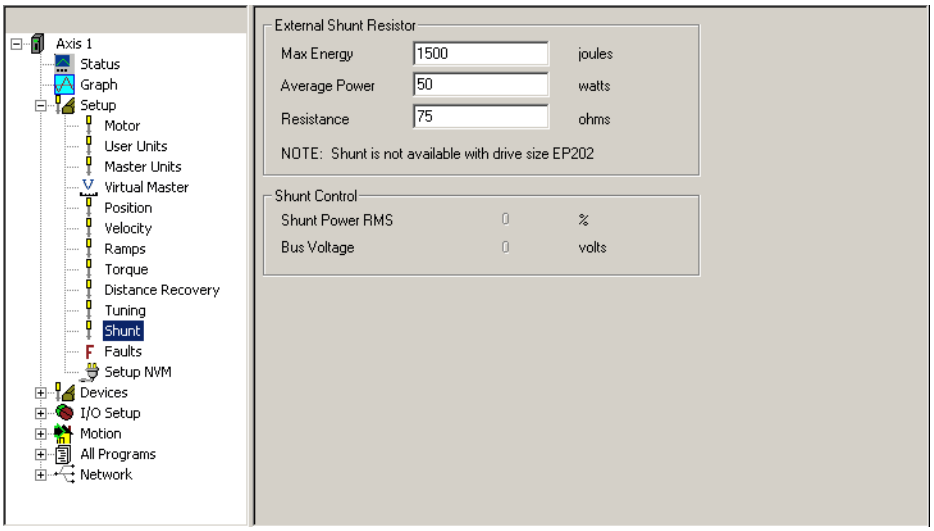


Figure 52: Shunt View

3.15.1 External Shunt Resistor Group

Max Energy

The total energy that can be absorbed by the resistor when its initial temperature is ambient temperature.

Average Power

The average power rating of the shunt resistor.

Resistance

The resistance value of the shunt resistor.

3.15.2 Shunt Control Group

The parameters in this group are read-only and only available when online with the drive.

Shunt Power RMS

This parameter models the thermal heating and cooling of the drive internal shunt. This parameter indicates the percent of shunt capacity utilization and is based on the Heat Sink RMS value. When this value reaches 100 percent the drive will generate a RMS Shunt Power Fault.

Bus Voltage

Displays the actual measured voltage of the DC power bus.

3.16 Faults View

The Faults View displays any active faults when online. Figure 53 below shows the Faults view offline.

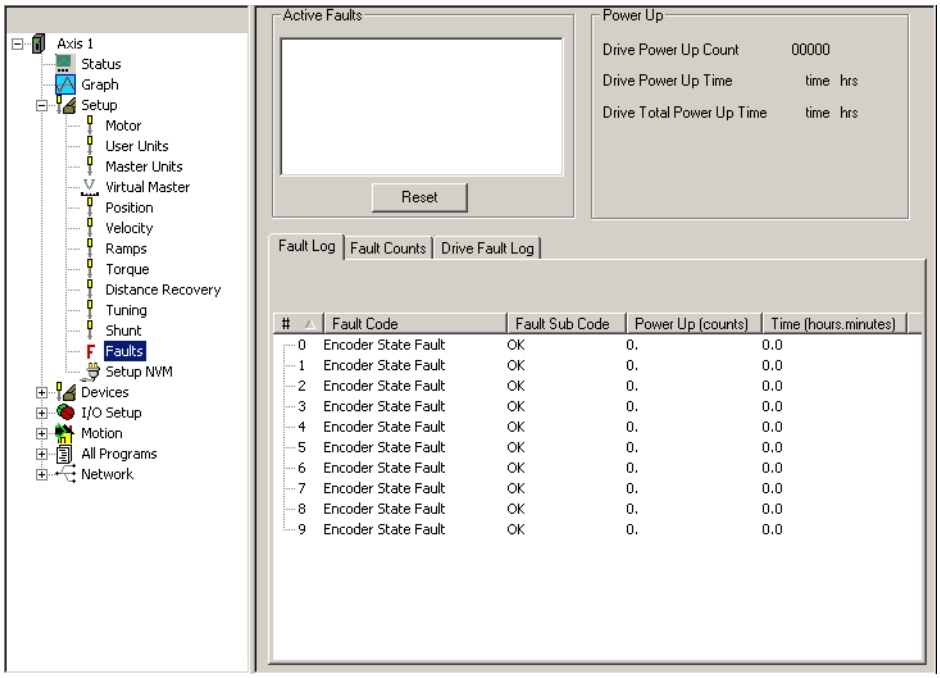


Figure 53: Faults View - Offline

When online and a fault is detected, the Faults window opens, showing the fault condition and allows the fault to be reset or ignored. Pressing Reset attempts to reset the fault if the cause of the fault has been removed. Pressing Ignore just closes the faults window.

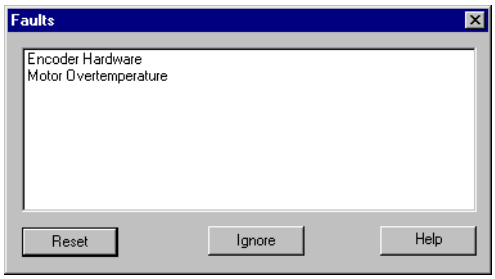


Figure 54: Faults Window

When online, the Active Faults window, Reset button and Power Up group window become active. There will also be three tabs that appear, Fault Log, Fault Counts, and Drive Fault Log.

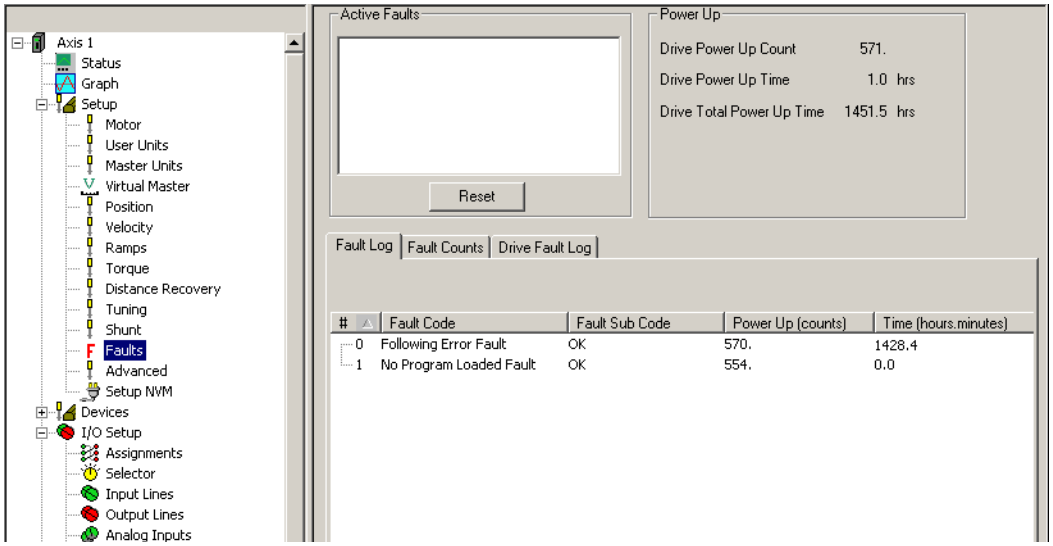


Figure 55: Faults View Online - Faults Log Tab

3.16.1 Active Faults Group

The Active Faults group contains the Active Faults window.

Active Faults Window

This window displays any active faults in the system. Those faults which do not require a reboot can be cleared by clicking on the Reset button. For more detailed fault information, refer to *Diagnostics and Troubleshooting* on page 193.

3.16.2 Power Up Group

These parameters will be active when online with the drive.

Drive Power Up Count

This parameter shows the number of times the drive has been powered up since the last reset by the factory.

Drive Power Up Time

This parameter shows the time, in hours, since the drive was last powered up.

Drive Total Power Up

This parameter shows the total time that the drive has been powered up since reset by the factory.

3.16.3 Fault Log Tab

The Fault Log tab is visible when online and consists of a list of the ten most recent faults detected by the drive. These are saved in non-volatile memory to be preserved during power down.

Faults are listed in reverse order of occurrence-the most recent fault is listed first, and older faults are pushed off the list.

Fault Code

The fault code has the same description of the fault that is or was reported in the Active Faults window.

Fault Sub Code

The fault subcode applies to only a few faults and provides some additional information about the fault when available. When there is no additional information for the fault, OK is displayed.

Power Up (counts)

This is the drive's power up counter at the time of the fault.

Time (hours.minutes)

This is the drive's total power up time in tenths of an hour at the time of the fault.

3.16.4 Fault Counts Tab

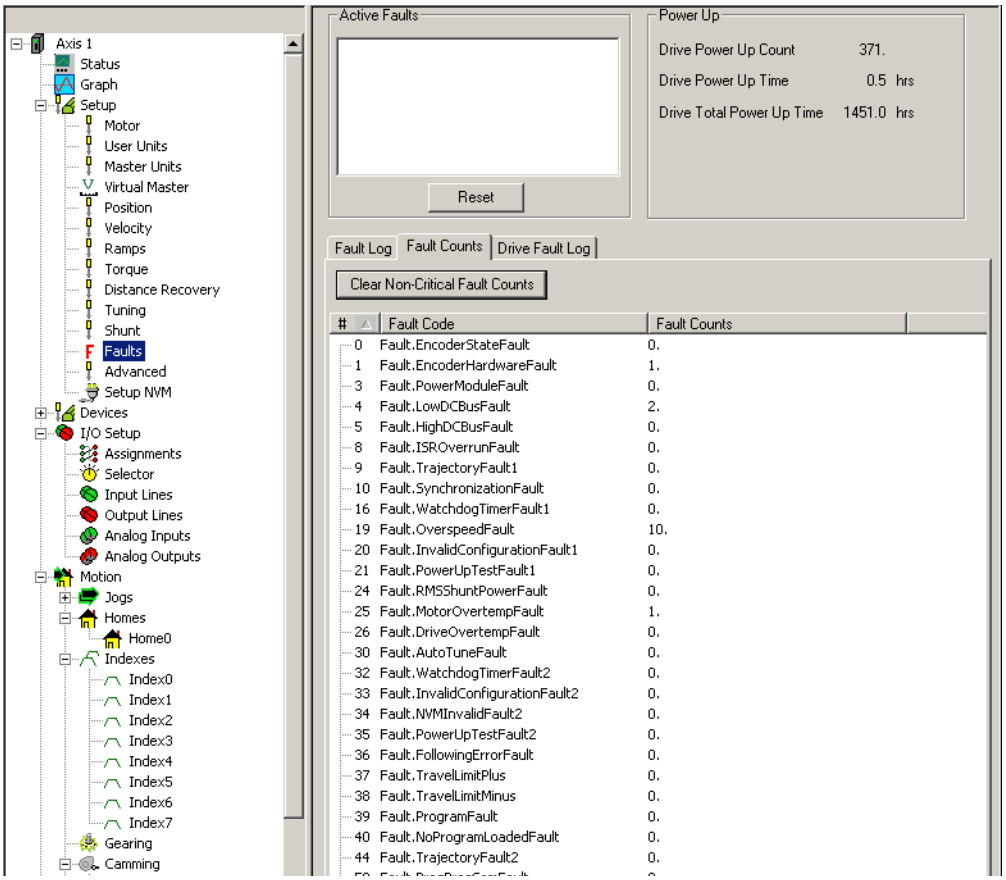


Figure 56: Fault View - Fault Counts Tab

The Fault Counts tab is visible when online and consists of a list of all supported faults and the number of times each fault has been detected. Most of the counts start at zero following a power up or a configuration download.

A few faults are saved in non-volatile memory so that the total number of times they have occurred can be easily viewed. These faults tend to have hardware significance.

Fault Code

This is the faults parameter name.

Fault Counts

This is the number of times that the fault has occurred.

Clear Non-Critical Fault Counts Button

Pressing this control button will zero the Fault Counts of all non-critical faults(i.e.software travel limit). Critical faults are faults that are used by the factory to verify warranty claim and aid in troubleshooting the drive.

3.16.5 Drive Fault Log Tab

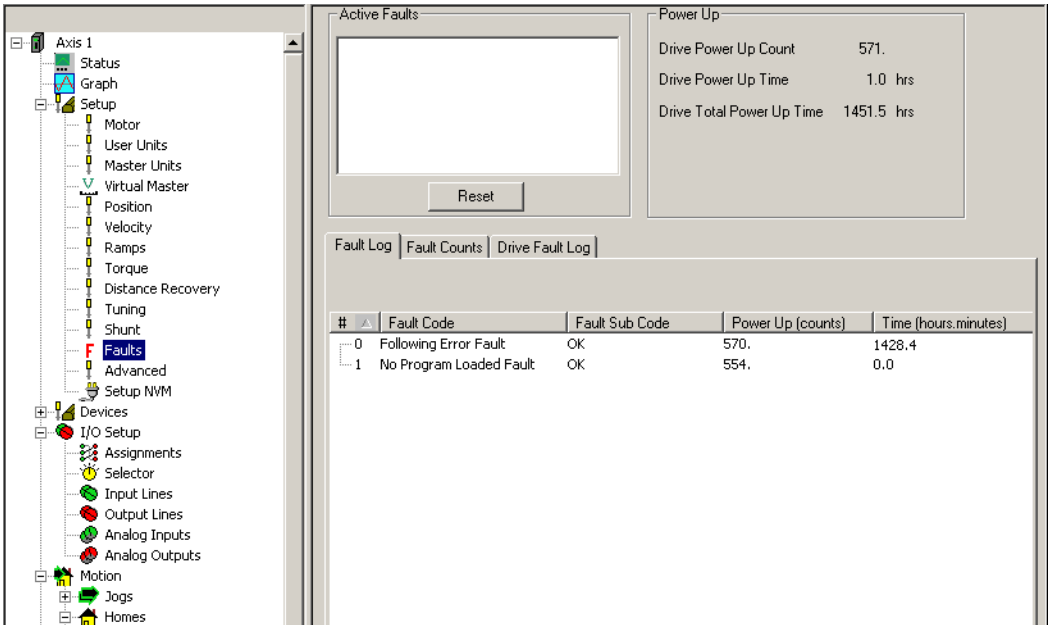


Figure 57: Faults View - Drive Fault Log Tab

The Drive Fault Log tab is visible when online and consists of a list of the ten most recent faults detected by the drive. These are saved in the drive's non-volatile memory to be preserved during power cycles. Faults are listed in reverse order of occurrence so that the most recent is listed first.

Fault Code

The fault code has the same description of the fault that is or was reported in the Active Faults window.

Power Up (Counts)

This is the Drive Power Up Counts when the fault was detected.

Time (Hours.Minutes)

This is the Drive Power Up Time when the fault was detected.

3.17 Setup NVM View

At power-down, parameters can be saved to Non-Volatile Memory (NVM). See the "How Communications Work" section of the "Operational Overview" chapter for more details. In PowerTools Pro, you can customize which parameters will be saved in non-volatile memory.

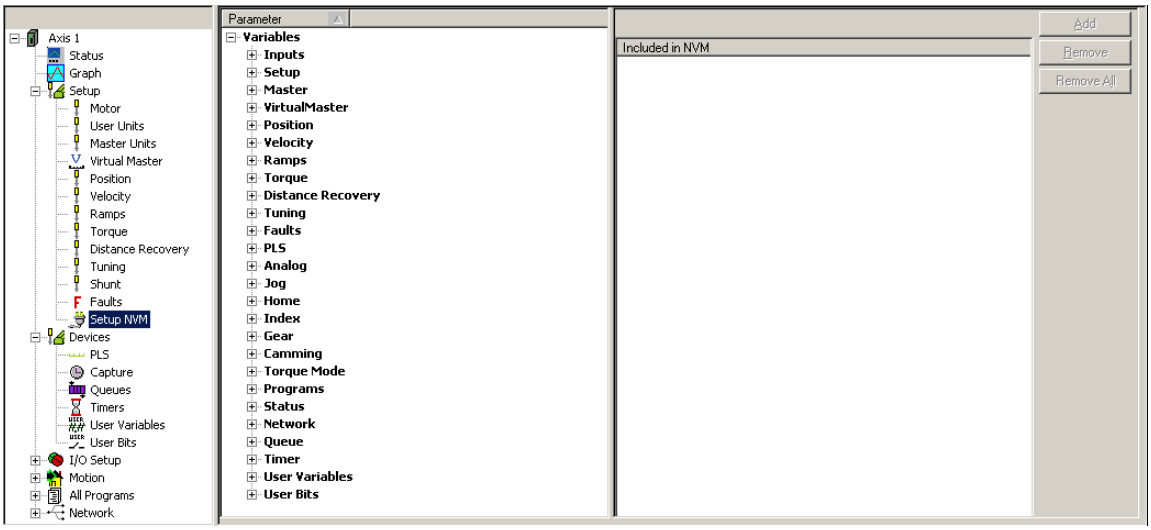


Figure 58: Setup NVM View

3.18 Devices / Vars Group

3.19 PLS View

The PLS View allows users to define Programmable Limit Switches (PLS) for advanced machine operation. By selecting PLS in the Hierarchy Tree, the PLS View will appear on the right (see Figure 59).

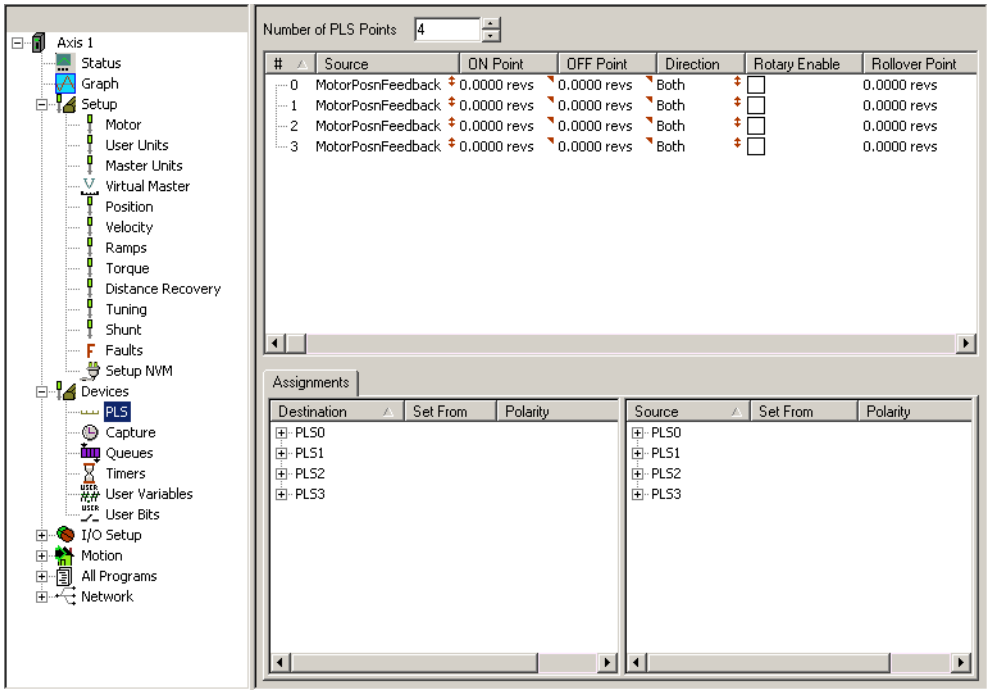


Figure 59: PLS View

A PLS can be used to turn on or off a bit based on feedback position, commanded position, or master feedback position. Eight global PLS's are available for a single application. To operate a PLS, first it must be enabled (see the PLS enable destinations in the assignments view) and then the Absolute Position Valid source must be active. Each PLS has its own OnPoint and Off Point, as well as a Rollover Point.

The terms OnPoint and Off Point assume movement in the positive direction. Those labels should be reversed if traveling in the negative direction.

Number of PLS Points

This parameter determines the number of PLS Points that will be used. Count always begins with 0, so 5 points will be 0 to 4. Up to eight PLS points may be used simultaneously.

Source

The source list box is used to select the source for the individual PLS and reference its OnPosn and OffPosn to determine the PLS.#.Status parameter. The source of a PLS (PLS.#.Source) can be assigned to the motor axis (MotorPosnFeedback, MotorPosnCommand), a master synchronization signal (MasterPosnFeedback) or the real time clock from the drive's built-in microprocessor (FreeRunTime). The term motor axis refers to the motor being controlled by the drive.

ON Point

PLS.#.Status will be active when the selected source position is between the PLS.#.OnPosn and the PLS.#.OffPosn.

Examples:

Example 1: Assume that the PLS.#.Source is MotorPosnFeedback and 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.

Example 2: Assume that the PLS.#.Source is MotorPosnFeedback and PLS.#.Direction is set to "Plus". When traveling in the positive direction and the feedback position executes the OnPosn, the PLS.#.Status will activate. As the motor continues in the positive direction, the PLS.#.Status will deactivate when feedback position reaches or exceeds the OffPosn. If motor travel changes to the negative direction after passing the OffPosn, the PLS.#.Status will remain deactivate. If the motor travel changes to the negative direction before passing the OffPosn the PLS.#.Status will deactivate. All on/off positions are defined by the selected PLS.#.Source, either user units or seconds.

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 in-active 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 (PLS.#.Direction) 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.

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.

If accessing this parameter from a network, the following table displays the value for this 16-bit integer.

0	N/A
1	Both
2	Plus
3	Minus

Rotary Enable

This parameter (PLS.#.RotaryRolloverEnable) is used to enable the RotaryRolloverPosn for this PLS.

Rollover Point

This parameter (PLS.#.RotaryRolloverPosn) 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.

Example: if the source of the PLS is FreeRunTime, this parameter is the time of the first repeat for this PLS. When enabled it causes the PLS to repeat every time this time is passed. The repeating range begins at the beginning of the trajectory update rate and ends at the RotaryRolloverPosn.

3.20 Capture View

Many applications require the ability to accurately capture a position at an exact moment in time so that the motion profile can be repeatably time accurate. The Epsilon EP-P drive allows for this by using the Capture component. The Capture component is fully controlled by the user through the Assignments View or through the Programs View. When the capture is activated, the following parameters are captured and stored: Time, Command Position, Feedback Position, and Master Feedback Position.

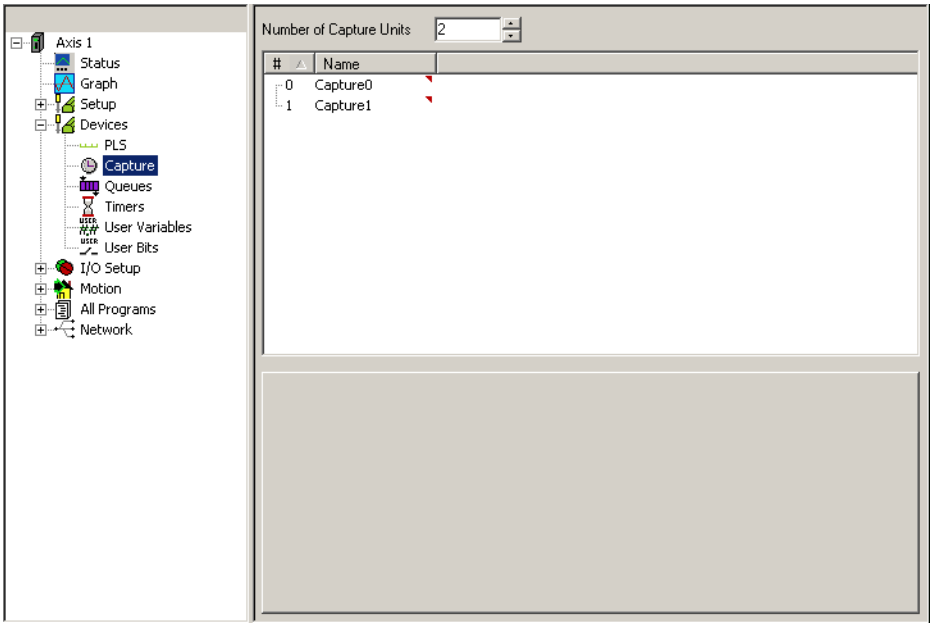


Figure 60: Capture View

The user must determine which signals are used to enable, activate, and then reset the Capture component. A CaptureTriggered destination is then available to indicate to the user that data has been captured and is available for use.

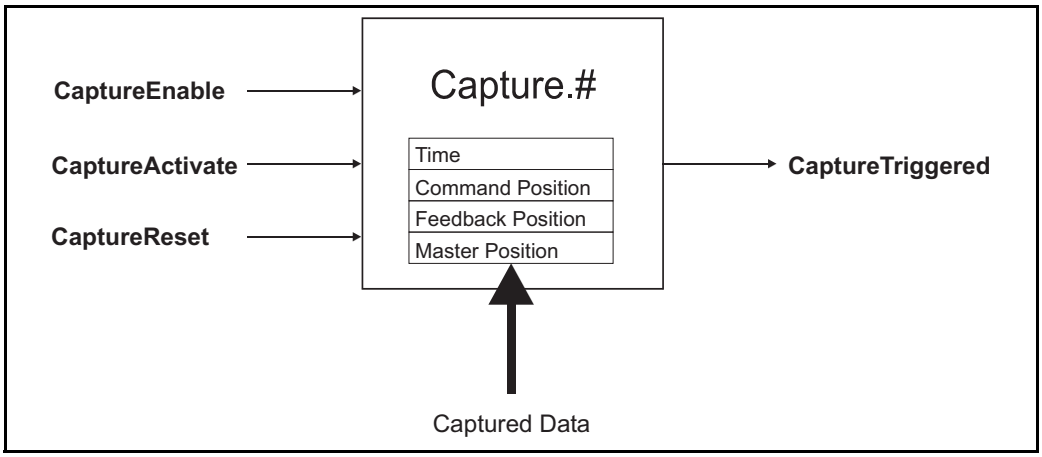


Figure 61: Capture Component

A detailed description of all the Capture parameters is below.

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 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.

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.

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

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.

Capture Number

This parameter defines the number of Capture objects available. Maximum is eight.

Captured Data

The data that is acquired by the position capture is 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 CaptureTriggered activated.

Capture.#.CapturedPositionCommand

The command position, in user units, at the time when CaptureTriggered activated.

Capture.#.CapturedPositionFeedback

The feedback position, in user units, at the time when CaptureTriggered activated.

Capture.#.CapturedMasterPostion

The master axis feedback position, in master axis distance units, at the time when CaptureTriggered activated.

The captured data remains in these parameters until the capture component is reset and CaptureActivate is activated. 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.

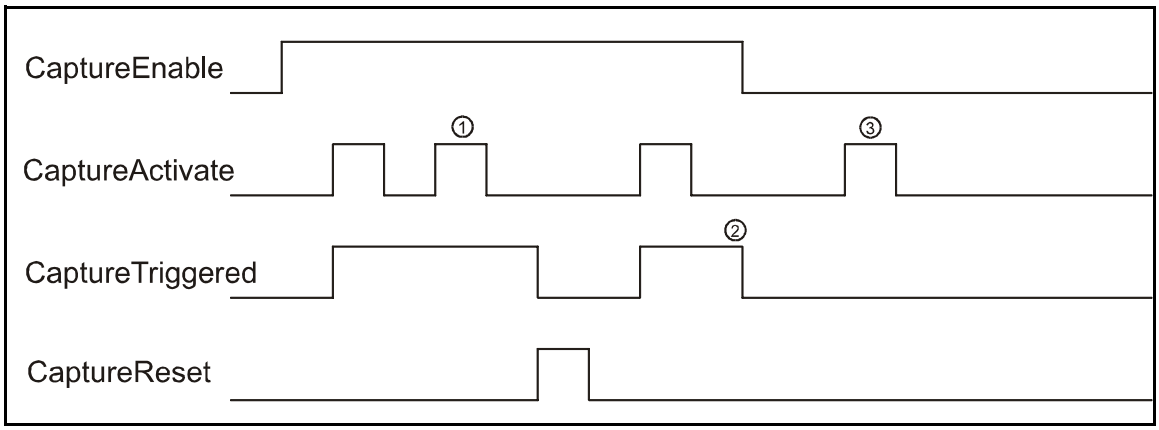


Figure 62: Capture Timing Diagram

Figure 62 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. The CaptureActivate has no effect when the CaptureTriggered is active. CaptureActivate is ignored until the capture component has been reset.
2. When CaptureEnable is deactivated, CaptureTriggered is deactivated, and the capture component 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.

Assignments that Automatically Use Position Capture

Certain assignments (Sources or Destinations) automatically generate position capture data internally without using the capture component. This data is used by the drive, but is not directly available to the user like the capture component data. Following is a list of assignments that automatically generate or use captured data.

Sources that generate capture data:

Drive Inputs 1-8 – The Epsilon EP-P drive Inputs 1-8 are constantly monitored by the processor, and when activated will automatically capture related data. The processor controls all resetting requirements. The capture only occurs on the rising edge of an input. When the input is activated, the captured data will automatically be passed to the destination that it is assigned to. The destination may then use the captured data to accurately initiate motion (if it is a motion-related destination).

Motor Encoder Marker – The rising edge of the motor encoder marker pulse will automatically capture data. This will allow the user to accurately initiate motion on the rising edge of the motor encoder marker pulse. The falling edge of the marker pulse does not capture data.

Master Encoder Marker – The rising edge of the master encoder marker pulse will automatically capture data. This allows the user to accurately initiate motion on the rising edge of the master encoder marker pulse. The falling edge of the marker pulse does not capture data.

Index/Jog Command Complete – Activation of the command complete signal at the end of indexes and jogs will automatically capture data. A subsequent index, jog, or dwell can then use the captured data to start itself extremely accurate at the end of the previous motion.

Index/Jog At Velocity – Activation of the command complete signal at the end of indexes and jogs will automatically capture

data. A subsequent index, jog, or dwell can then use the captured data to start itself extremely accurately at the end of the previous motion.

PLS Status – A rising or a falling edge of a Global PLS will automatically capture data for use in initiating motion. In order to accurately initiate motion from a Global PLS, an assignment can be made from PLS.#.Status to the initiate destination.

Destinations that use captured data:

Index/Jog Initiates – When one of the sources listed above is assigned to an Index or Jog 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 get the accurate stopping position for the registration index.

3.21 Queues View

Many applications require the ability to store data in a temporary memory location as the data comes in. The user then has access to the data for use in a program or other operation. The Epsilon EP-P drive use an object called a queue to store data in this way. The queue is a first-in-first-out (FIFO) type memory device. In other words, the first piece of data put into the queue is the first piece of data to come out of the queue. The user has complete control over what data is stored in the queue, and when to put data into the queue, as well as when to remove it from the queue.

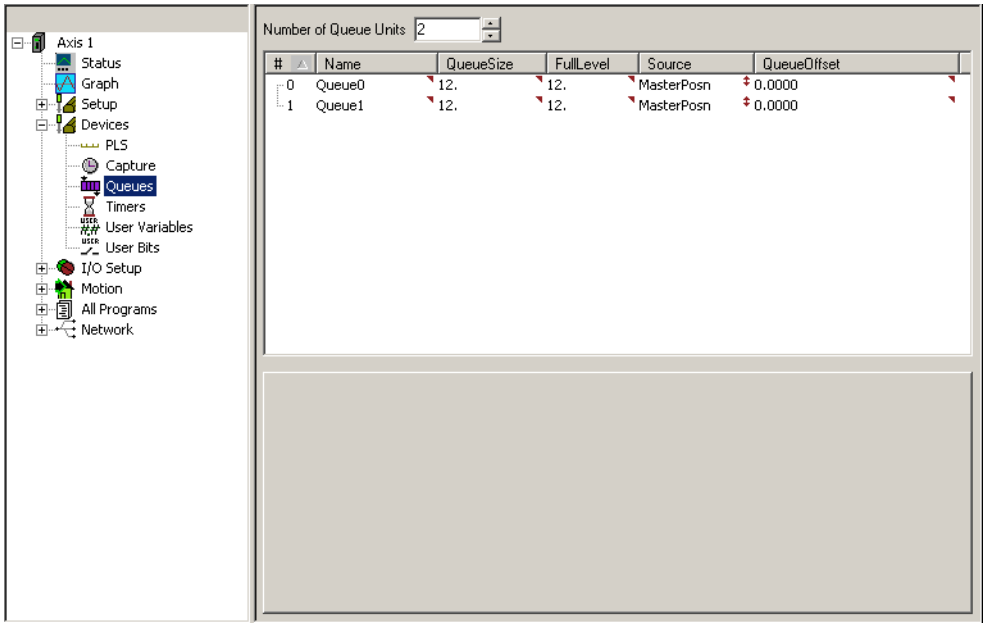


Figure 63: Queues View

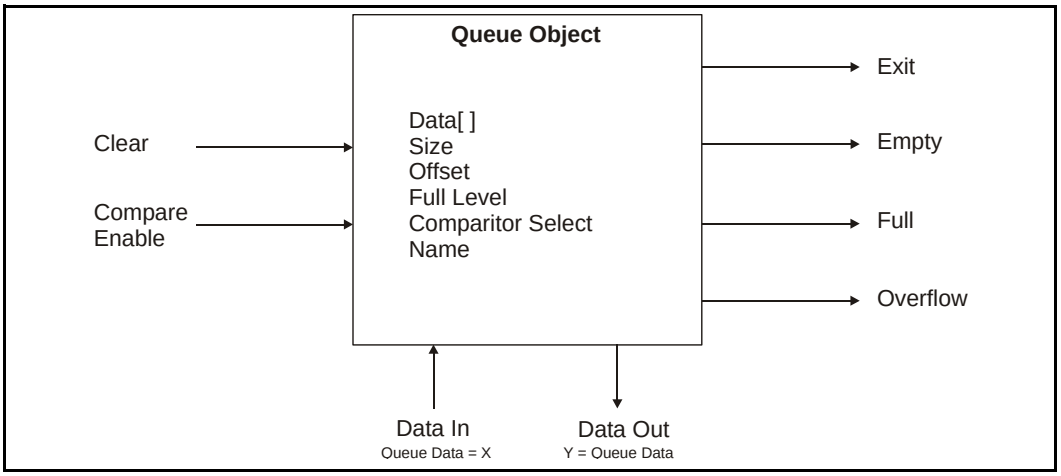


Figure 64: Queue Object and Components Diagram

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.

Queue Size

This is the maximum number of elements that can be stored in the queue. If more pieces of data than this number are in the queue at a time, then a Queue Overflow source will activate.

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.

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. This is only a flag and does not indicate a fault of any kind.

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 FeedBackPosn (Position Feedback), the sum of the data and offset are compared to the Position Feedback parameter. If set to MasterPosn (Master Position), then the sum is compared to the Master Feedback Position parameter, and if set to CommandPosn (Command Position), then the sum is compared to the Motor Commanded Position.

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.

Number of Queue Units

This selects the number of Queues available. Maximum of eight.

3.21.1 Queue Sources and Destinations

Sources

Queue Exit

This source activates when the Comparitor 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 comparitor 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 Comparitor Select parameter.

To fully understand the operation of the queue, the following diagram has a more detailed view of the Queue object.

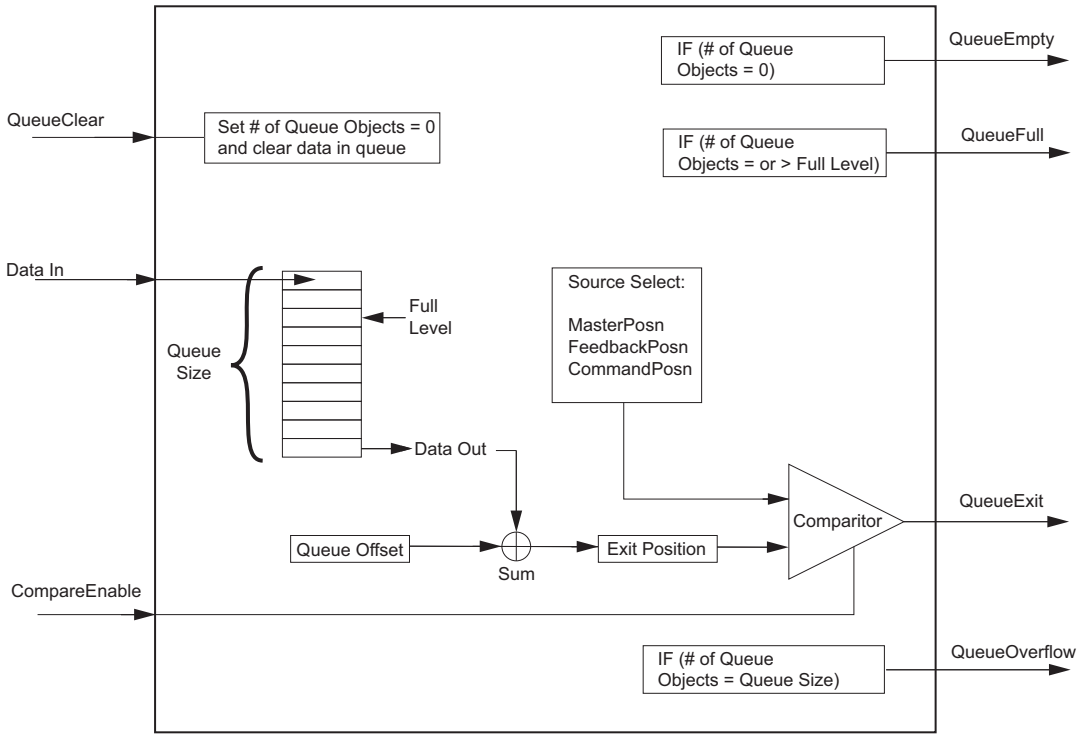


Figure 65: Detailed Queue Diagram

3.22 Timers View

The Timer 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.

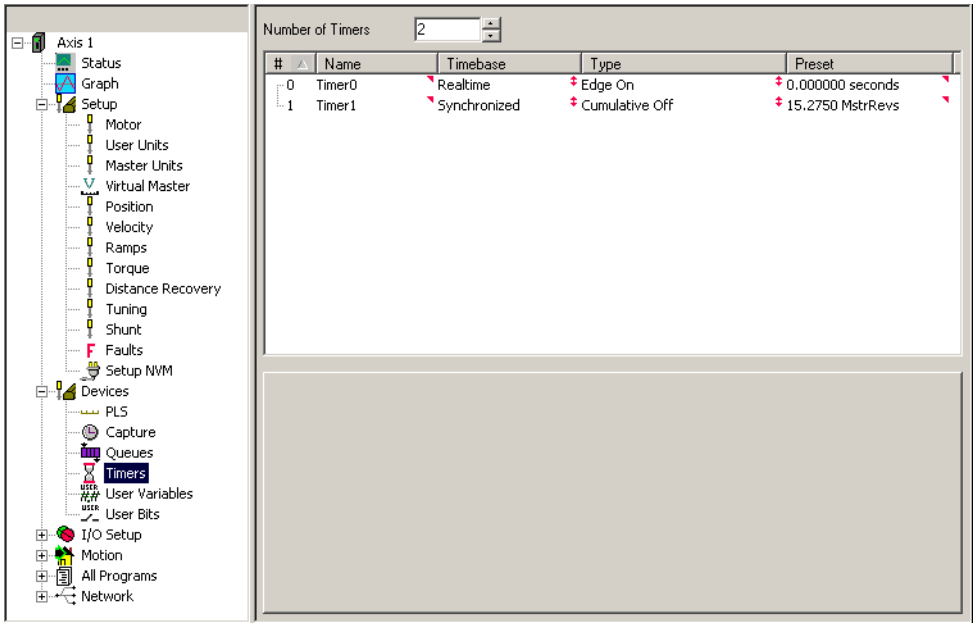


Figure 66: Timers View

Number of Timers

This parameter determines the number of Timer objects that are available in the configuration. Use the spin control 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 combobox 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.

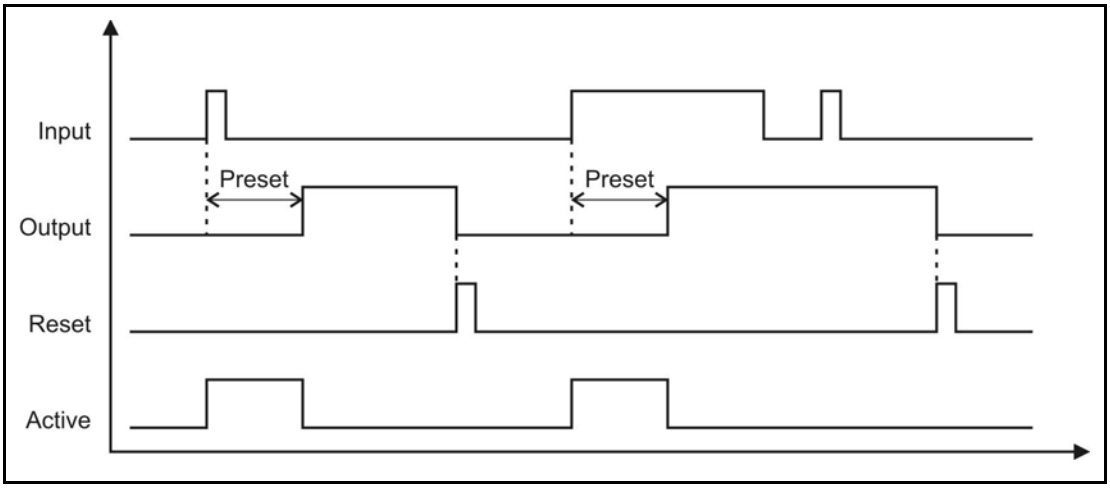


Figure 67: Edge On Timer - Timing Diagram

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.

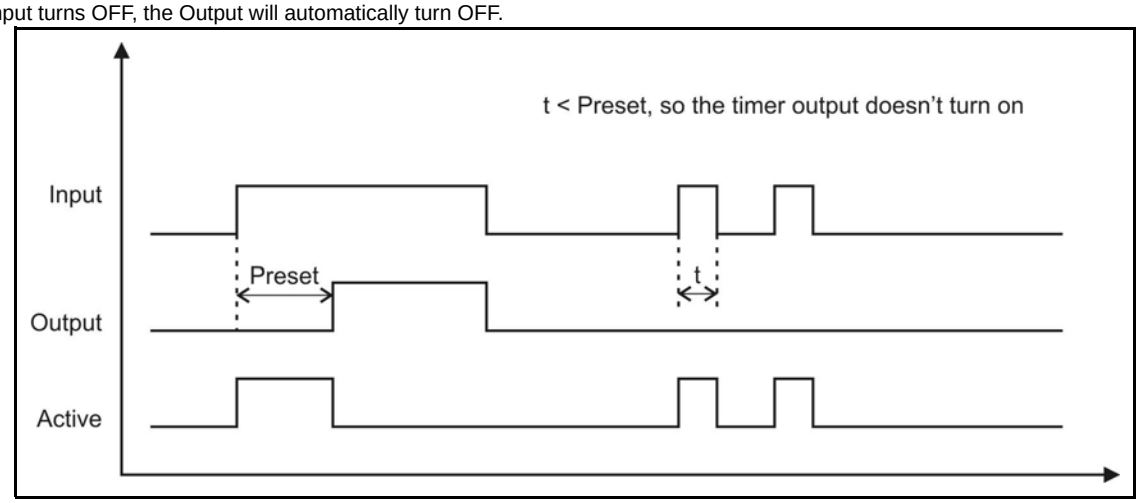


Figure 68: Level On Timer - Timing Diagram

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.

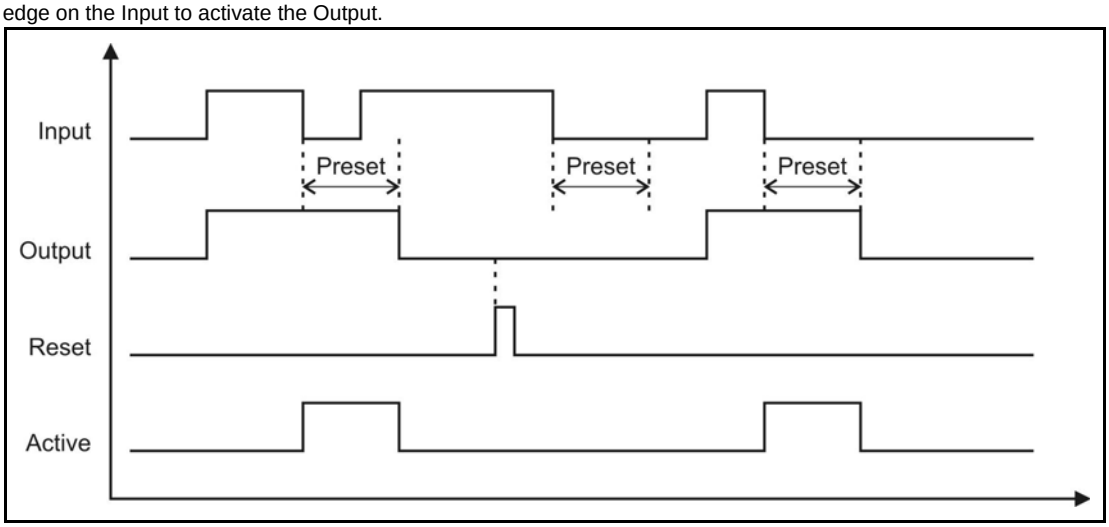


Figure 69: Edge Off Timer - Timing Diagram

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.

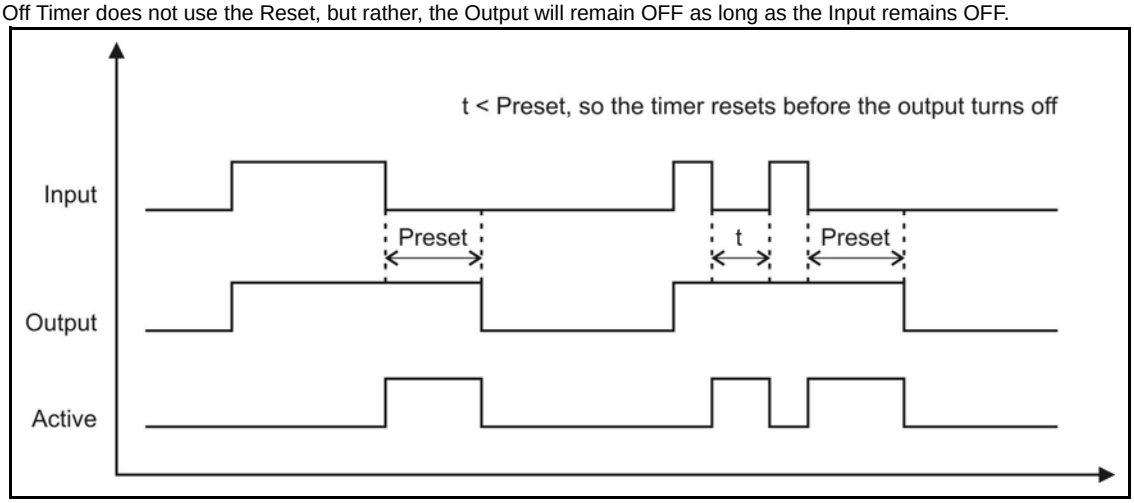


Figure 70: Level Off Timer - Timing Diagram

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 6 below shows an example timing diagram of the Cumulative On Timer.

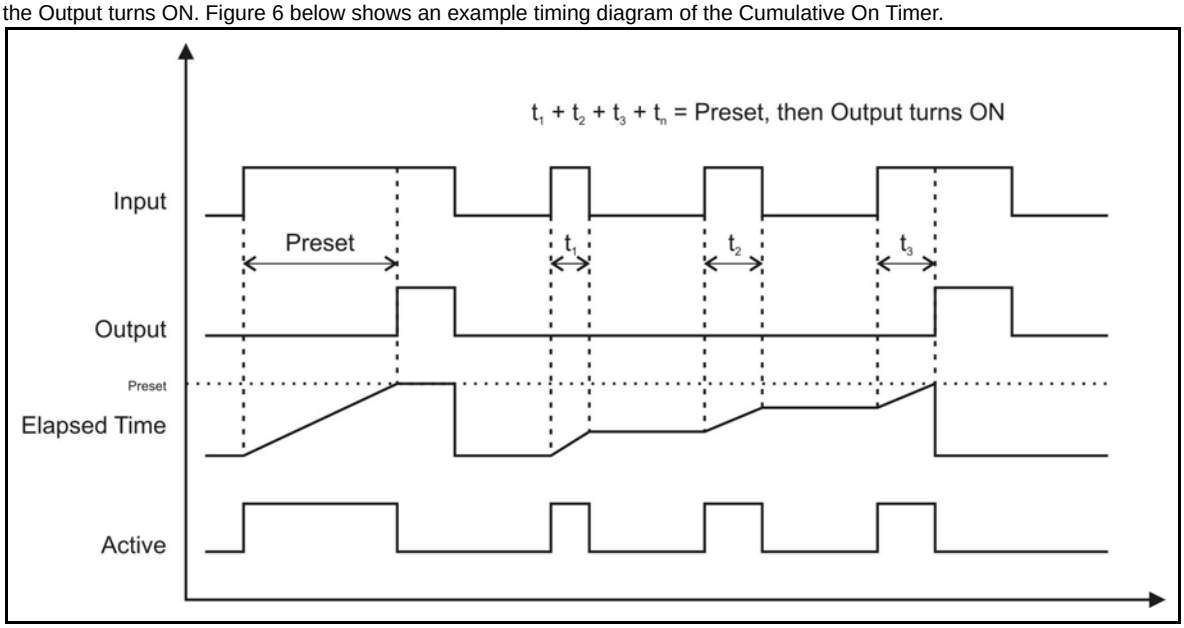


Figure 71: Cumulative ON Timer, No Reset - Timing Diagram

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.

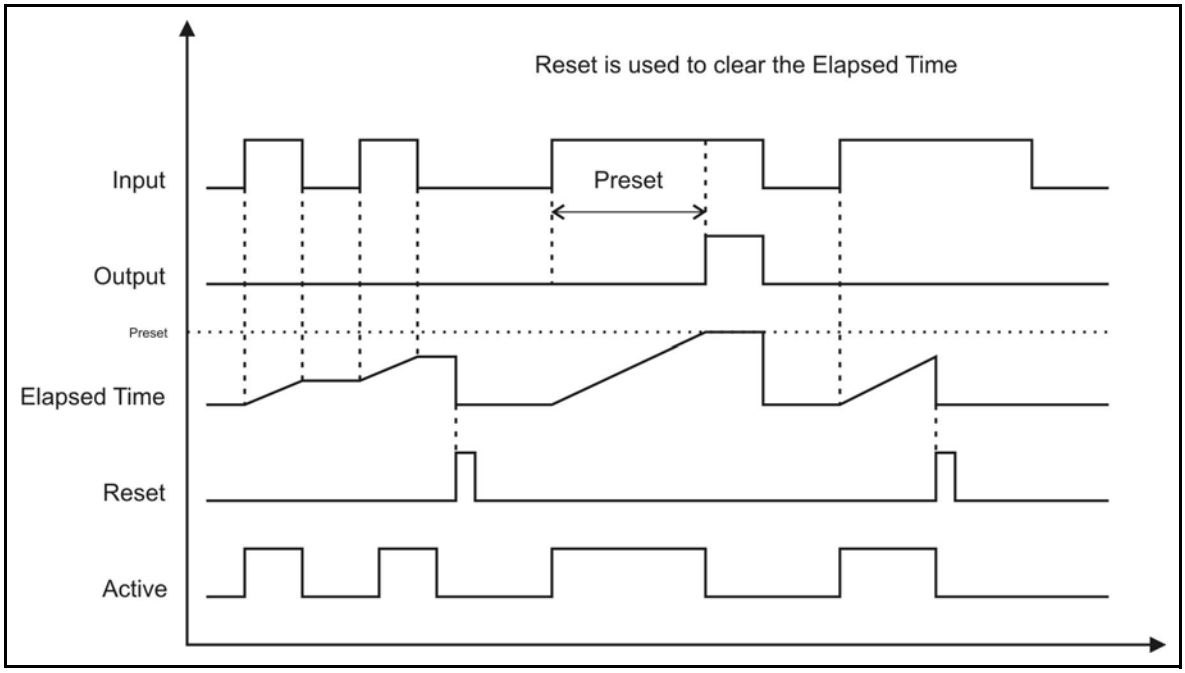


Figure 72: Cumulative ON Timer with Reset - Timing Diagram

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.

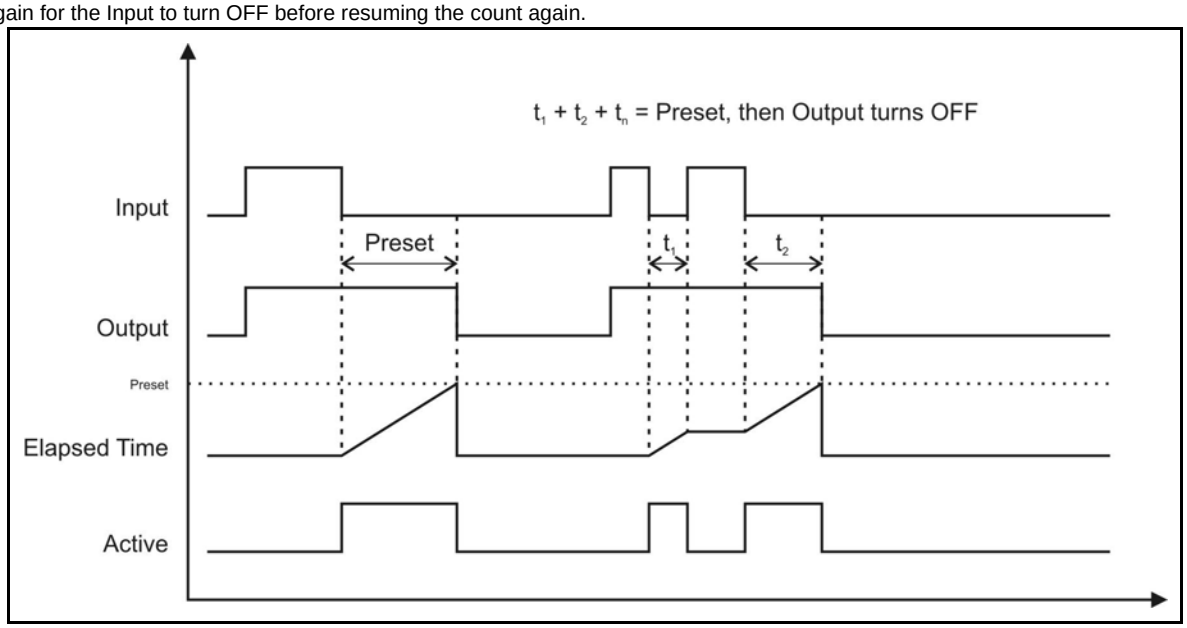


Figure 73: Cumulative OFF Timer - Timing Diagram

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.

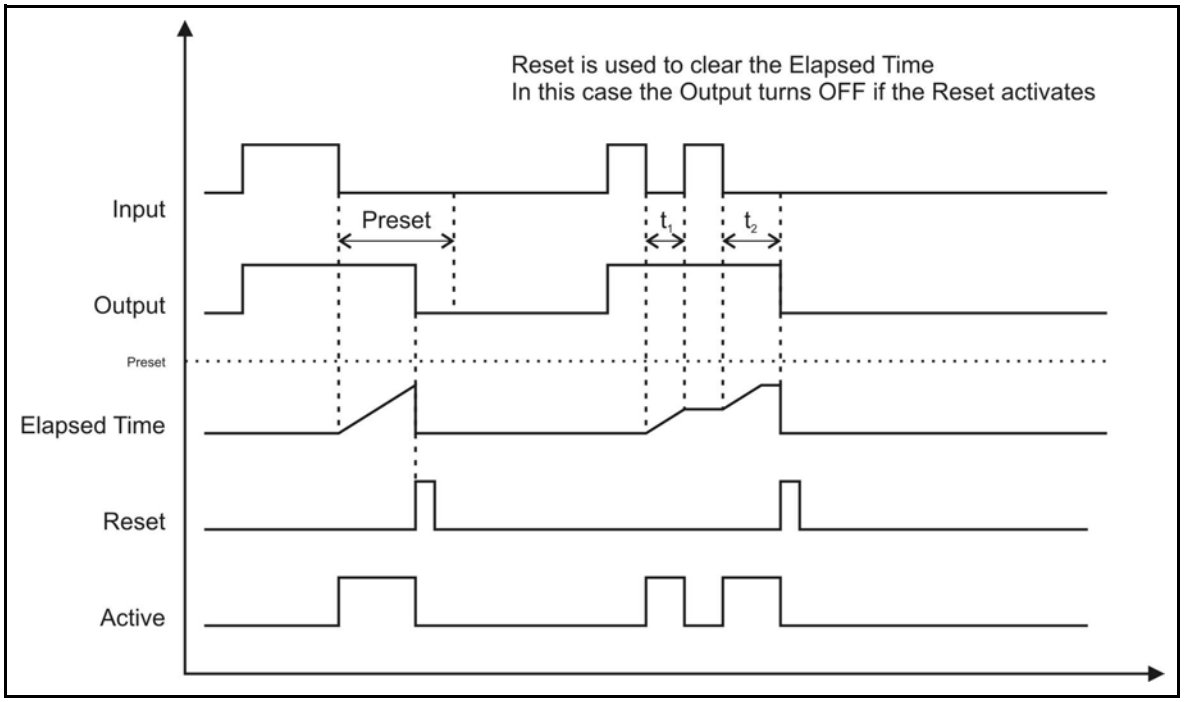


Figure 74: Cumulative OFF Timer with Reset - Timing Diagram

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.

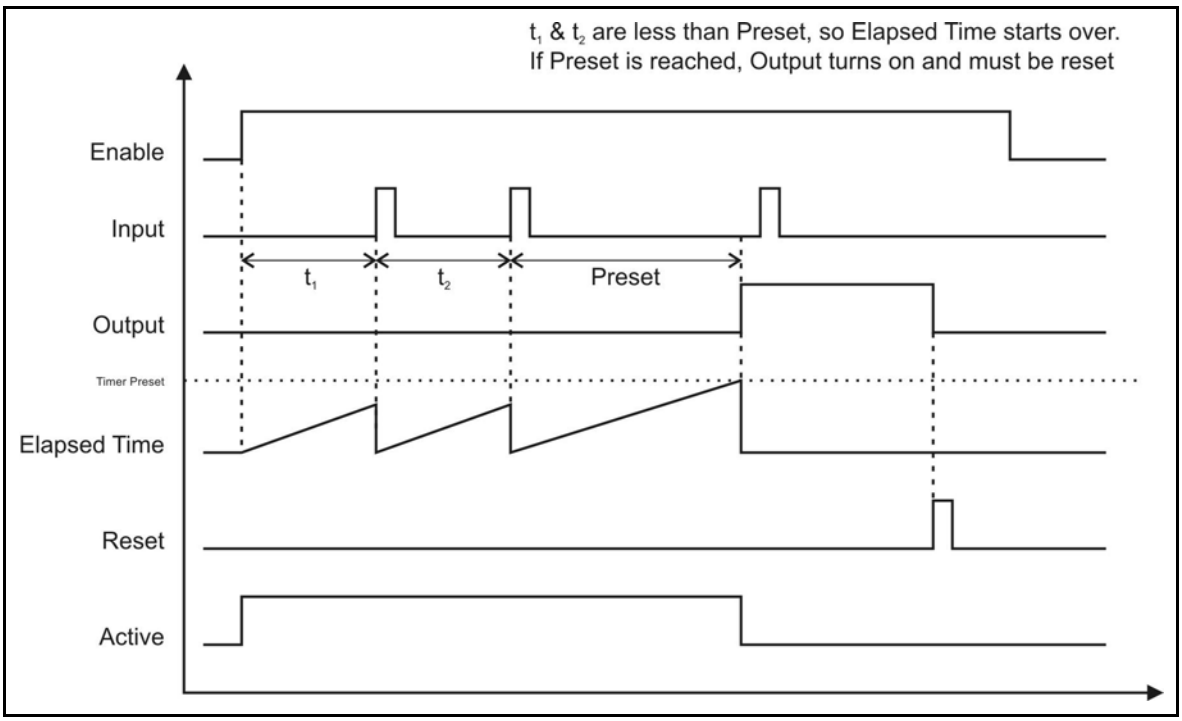


Figure 75: Watchdog Timer - Timing Diagram

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 past.

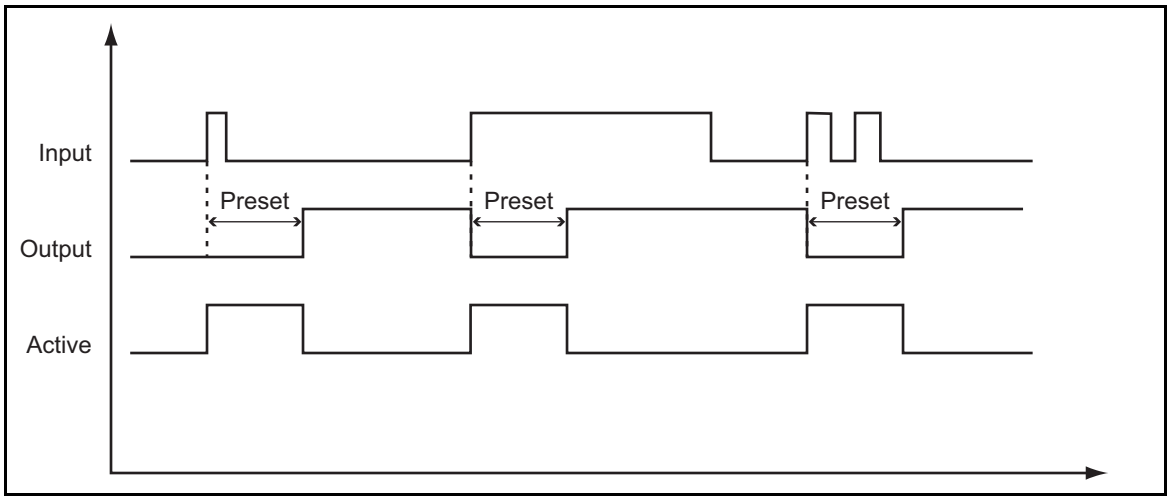


Figure 76: One Shot Timer - Timing Diagram

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.

3.22.1 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 77 below.

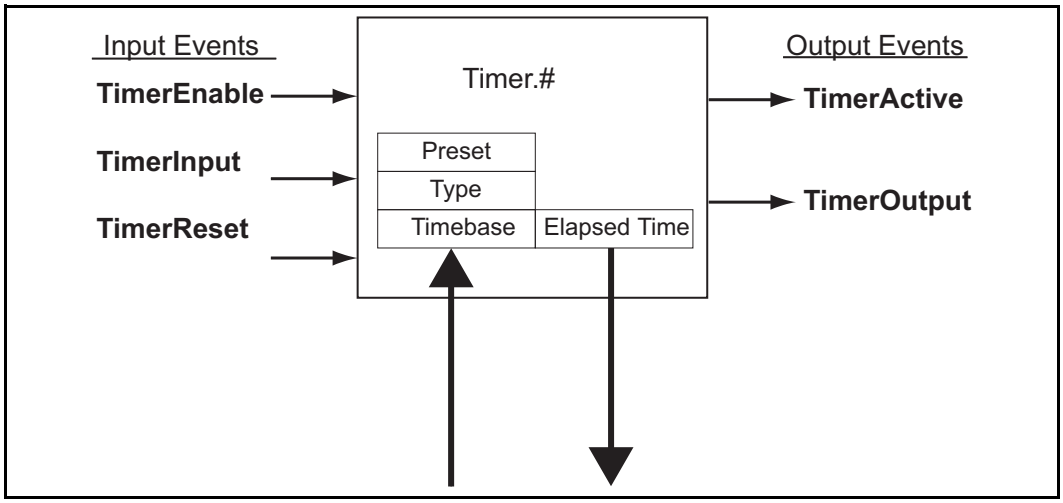


Figure 77: Timer Object Block Diagram

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 78 and Figure 79.

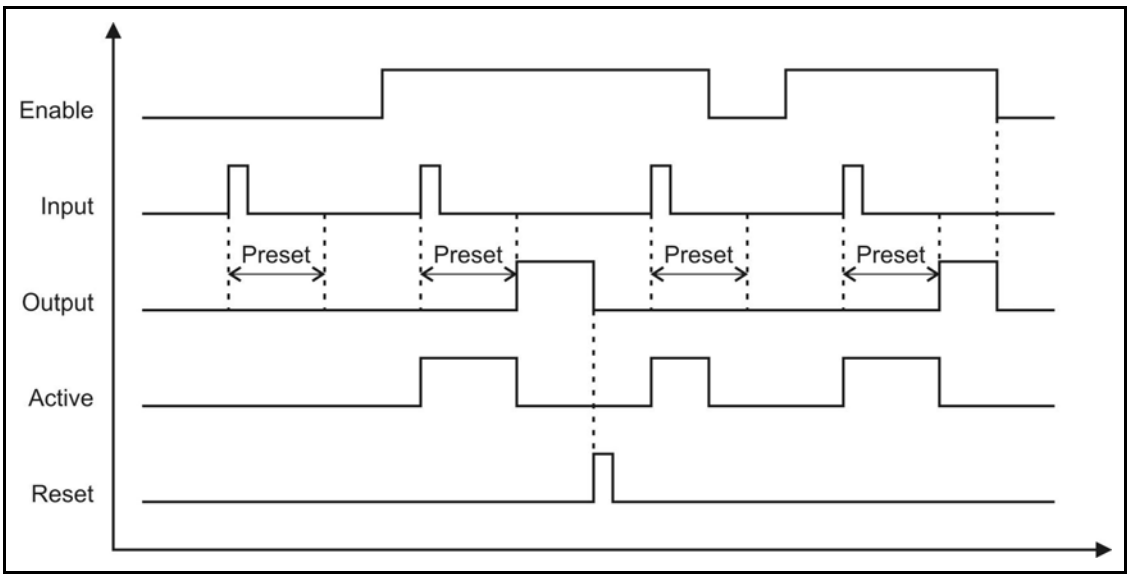


Figure 78: Timer Enable with Edge ON Timer

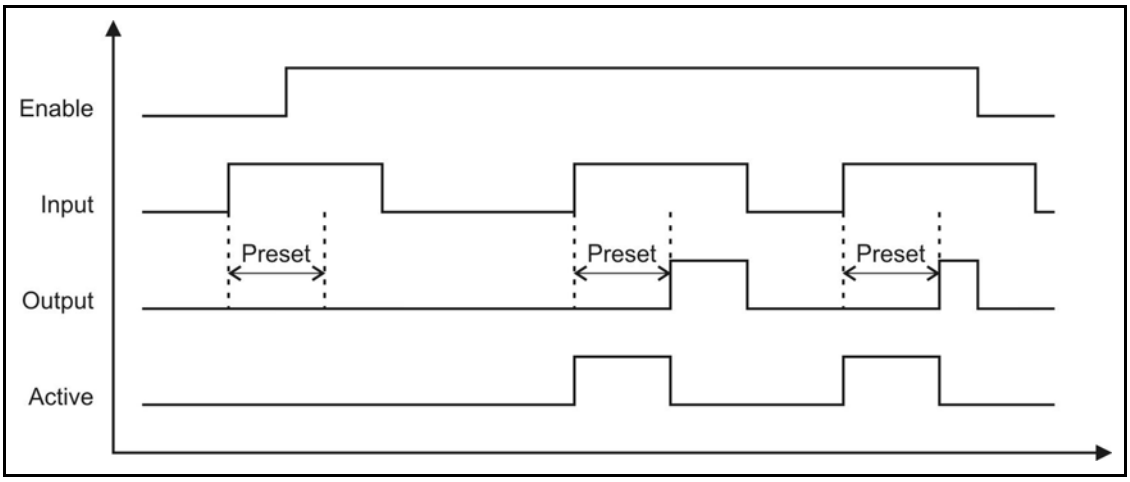


Figure 79: Time Enable with Level ON Timer

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 in Figure 67 through Figure 76 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 in Figures 2 through 10 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.

3.22.2 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.#.Input = ON
```

The user must also turn the Input off as necessary to control the timer:

```
Timer.#.Input = OFF
```

To use the output of a Timer within a program:

```
Wait For Timer.#.Output = ON (or = OFF)
```

To change the value of a Timer Preset in a program:

```
Timer.#.Preset = 12.345 'Seconds
```

To check if a timer is running:

```
If (Timer.#.Active = ON) Then
    Some code
    .
    .
    Endif
```

To wait for a Timer to start running:

```
    Wait For Timer.#.Active = ON
```

Other examples:

```
If((Timer.#.Active=ON) AND (Timer.#.ElapsedTimer<0.500)) Then
    Some code
    .
    .
Endif

If ((Timer.#.Output = ON) AND (Index.#.CommandComplete)) Then
    Timer.#.Reset = ON 'Self-resetting - No need to turn OFF
Endif
```

3.23 Variables View

Variables allow the user to store data related to their system into a parameter, which the user can name. The user must define each user variable by giving it a name, resolution (number of decimal places), and initial value. All variables are signed 32-bit parameters. Setup for the Variables is done on the Variables view located under the Drive /Var group in the Hierarchy Tree. The Variables view is shown in Figure 80.

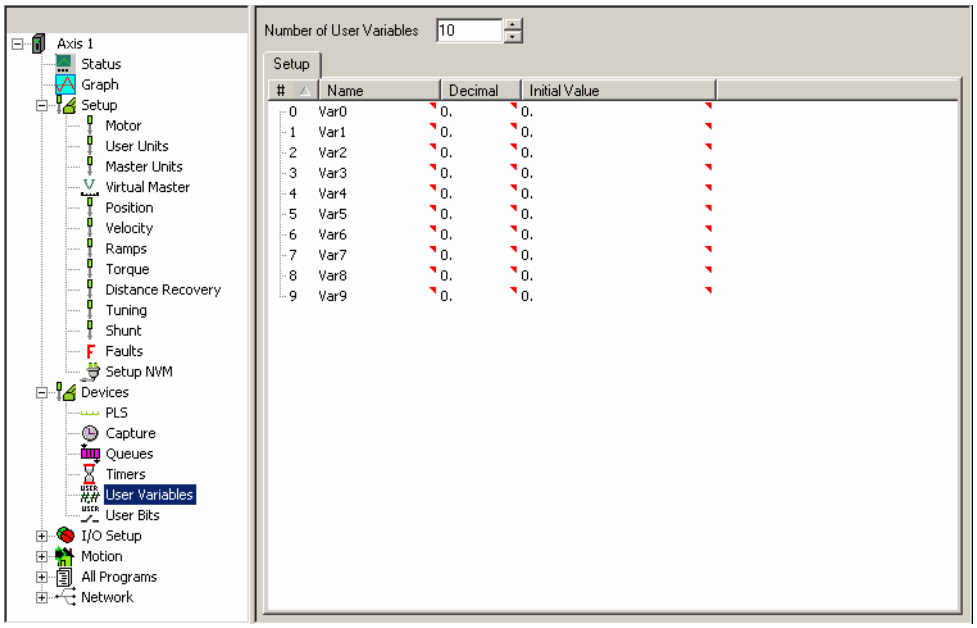


Figure 80: Variables View

The following parameters are part of the User 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 user variable.

Decimal

This parameter (Var.Var#.Decimal) 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 user variable that will be used on power up. If the user variable has been configured as a Save to NVM parameter, then the value in NVM will overwrite the initial value on power up.

3.23.1 Adding and Deleting Variables

The default number of variables is ten. To add more user variables, click on the up arrow next to the “Number of User Variables” box on the User Variables view. The maximum number of user variables is 500.

Only the last variable in the list can be deleted. To delete the last variable, simple click on the down arrow next to the “Number of User Variables” box.

User variables are all of a Global type, meaning that they can be accessed from any program.

Online Tab (not shown)

While online with the Epsilon EP-P drive, an online tab will be shown next to the Setup tab. This online tab will show the current online value of each of the user variables.

3.23.2 Using Variables in a Program

Once setup, user variables can be used inside a program in calculations, motion profile setup, or any other user-desired function. To access user variables, click on the ***Drag in Variables*** button in the user program toolbar. User Variables is a branch in the Select Variables From Tree dialog box.

3.24 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 in an alpha character (non-numeric character). Spaces are not available in the Name for a Bit, however the underscore character ("_") may be used.

The Initial Value for each user bit is configured using the 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 User Bit will be “Off” or “Inactive”.

User Bits are configured on the Bits view as shown in Figure 81.

Bits may be accessed on 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.

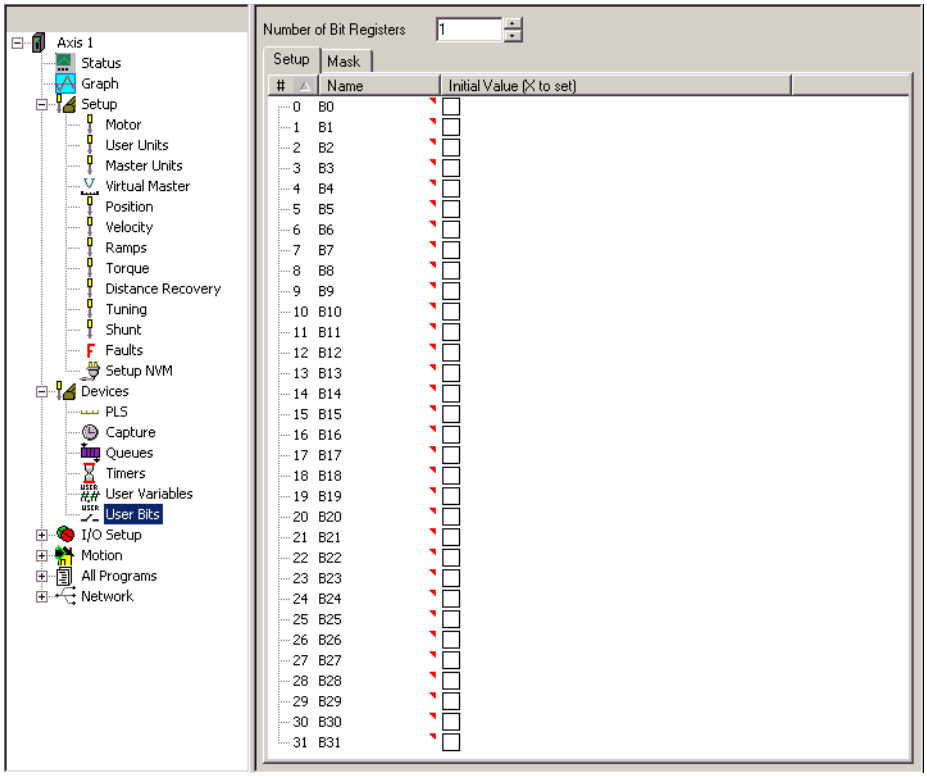


Figure 81: User Bits View

3.24.1 Adding and Deleting User Bits

Bits can be added or deleted in groups of 32-bits. Individual bits cannot be added or deleted. The default number of User Bits available is 32. To add an additional 32 bits, simply click on the up arrow next to the “Number of Bit Registers” box at the top of the User Bits view (see Figure 81).

To decrease the number of User Bits by 32, click on the down arrow next to the “Number of Bit Registers” box. When decreasing the number of User Bits, it is always the last 32 bits in the list that will be eliminated.

3.24.2 User 32-bit Bit Register and User Bit Masking

When using different communications protocols (i.e. DeviceNet, Profibus, Modbus), it is often desirable to access groups of User Bits in a single parameter, rather than having to access them individually. In the Epsilon EP-P drive it is possible to access 32 User Bits in a single parameter. This parameter is named BitRegister.#.Value. Because some of the 32 User 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 User Bit Register (group of 32 User 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 82.

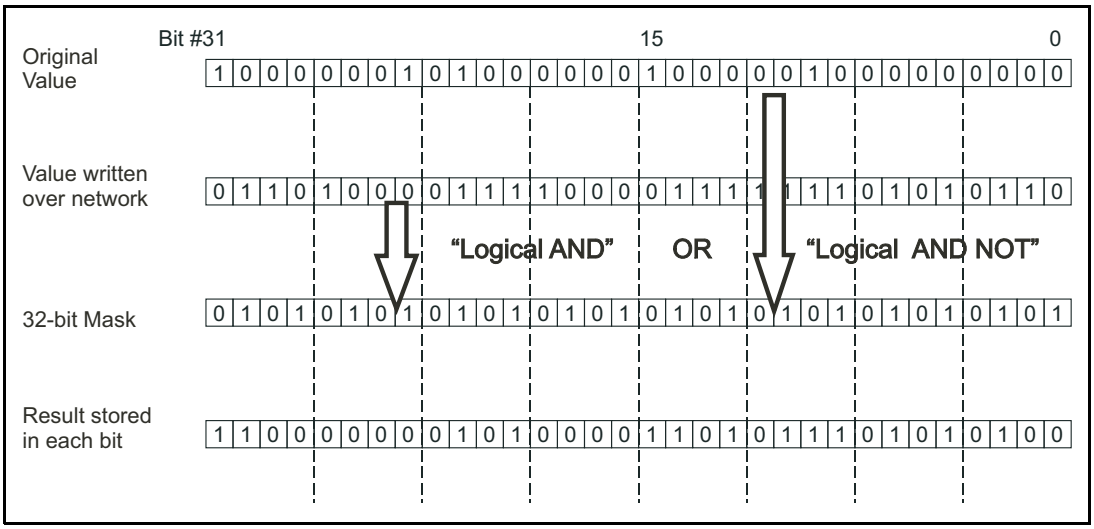


Figure 82: Writing to the User Bit Register

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.

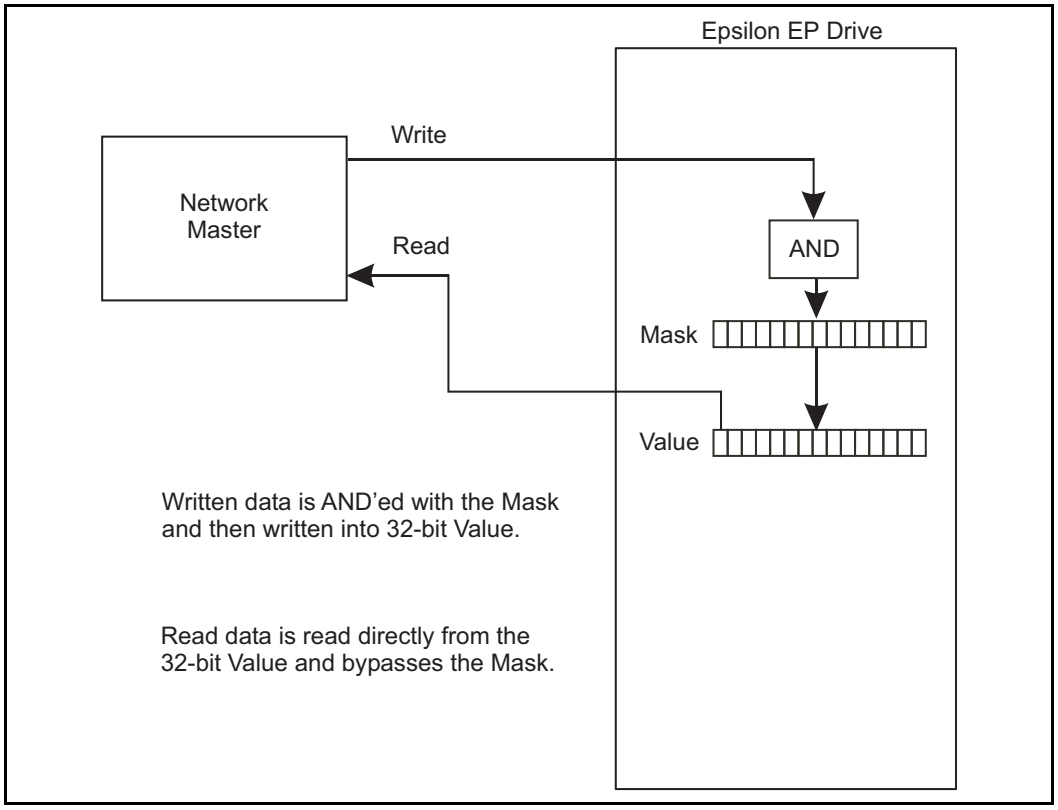


Figure 83: User Bits Read/Write Process

3.24.3 Configuring the User Bit Mask Register

The User Bit Mask is a 32-bit parameter that can be configured through Power Tools Pro, 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 Pro, navigate to the Mask tab on the User Bits view, see Figure 84.

In the User Bits Mask view, 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 User Bits that are added, a new Mask parameter will appear for that group. Mask 0 will control the mask for User 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.

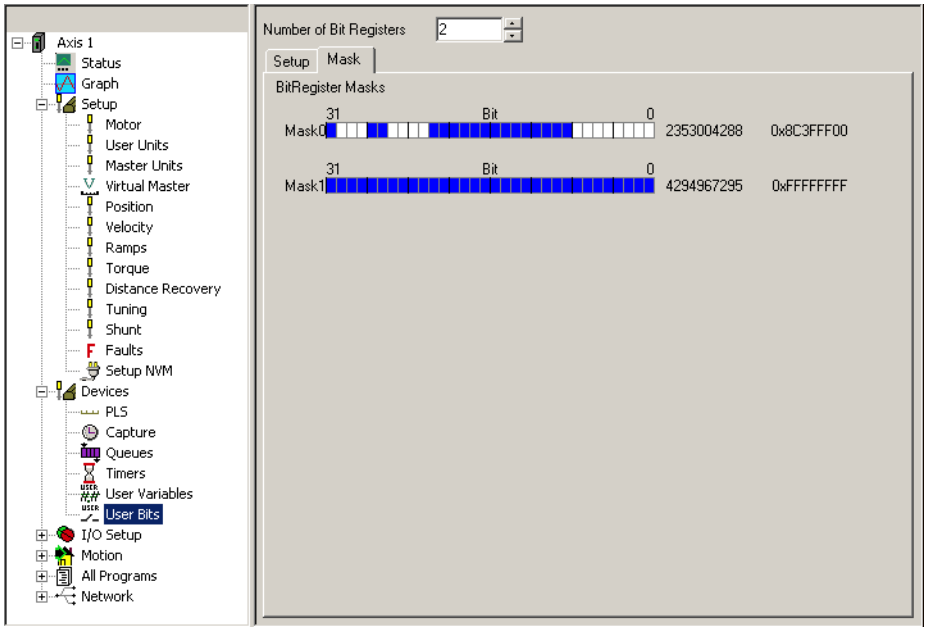


Figure 84: User Bits Mask View

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.1.ValueMask = 0xFFFF0000

This example writes a 1 into all bits of the upper sixteen bits, and 0 into each of the lower sixteen bits using hexadecimal value. To write the same value using decimal notation, the following instruction would be used.

Example: BitRegister.1.ValueMask = 4294901760

This instruction would also write a 1 into each of the upper sixteen bits, and a 0 into each of the lower sixteen bits.

3.24.4 Packed Bits

Packed Bits are 16-bit words configured by the user to read and write groupings of bit-level parameters (booleans 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 85). 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 Control Word in the Figure 85 below would not automatically change to 1. Bit0 will only be 1 if the master device sets that bit when writing to the Control Word.

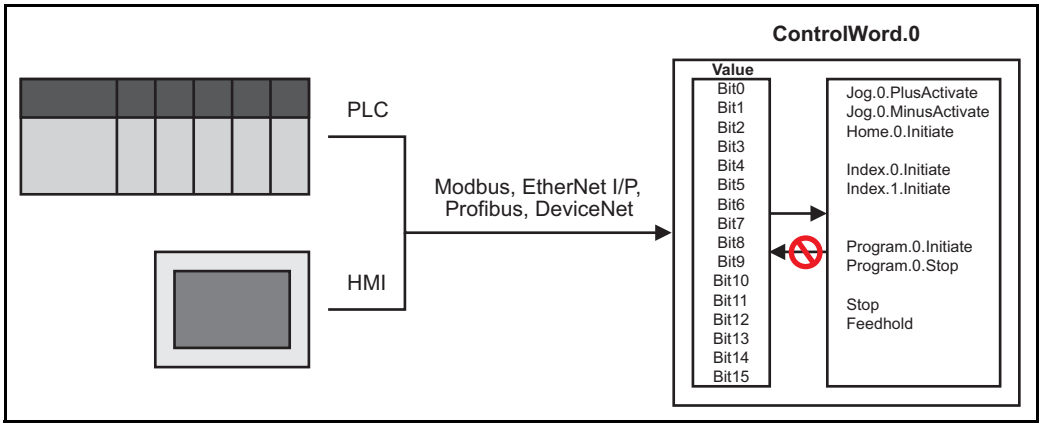


Figure 85: Control Words Communications Example

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 86 below).

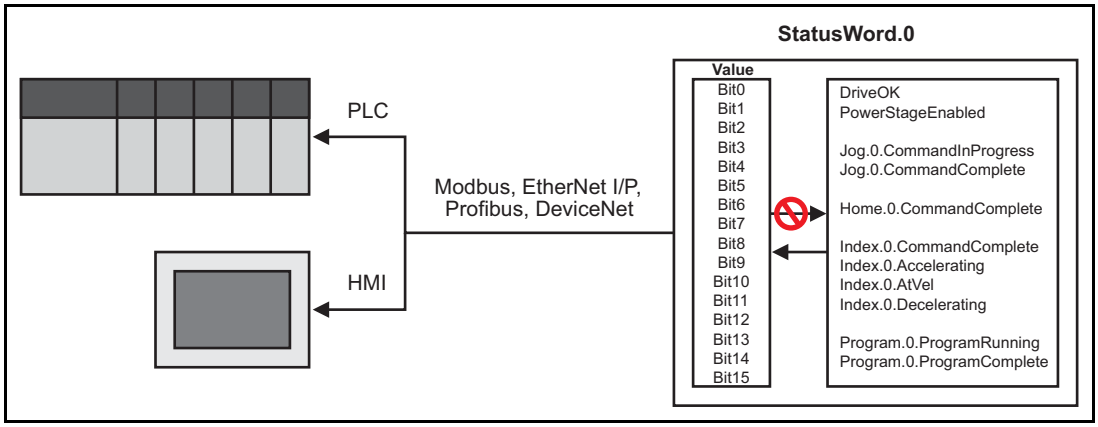


Figure 86: Status Word Communication Example

3.24.5 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 87 below shows the Packed Bits Control Words view onto which the user has dragged several motion initiate parameters.

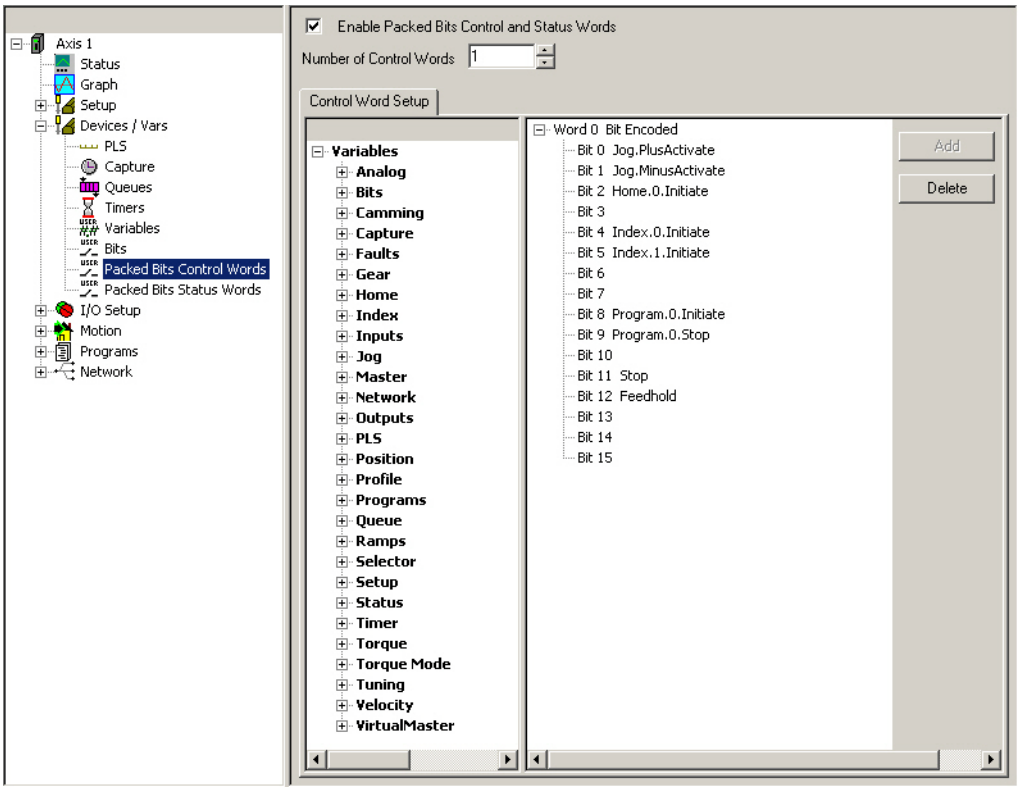


Figure 87: Packed Bits Control Words View

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 checkbox. If the checkbox is clear, the view remains blank (default). If the checkbox is selected, then the mapping tree appears which allows the user to configure the Control Word(s) as desired. This checkbox 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 as 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 87 above) should become enabled. 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 88 below). The data is displayed in binary form for each individual bit of the Control Word(s), as well as hexidecimal and decimal format for each Control Word.

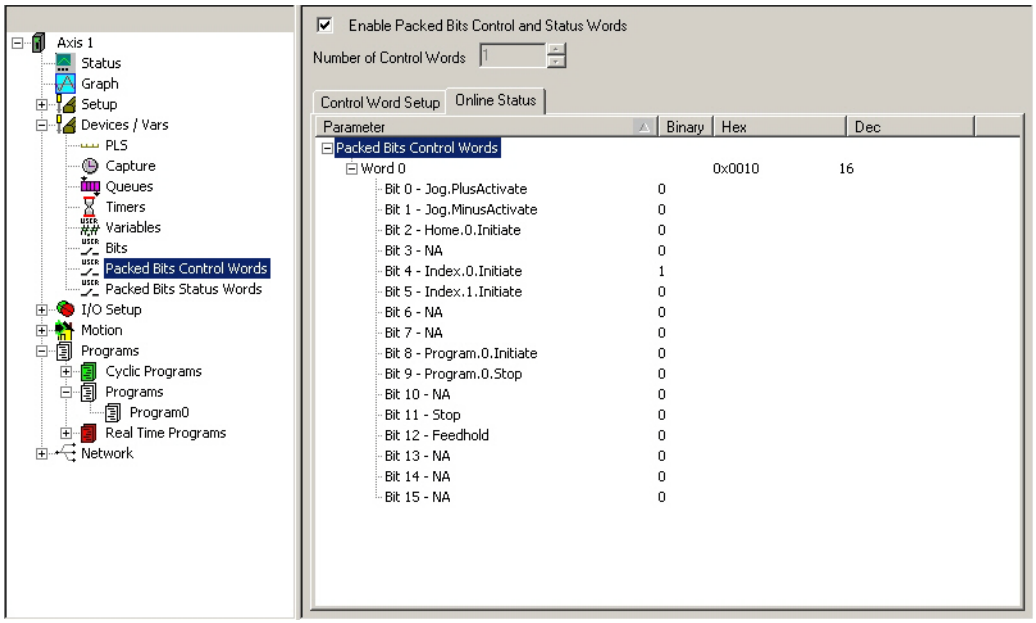


Figure 88: Control Word Online Status View

Using Pack Bits Control Words

Once the Control 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. Figure 89 shows an example in which the user will write to the Control Word 0 and 1 via Ethernet I/P.

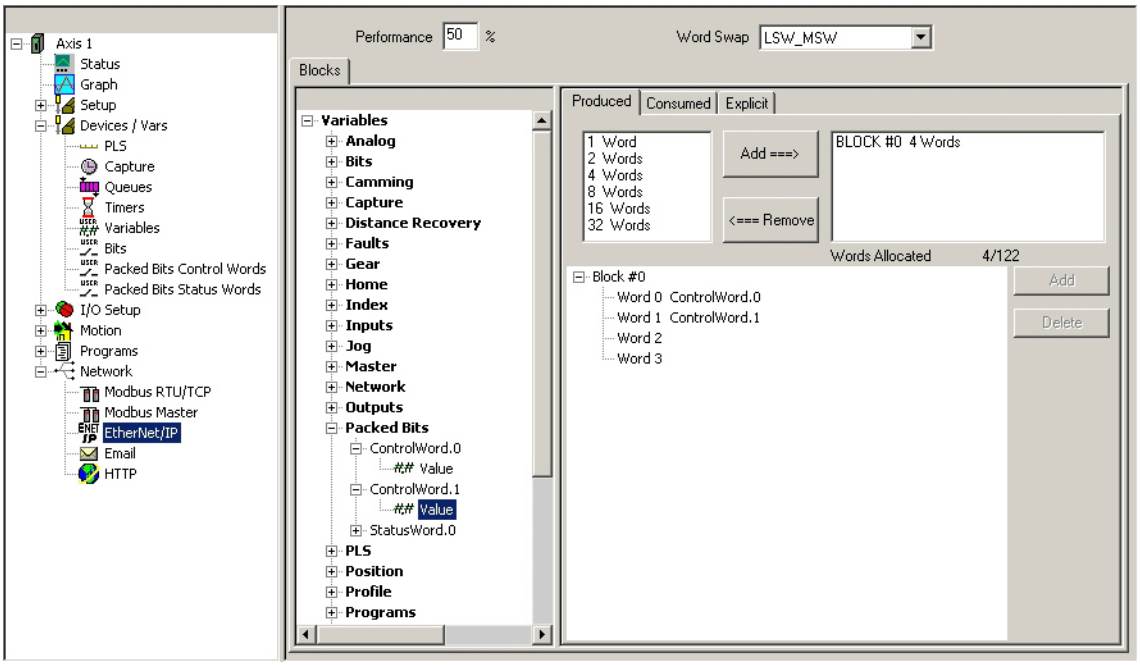


Figure 89: Using Ethernet to Communicate to the Control Words 0 and 1

3.24.6 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 90 below shows the Packed Bits Status Words view onto which the user has dragged several status parameters.

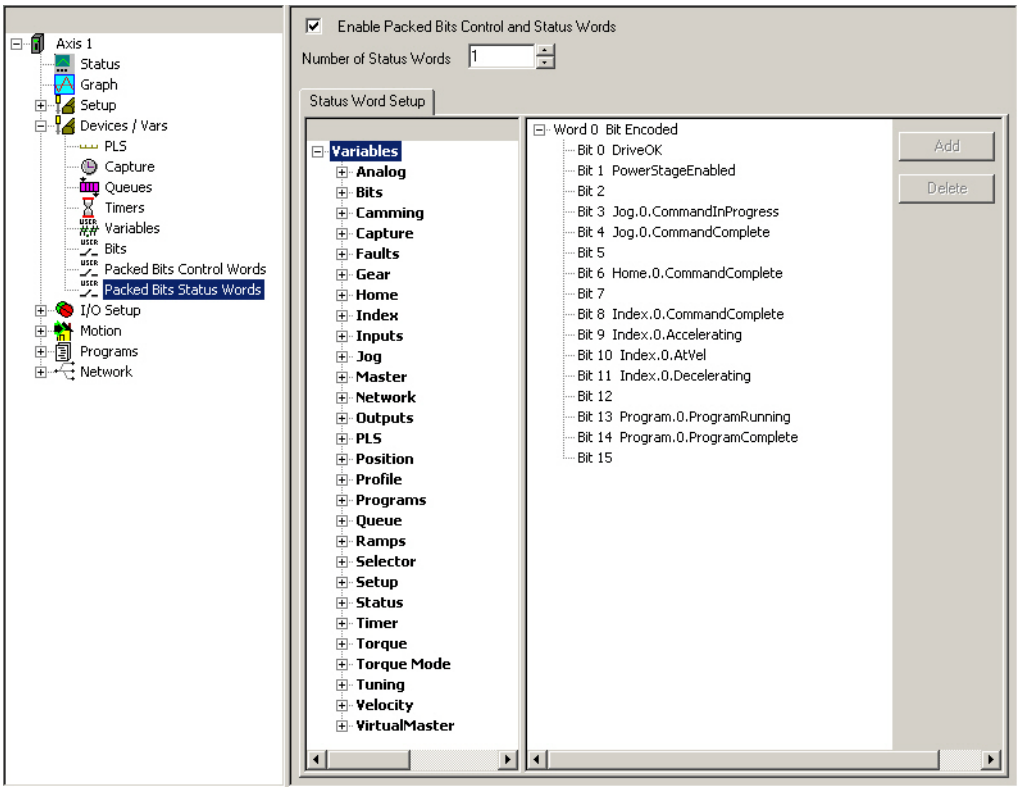


Figure 90: Pack Bits Status Words View

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 checkbox. If the checkbox is unchecked, the view remains blank (default). If the checkbox is checked, then the mapping tree appears which allows the user to configure the Status Word(s) as desired. This checkbox 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 spinner control 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 91). The data is displayed in binary form for each individual bit of the Status Word(s), as well as hexidecimal and decimal format for each Status Word.

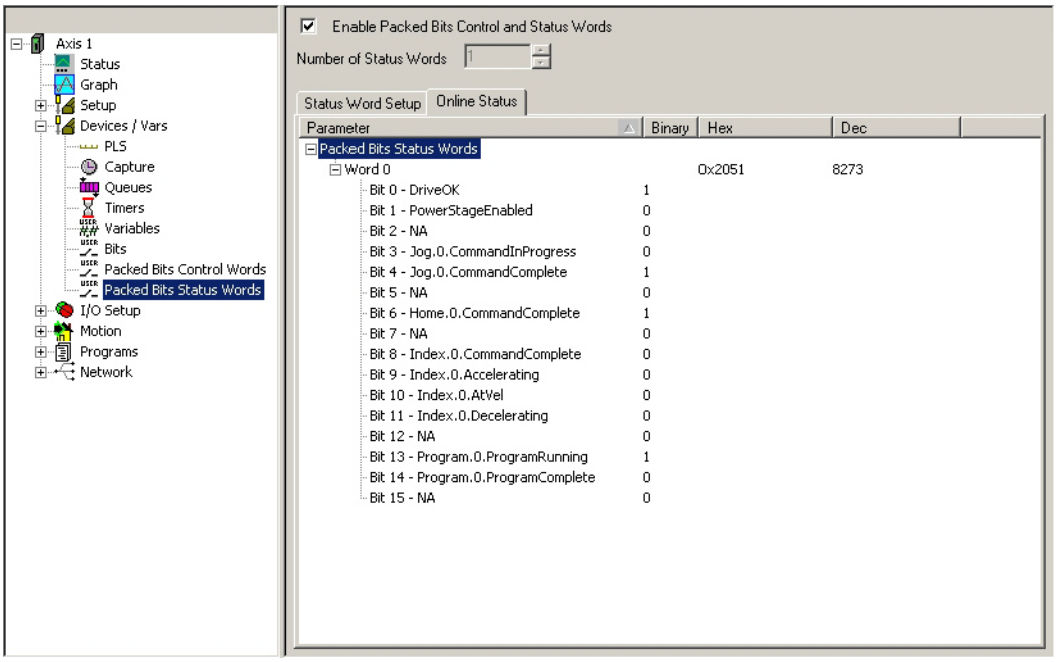


Figure 91: Pack Bits Online Status Tab

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. Figure 92 below shows an example in which the user will read Status Word 0 from the drive via Ethernet I/P.

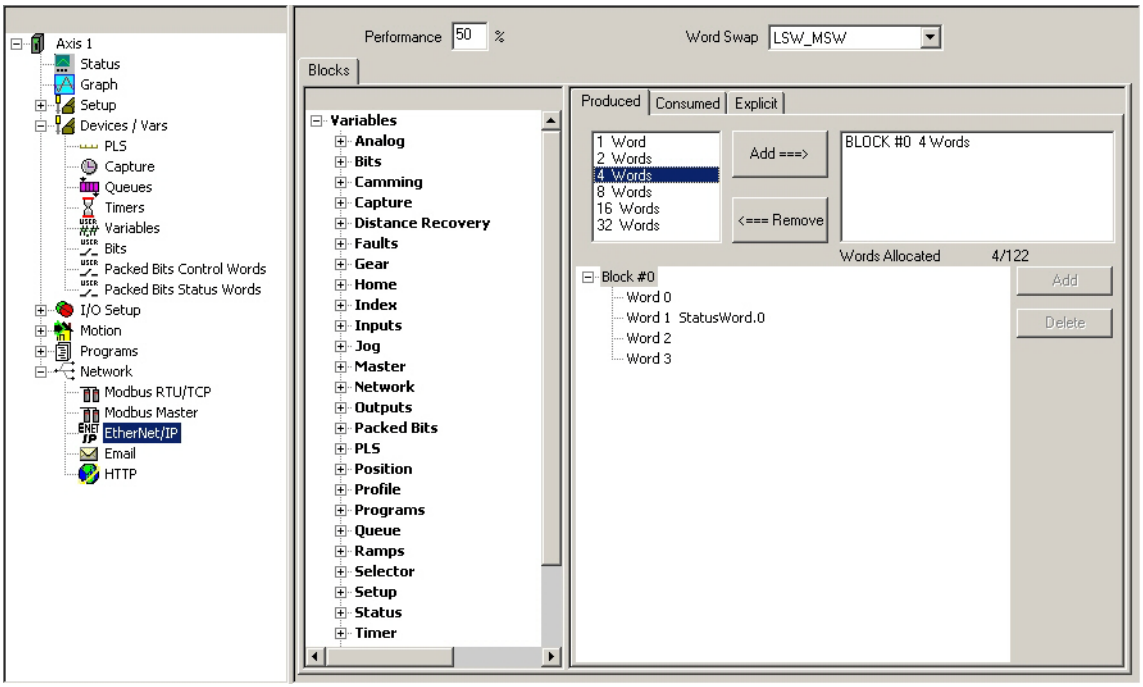


Figure 92: Configuring Ethernet to Communicate Status Word 0 from the Drive

3.25 I/O Setup Group

The I/O Setup group contains views that control input and output functions as well as other drive functions. These views are as follows: Selector, Assignments, Input Lines, Output Lines, Analog Inputs, and Analog Outputs. These can be viewed by expanding I/O Setup then simply clicking on any one of the setup views underneath the I/O Setup.

3.26 Assignments

External control capability is provided through the use of assignments to the Sources (Drive Inputs) or the Destinations (Drive 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. External controllers, such as a PLC or other motion controllers, may be connected to affect or monitor the drive's operation.

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 200 mA for each digital output.

The drive is equipped with sixteen optically isolated input lines (one dedicated to a Drive Enable function) and can be accessed through the 26-pin dsub connector (J3) located on the front of the drive.

3.27 Assignments View

The Assignments View not only displays information but also makes assignments regarding the source and the destination.

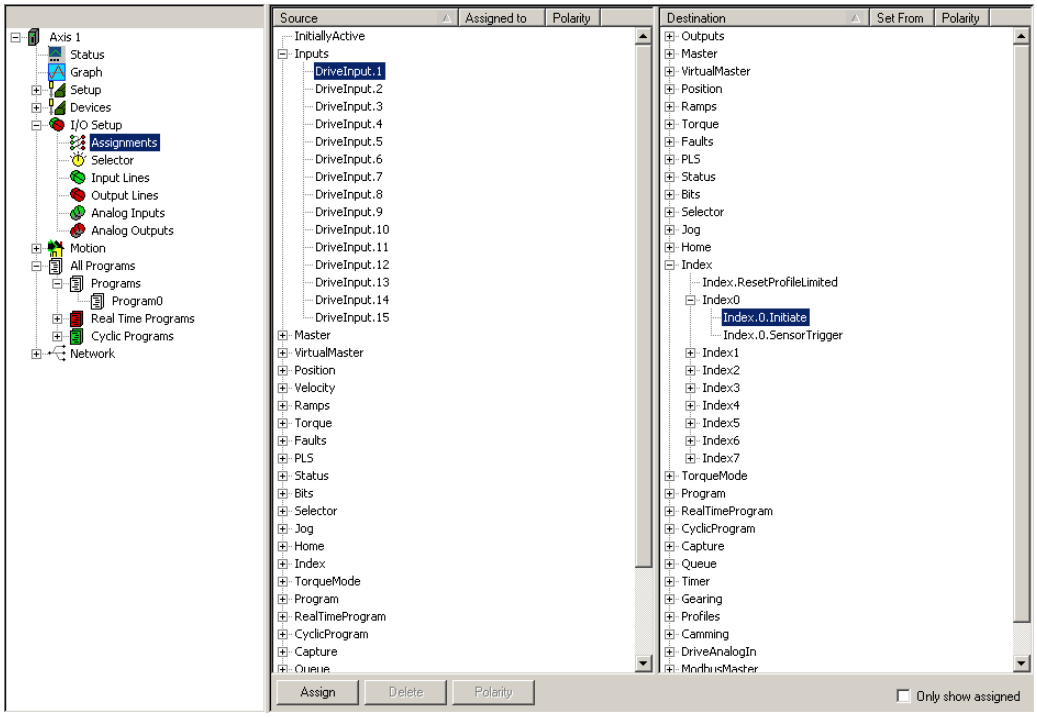


Figure 93: Assignments View

The Assignments View is used to tie a source to a destination. Destinations 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 events that occur in the drive. These events are based on drive activity. By expanding individual groups, you will see more detailed parameters. For example, in an Epsilon EP-P drive configuration if you expand the Inputs source group, you will see DriveInput.1 through DriveInput.15, as shown in Figure 93. You can use these events to trigger certain actions (or destinations) on the right side of the view.

To make an assignment, a source must be tied to a destination. Any source can be tied to any destination to create the desired system operation.

3.27.1 Creating An Assignment

Various methods can be used to tie a source (such as DriveInput.1) to a destination, (such as Index.0.Initiate).

Drag and Drop Method

First, position the mouse pointer over the source on the left to assign to the destination on the right. Press the left mouse button while over the source, and hold the button down. While holding the left button down, drag the source until the pointer is positioned over the desired destination and release the left mouse button.

Destinations can also be dragged over to sources, see Figure 94.

Assign Button Method

Click on both the source and destination you wish to assign to each other. Once both are selected, the ***Assign*** button in the lower left corner of the view will become enabled. Click the ***Assign*** button to complete the assignment. Figure 93 shows the source and destination highlighted, and the ***Assign*** button available to click on.

Once an assignment has been made, you will see the “Assigned to..” and the “Set From” columns filled in for the specific sources and destinations. This indicates what destination(s) an individual source has been assigned to, and what source(s) an individual destination is assigned to.

Any source can be assigned to up to ten different destinations maximum. Any destination can have as many sources as desired assigned to it.

3.27.2 Deleting An Assignment

Delete Button Method

Click on the source or destination you wish to delete. Once selected, the ***Delete*** button will become available. Click the ***Delete*** button to remove the assignment.

Right Click Method

Position the pointer over the specific assignment to delete then right click. A selection box will appear. From this selection box, choose Delete.

After either of these procedures, the assignment will disappear.

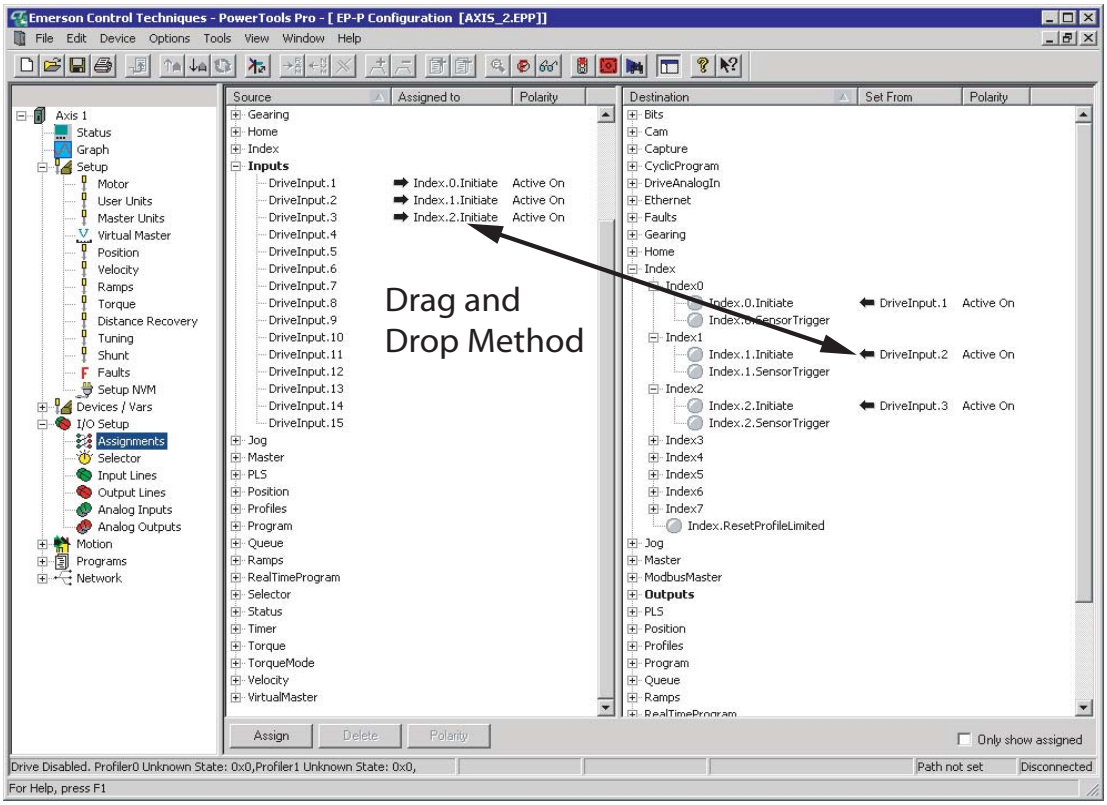


Figure 94: Tying a Source to a Destination

3.27.3 Assignment Polarity

The active state of an assignment can be programmed to be Active Off, Active On, or Custom using PowerTools Pro. Making an assignment “Active On” means that the destination will be 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 is 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 deactivate based on two different sources.

NOTE

Destination functions which initiate motion (Jog.PlusInitiate, Jog.MinusInitiate, Index.#.Initiate, and Home.#.Initiate) cannot be set “Active Off”.

Default polarity for a new assignment is Active On. Two methods will change the polarity of an assignment.

Polarity Button Method

Click on either the Source or the Destination you wish to change the polarity of. Once highlighted, the ***Polarity*** button will become available in the lower right corner of the view. Click on the ***Polarity*** button and change the settings as desired in the Polarity dialog box. Click ***OK*** to apply the changes.

Right Click Method

Position the pointer over the specific assignment you with to change polarity of and click the right mouse button. A selection box will appear. From this selection box, choose Polarity. the Polarity dialog box will appear. Change the Polarity settings as desired and click OK to apply the changes.

3.27.4 User Level

The User Level filters the available assignments. The User Level is changed on the Options menu at the top of the PowerTools Pro toolbar. Choose **Options**/**Preferences**/**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.

3.27.5 Only Show Assigned Check Box

When this check box is selected it removes the unassigned sources and destinations from the view. It allows the user to quickly see how many sources and destinations have been assigned.

3.27.6 Assignments that Automatically Use Position Capture

Certain assignments (Sources or Destinations) automatically generate position capture data internally for greater performance and accuracy. This captured data is used by the Epsilon EP-P drive, but is not directly available to the user. Following is a list of assignments that automatically generate or use captured data.

Sources that generate capture data

Drive Inputs 1-15

The Epsilon EP-P Inputs are constantly monitored by the processor, and when activated will automatically capture related data. The processor controls all resetting requirements. The capture only occurs on the rising edge of an input. When the input is activated, the captured data will automatically be passed to the destination that it is assigned to. The destination may then use the captured data to accurately initiate motion (if it is a motion-related destination).

Motor Encoder Marker

The rising edge of the motor encoder marker pulse will automatically capture data. This will allow the user to accurately initiate motion on the rising edge of the motor encoder marker pulse. The falling edge of the marker pulse does not capture data.

Master Encoder Marker

The rising edge of the master encoder marker pulse will automatically capture data. This allows the user to accurately initiate motion on the rising edge of the master encoder marker pulse. The falling edge of the marker pulse does not capture data.

Index/Jog Command Complete

Activation of the command complete signal at the end of indexes and jogs will automatically capture data. A subsequent index, jog, or dwell can then use the captured data to start itself extremely accurately at the end of the previous motion.

Index/Jog At Velocity

Activation of the command complete signal at the end of indexes and jogs will automatically capture data. A subsequent index, jog, or dwell can then use the captured data to start itself extremely accurately at the end of the previous motion.

PLS Status

A rising or a falling edge of a Global PLS will automatically capture data for use in initiating motion. In order to accurately initiate motion from a Global PLS, an assignment can be made from PLS.#.Status to the initiate destination.

Destinations that use captured data:

Index/Jog Initiates

When one of the sources listed above is assigned to an Index or Jog 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 get the accurate stopping position for the registration index.

3.28 Selector View

The Selector view is located under I/O Setup in the Hierarchy Tree on the left of the view.

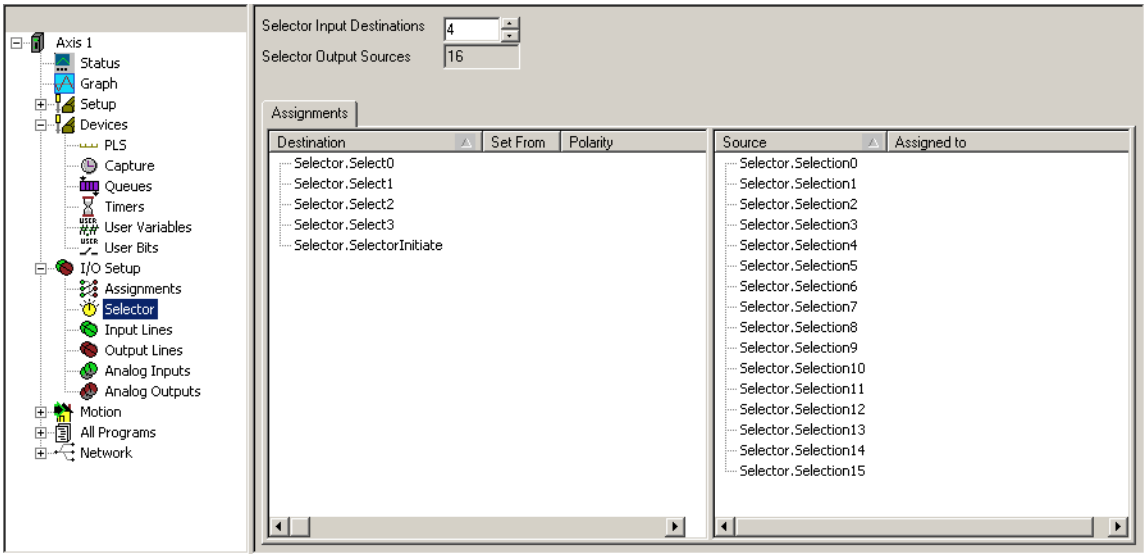


Figure 95: Selector View

The selector allows conservation of the number of input lines by using a binary input conversion to decimal. The binary select lines are set up by assigning sources to the Selector.Select destinations on the Assignments view. In most cases, hardware inputs are assigned to the Selector.Select functions (see Figure 96).

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, the Selector Input Destinations scroll box defines how many binary select lines will be used. The number of Selector.Selections is a direct result of the number of select lines. The formula is as follows:

# of selections = 2^n where n is the number of select lines.

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 must then be made in the Assignments view.

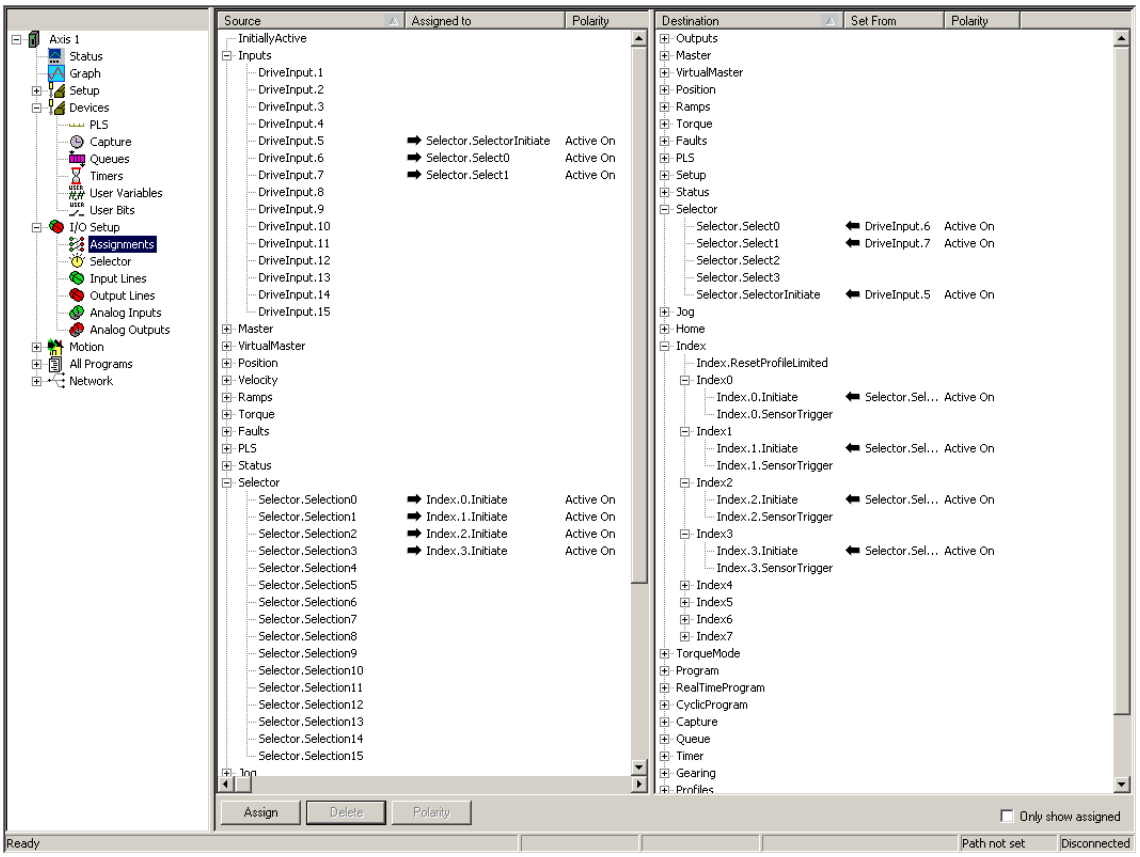


Figure 96: Assignment View

For example, if we entered 3 for the number of Selector Input Destinations, we would have 8 selection lines (Selector.Selection0 through Selector.Selection7). The Selector.Selection number that is activated is determined by the status of the Selector.Select lines when the Selector.Selector Initiate bit is activated. Each select line has a specific binary value.

The binary value is determined as follows:

$S_n \times 2^n$ where S_n = 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: 1

If Selector.Select2 is active, Selector.Select1 is inactive, and Selector.Select0 is active, then the total binary value is as follows:

$S_2 = 1$, $S_1 = 0$, and $S_0 = 1$. 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, then Selector.Selection5 will activate.

Example: 2

If Selector.Select2 is inactive, Selector.Select1 is active, and Selector.Select0 is active, then the total binary value would be as follows:

$S_2 = 0$, $S_1 = 1$, and $S_0 = 1$. 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, Selector.Selection3 would 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 96 shows the four selection lines being tied to Index 0 through Index 3 initiates. By doing this, we could initiate up to four indexes with only two select lines and a selector initiate. This can help minimize the number of inputs required to initiate a large number of indexes or programs.

3.29 Input Lines View

The Input Lines View displays any functions that have been assigned to the drive hardware inputs. See Figure 97.

NOTE

No assignments can be made using the Input Lines View, assignments are only displayed in the Input Lines View.

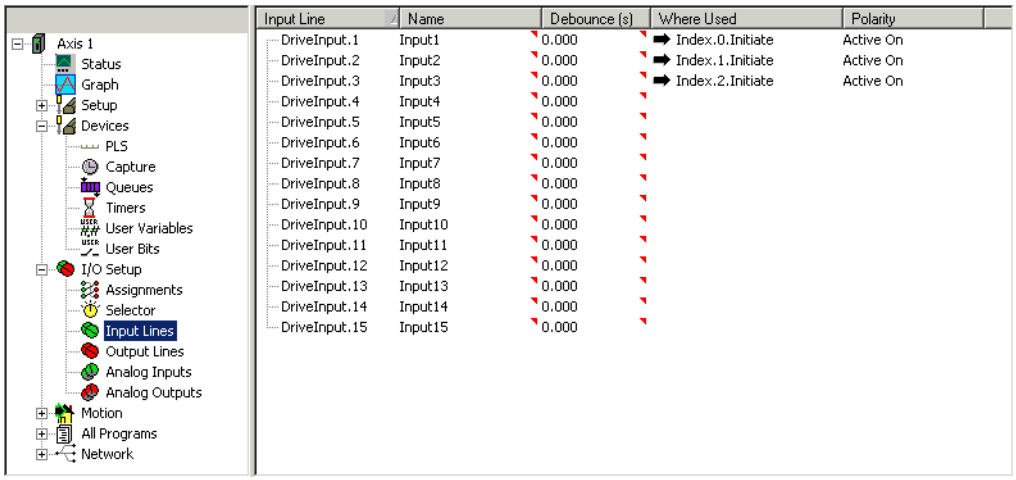


Figure 97: Input Lines View

The following two functions can be performed on the Input Lines view.

Name

You can assign a descriptive name to each input and make the setup easier to follow. The length of the text string is limited to a maximum of 12 characters. Simply double click on the Name field of any input line to assign a name to it.

Debounce

You can program a “Debounce Time” to any input line, which means the motion profile will need to be steady for at least the debounce time before it is recognized. This feature helps prevent false triggering in applications in noisy electrical environments. At the end of the debounce time, the next action can occur.

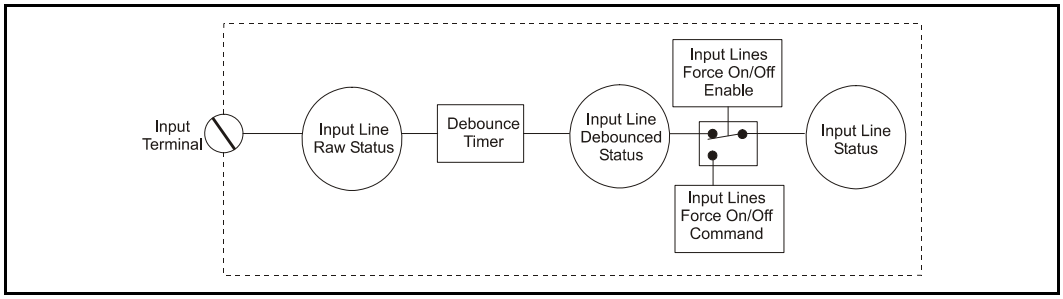


Figure 98: Input Line Diagram

If the Input Line attached to the home sensor is debounced, the actual rising edge of the Home Sensor is used to determine the Home Reference Position (the debounce time ensures a minimum pulse width).

3.30 Output Lines View

The Output Lines View displays any functions that have been assigned to the drive hardware outputs. See Figure 99.

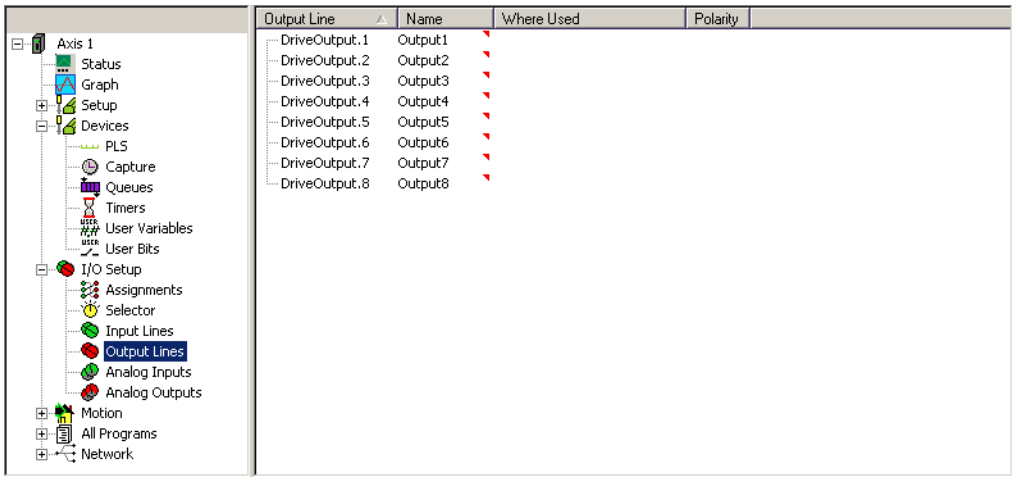


Figure 99: Output Lines View

Names

Descriptive text names can be assigned to individual output lines to make the setup easier to follow.

3.31 Analog Inputs View

The analog input is scaled from +10 V to -10 V range to either the units of the selected variable or to the defined units. A linear scale with offset is defined by entering two points on the scale, a min and max to correlate the user unit to analog voltage. The actual minimum and maximum range of either user unit or analog voltage does not need to be entered as the algorithm will extrapolate the range.

The drive has one Analog input channel that accepts a +10 V to -10 V signal. The drive has a 12-bit over sampled to 14-bit analog to digital converter (A/D), which is used to transform the analog voltage to a usable parameter in the drive. The analog input is scanned by the drive every 100 microseconds.

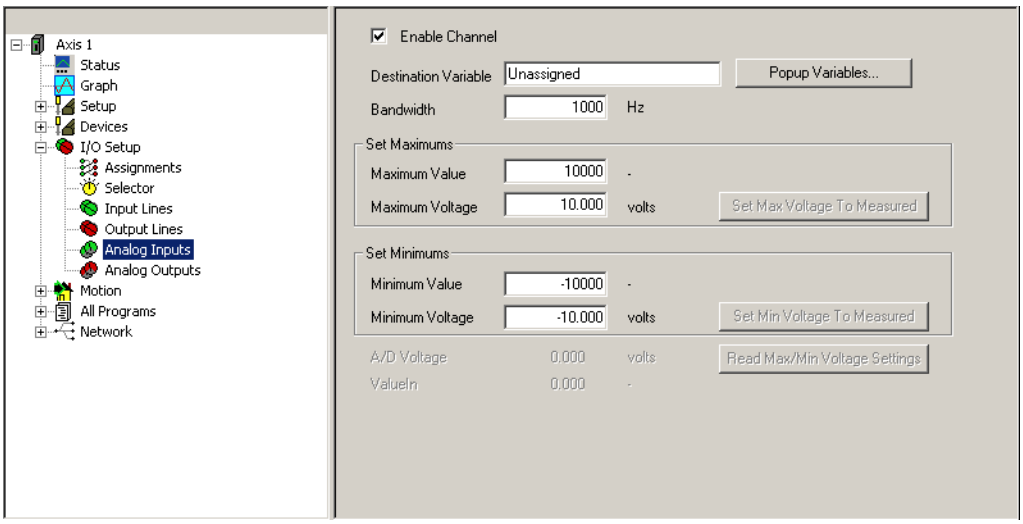


Figure 100: Analog Inputs View - Enabled

3.31.1 Enable Channel Check Box

By default, the analog input channel is not enabled meaning that the drive is not reading the A/D value read by the analog circuit. If the check box is clear, the channel is not enabled and the configuration parameters for the analog input are unavailable and therefore have no effect.

To enable the analog input, simply select the Enable Channel check box, and the configuration parameters will become available to edit. With the channel enabled, the trajectory loop update will transfer data from the drive into DriveAnalogInput.RawValue as volts and into DriveAnalogInput.ValueIn scaled.

Destination Variable

The Destination Variable (DriveAnalogInput.CustomDestinationVariable) can be entered one of two ways: by just typing any parameter using the program format (dot notation) 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 Destination Variable text box.

Bandwidth

This parameter (DriveAnalogInput.LowPassFilterFrequency) sets the low-pass filter cutoff frequency applied to the analog input. Signals exceeding this frequency will be filtered at a rate of 20 db per decade.

3.31.2 Set Maximums Group

Maximum Value

This parameter (DriveAnalogInput.MaxUserValue) is used for user unit scaling. Enter the maximum value in analog user units to which the maximum analog voltage should correspond.

Maximum Voltage

Enter the maximum voltage that will be seen on the analog input terminals. The user can enter the value in this text box (DriveAnalogInput.MaxVoltageValue) by hand, or set the analog source to it's maximum value with just a click of the "Set Max Voltage to Measured" button next to the text box.

Set Max Voltage To Measured Button

Click this button to read the current value on the analog channel and enter it into the Maximum Voltage text box.

3.31.3 Set Minimums Group

Minimum Value

Enter the minimum value in analog user units that the minimum analog voltage should correspond to.

Minimum Voltage

Enter the minimum voltage that will be seen on the analog input terminals. The user can enter the value in this text box (DriveAnalogInput.MinVoltageValue) by hand, or set the analog source to it's minimum value with just a click of the "Set MinVoltage to Measured" button next to the text box.

Set Min Voltage To Measured Button

Click this button to read the current value on the analog channel and enter it into the Minimum Voltage text box.

3.31.4 A/D Voltage

This parameter (DriveAnalogInput.RawValue) is visible while online. It is the raw analog input in Volts.

3.31.5 ValueIn

This parameter (DriveAnalogInput.ValueIn) is visible while online. This is the results of the analog value scaled to the user unit value.

3.31.6 Read Max/Min Voltage Settings

This button when pressed will read the Min and Max voltages from the drive and the voltages are entered into the Min and Max Voltage text boxes in the configuration file.

3.32 Analog Outputs View

The drive has two 10-bit Analog Outputs that may be used for diagnostics, monitoring or control purposes. These outputs are referred to as Channel 1 and Channel 2. They can be accessed from the analog/sync output connector (J5) on the Epsilon EP-P drive.

The analog output is scaled from the units of the selected variable to the +10/-10 volt range of the Analog Output. It is a linear scale with offset defined by entering two points on the scale, a minimum and maximum to correlate the user units to analog voltage. The actual minimum and maximum range of either user unit or analog voltage does not need to be entered as the algorithm will extrapolate the ranges.

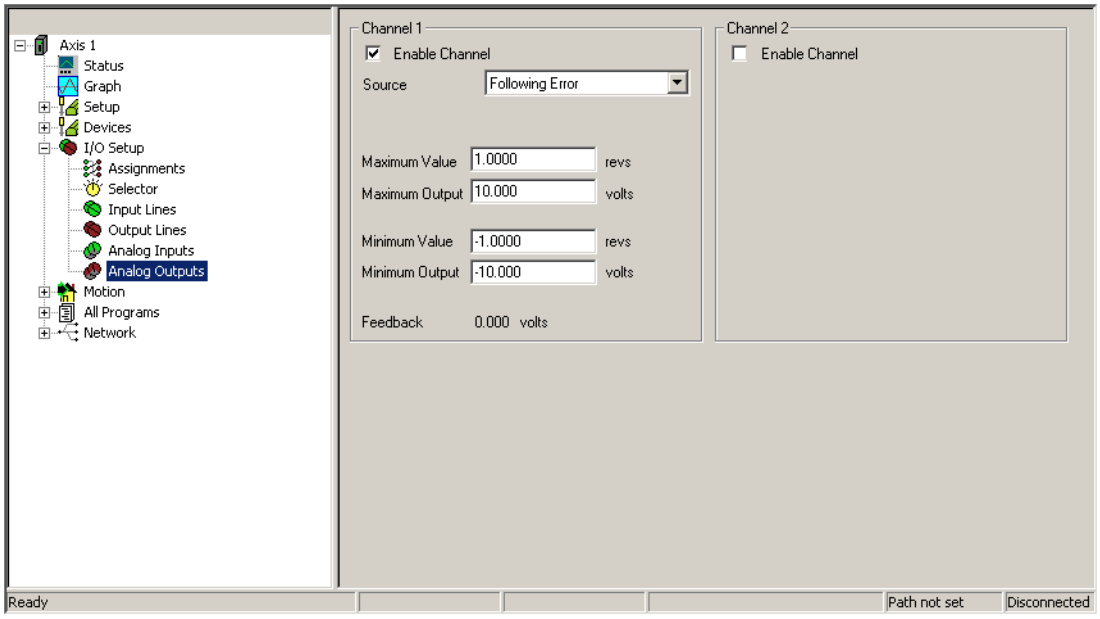


Figure 101: Analog Outputs View

3.32.1 Enable Channel Check Box

By default, the analog output channels are not enabled meaning that a value is not being sent to the analog circuit. When the channel is disabled (check box is clear), the configuration parameters for that analog output are unavailable and therefore have no effect. To enable the output, simply select the Enable Channel check box, and the configuration parameters will become available to edit.

If the user wishes to control the Analog Output through other means, it is necessary to clear the Enable Channel check box.

3.32.2 Source

This list box allows the user to create a direct connection from a Source parameter to the Analog Output. 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.

Custom Variable

The Custom Variable parameter is only available once the user has selected Custom Variable from the Source list box above. This text box is used to define what parameter will control the Analog Output. The selected parameter will directly determine the value of the analog output based on the Max and Min scaling values entered on this view. The parameter is entered here in program format or see Popup Variables Button, below.

Popup Variables Button

Click this button to open the Select Variables from Tree window. Select the variable and drag the variable over to the Custom Variable text box to assign the variable to the Analog output channel.

3.32.3 Maximum Value

The analog output is a linear interpolation of the selected drive variable between the minimum and maximum specified end points. Each end point is specified as the user value and the corresponding output value at that point. The number of decimal places for both values is taken from the selected drive variable. MaxUserValue is the maximum user unit value which corresponds to the maximum analog output value.

3.32.4 Maximum Output

The analog output is a linear interpolation of the selected drive variable between the minimum and maximum specified end points. Each end point is specified as the user value and the corresponding output value at that point. The number of decimal places for both values is taken from the selected drive variable. MaxOutputValue is the maximum analog output value which corresponds to the maximum user value.

3.32.5 Minimum Value

The analog output is a linear interpolation of the selected variable between the minimum and maximum specified end points. Each end point is specified as the user value and the corresponding output value at that point. The number of decimal places for both values is taken from the selected variable. MinUserValue is the minimum user unit value which corresponds to the minimum analog output value.

3.32.6 Minimum Output

The analog output is a linear interpolation of the selected variable between the minimum and maximum specified end points. Each end point is specified as the user value and the corresponding output value at that point. The number of decimal places for both values is taken from the selected variable. MinOutputValue is the minimum analog output value which corresponds to the minimum user value.

3.32.7 Feedback

Analog Output Feedback is the Analog output voltage to be sent out after scaling the selected source parameter.

3.33 Motion Group

All motion parameters related to Jogs, Homes, Indexes, Gearing and Camming are located in the Motion hierarchy group.

Motion views will use units that correspond to Realtime or Synchronized motion. This choice is made on each motion view. The units are customized in the Setup Group: Realtime units are defined on the User Units View, and Synchronized units are defined on both the User Units View and the Master Units View.

Each of the motion views, when online, have an Online tab that displays feedback information and provides buttons to initiate motion.

3.34 Jog View

Jogging produces rotation of the motor at controlled velocities in a positive or negative direction. The jog is initiated with the Jog.#.Initiate destination or from a program.

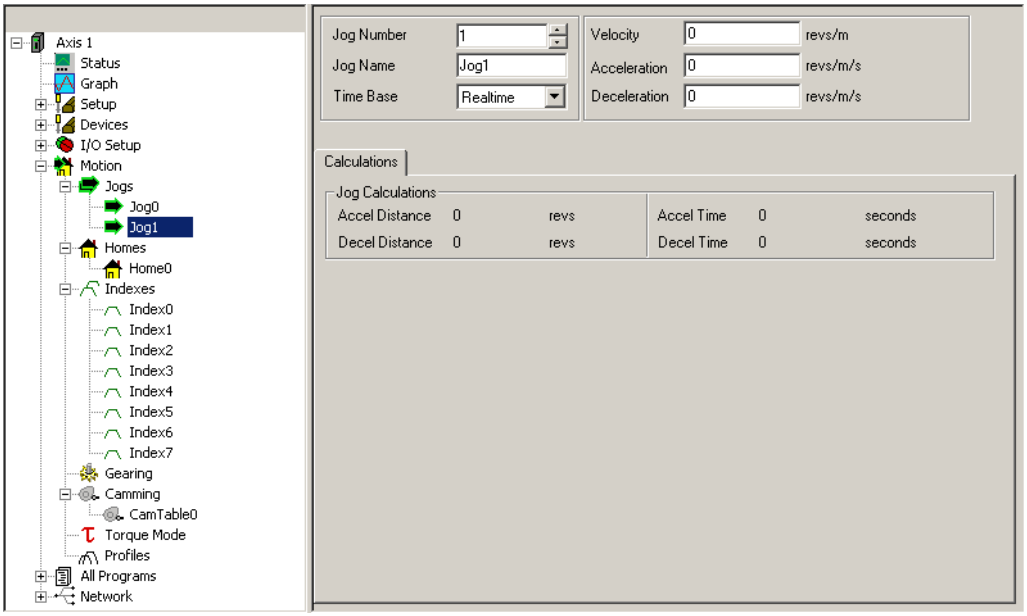


Figure 102: Jog View

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 between a jog that is based on time (Realtime) as defined by user units, normally in seconds, or a time based on Master position via an external encoder (Synchronized) set in the Master Units View.

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 Pro.

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 Pro.

3.34.1 Jog Sources and Destinations

Sources

The following source functions can be found in the Assignments view under the Jog setup group:

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 Jog 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.

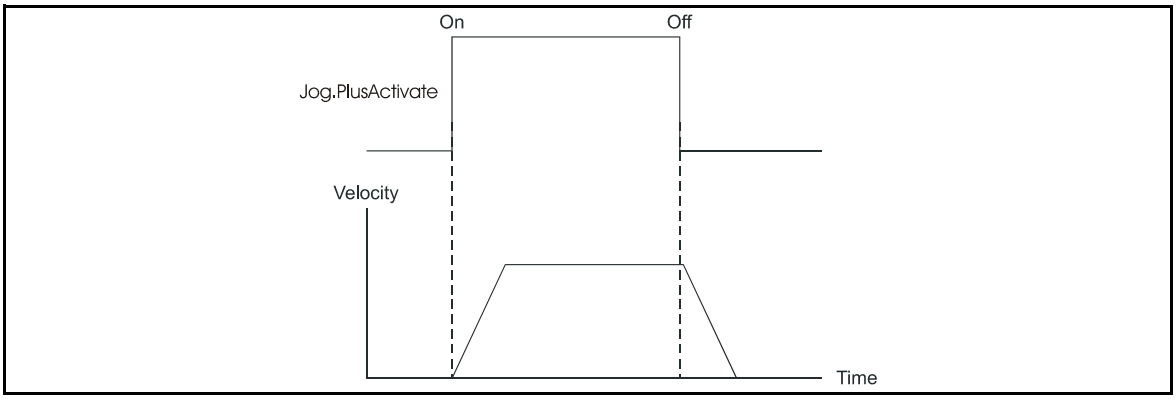


Figure 103: Jog Activation

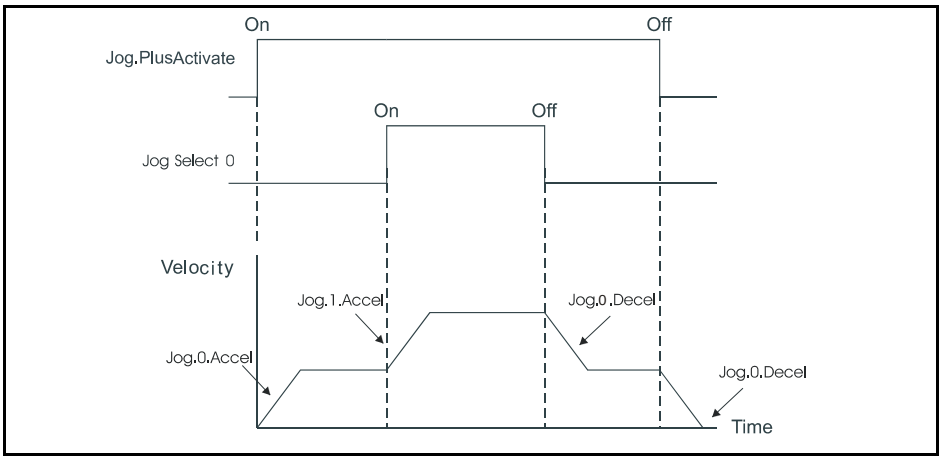


Figure 104: Jog Select Details

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 “R”, for ready.

If the drive 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.

If the motor reaches a Travel Limit, you can Jog off the Travel Limit in the opposite direction. (Use Jog.PlusActivate to move off a Travel Limit -).

3.35 Home View

The Home is used in applications in which the axis must be precisely aligned with some part of the machine. The Home is initiated with the Home.#.Initiate Destination or from a program.

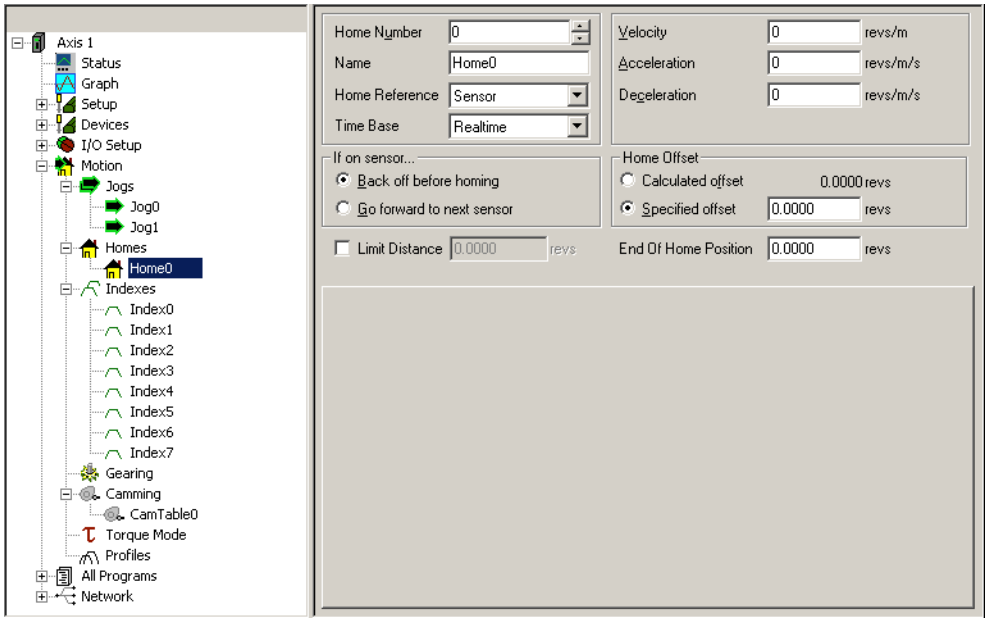


Figure 105: Home View

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 the user to assign a descriptive name to the home sequence up to 10 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 'Marker then Sensor'. 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. Real-time and sync 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 1600µs. 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

LimitDistEnable

This check box enables the specified Home 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.

Figure 106 is a diagram of a home using the "Back off before homing" radio box, a Home Reference of "Sensor", and using a

"Calculated Offset".

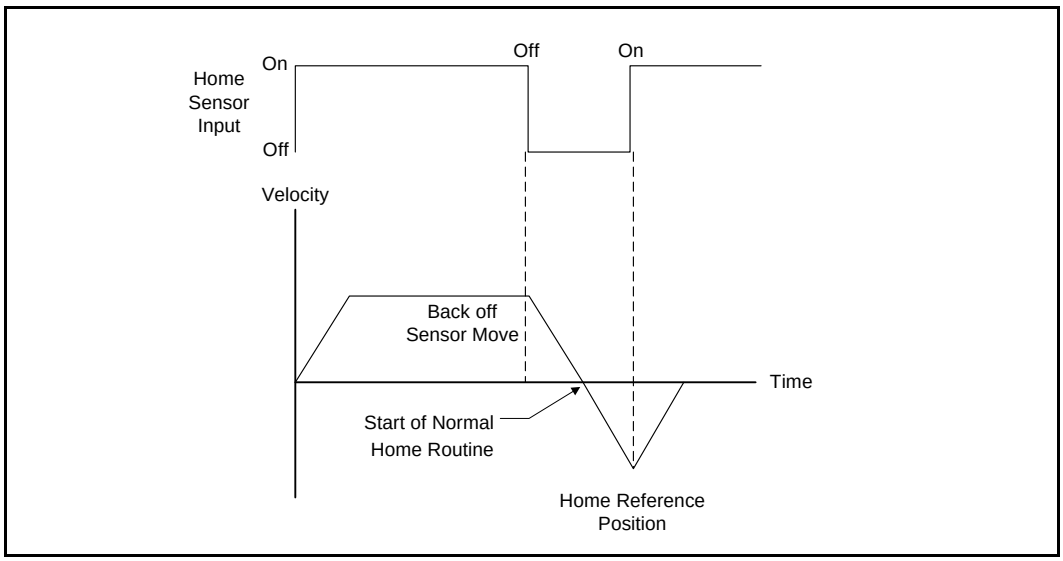


Figure 106: Home Reference Position

3.35.1 Home Sources and Destinations

Sources

Home.AbsolutePosnValid

This source is activated when a Home is successfully completed. It indicates that the drive has been homed properly. It is will be deactivated by the Home.#.Initiate destination, an encoder fault, a reboot, or when the drive's 24 V logic is powered down.

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 drive 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 drive 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.

3.36 Index View

An index is a complete motion sequence that moves the motor a specific incremental distance or to an absolute position. The index is initiated with the Index.#.Initiate destination or from a program.

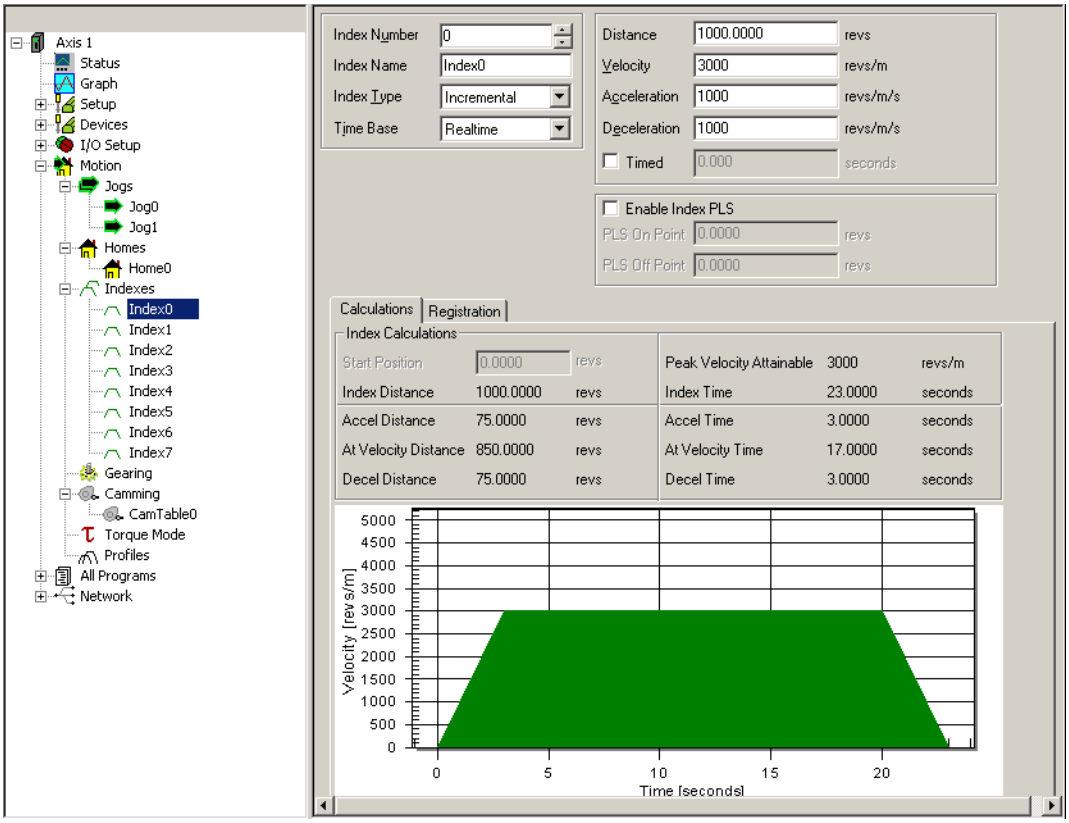


Figure 107: Index View

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 Track Cont., Posn Track 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. Real-time and Sync are the allowed selections.

Distance/Position

The Distance/Position parameter is a signed value which 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.

3.36.1 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 Epsilon EP-P drive will automatically calculate the necessary velocity, accel, and decel in order to achieve the programmed distance in the specified time. A Timed Index can not be compounded into or out of.

All index types can be specified as a Timed Index, except for Registration and Posn Track type indexes. This is because these types of indexes do not have a specified distance or absolute position. If a Registration or Posn Track type index is selected, then the Timed check box will become disabled.

Based on the Distance entered (or Position for Absolute indexes) and the Time 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 below shows a screen capture in which the Time check box has been enabled. Notice how the parameters which normally say Velocity, Acceleration, and Deceleration have changed to say Max. Velocity, Max. Acceleration, and Max. Deceleration. When the Time check box is enabled, these parameters automatically become maximums for use in the calculations.

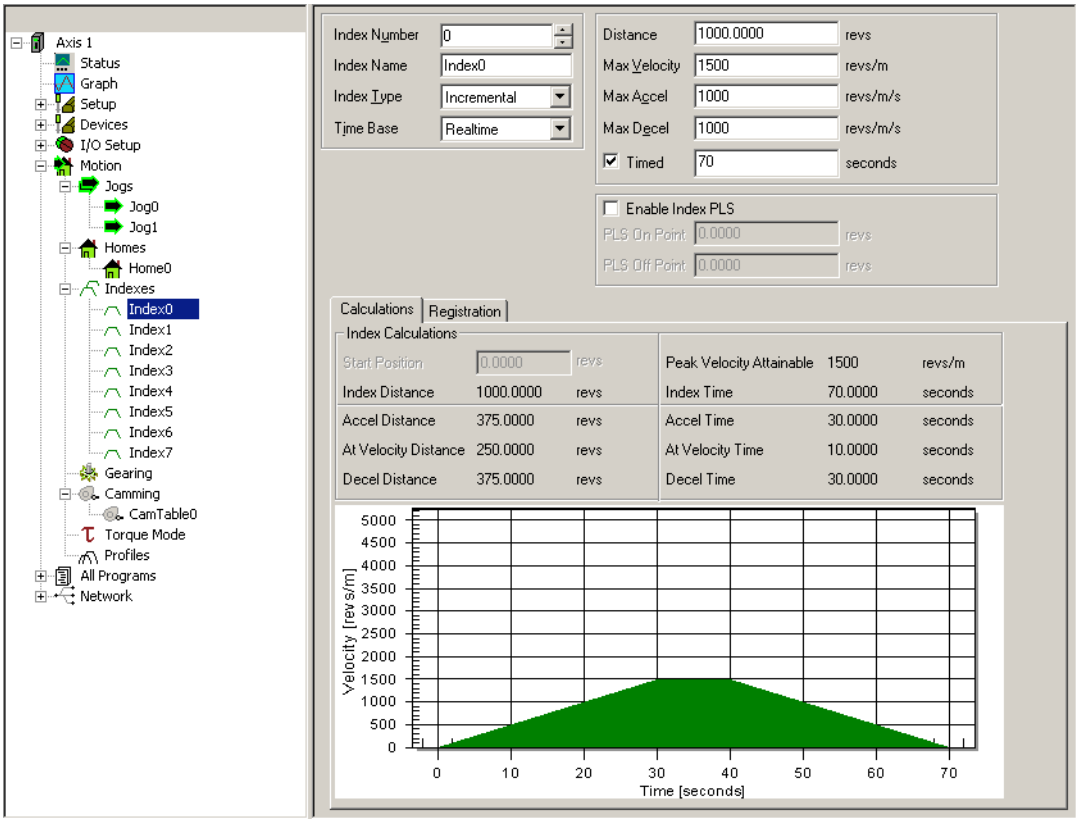


Figure 108: Time Check Box Enabled

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. 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 Time parameter depend on the current setting of the Time Base parameter. If Time Base is set to “Realtime” (default), then the units for the Time 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 Time parameter are defined by the Master Distance Units found on the Master Units view.

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. If the Maximum Velocity is low enough such that the profile will become trapezoidal (some duration at max velocity). 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.

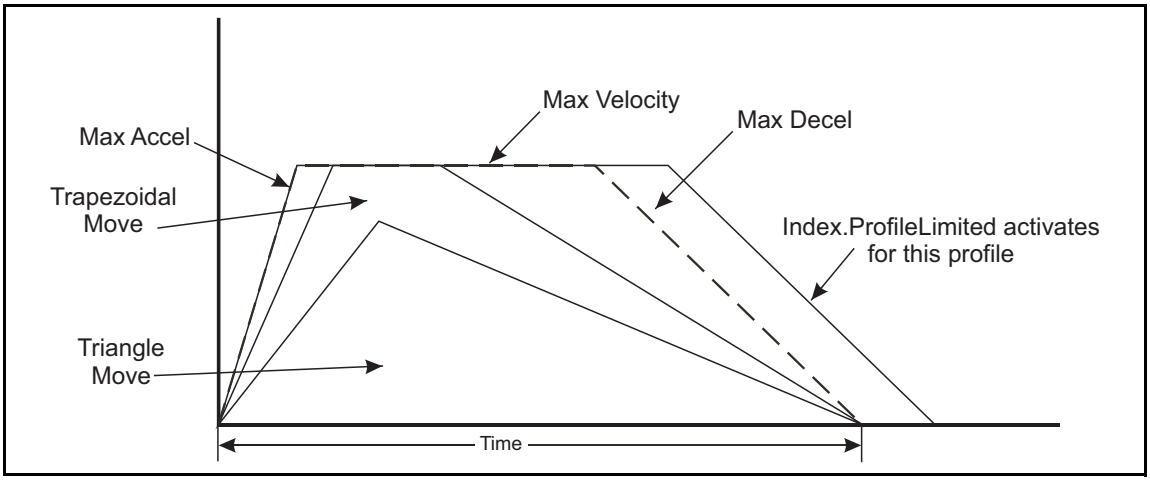


Figure 109: Timed Index Profiles

Enable Index PLS

This check box enables (when selected) or disables 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.

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 PLS’s 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 entered on the Registration tab and are only available 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 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, Torque Command, or Torque 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 In is selected from the list box, 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 view when Analog In 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 to continue travelling 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

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: 1

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 text box, 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 it's 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).

3.36.2 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 source 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.

3.36.3 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.

Toolbar button Method

The Add Index button (shown below) will add a new index to the user configuration. Indexes are added in sequential order. Clicking on the button will add an index and bring you to the Index setup view allowing you to enter the index parameters.

The Delete Index button (shown below) 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 you wish to delete, and then click on the Delete Index button.

PowerTools Pro Menu Bar Method

Adding an Index

From the PowerTools Pro menu bar, select Edit/New/Index. An index will be added in sequential order and you will be brought to the Index setup view allowing you to enter the index parameters.

Deleting an Index

Navigate to the Indexes View, and select the Index you wish to delete. From the PowerTools Pro menu bar, select Edit/Delete/Index. The selected Index will be deleted from the configuration.

Right Click Method

Adding an Index

Navigate to the Indexes View. Position the mouse pointer in the right side of the view and right-click the mouse. A selection box will appear allowing the user to add a New Index or Delete an Index. Click on New Index and an index will be added in sequential order and that Index's setup view will open allowing the user to enter the index parameters.

Deleting an Index

Navigate to the Indexes View. Select the Index you wish to delete, and then right-click the mouse. A selection box 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.

3.37 Gearing View

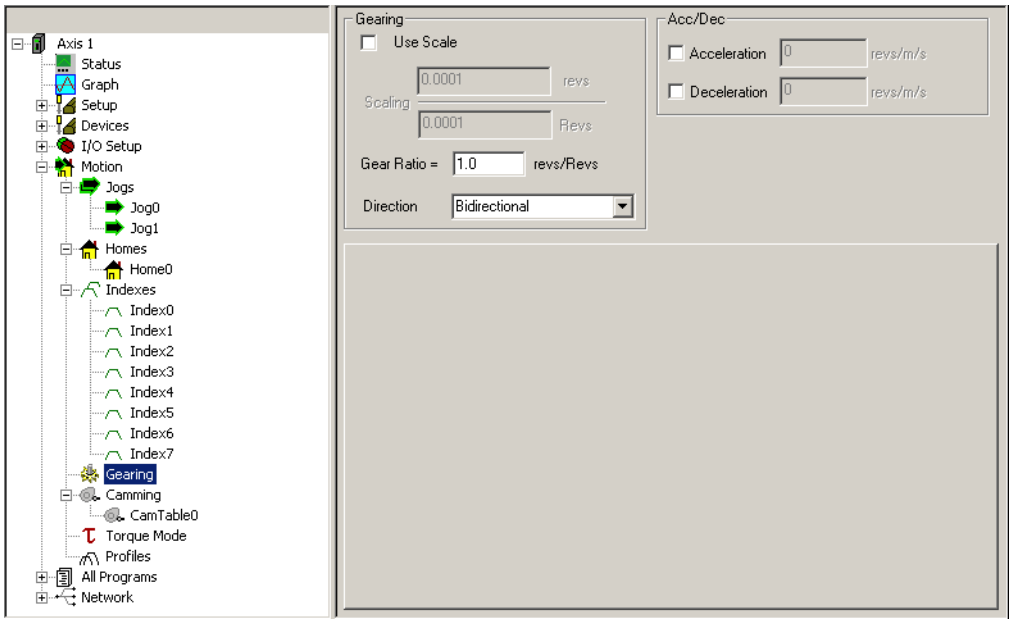


Figure 110: Gearing View

3.37.1 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:

$$\text{Scaling} = \frac{\text{Gear.ScaleNumerator (User Units)}}{\text{Gear.ScaleDenominator (Master Units)}}$$

3.37.2 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 master distance unit of travel. The master distance units are configured on the Master Units view. 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 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 (800µsec ≤ update rate ≤1600 µsec). Analogously, when gearing is deactivated, the motor will use peak torque to bring the motor to a stop without a deceleration ramp.

3.37.3 Direction

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.

ComMinus

The follower will follow only the minus master axis command.

ComPlus

The follower will follow only the plus master axis command.

3.37.4 Acc/Dec Group

Acceleration and Deceleration ramps may be enabled on the Gearing view as seen in Figure 110 above. 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 111 below 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).

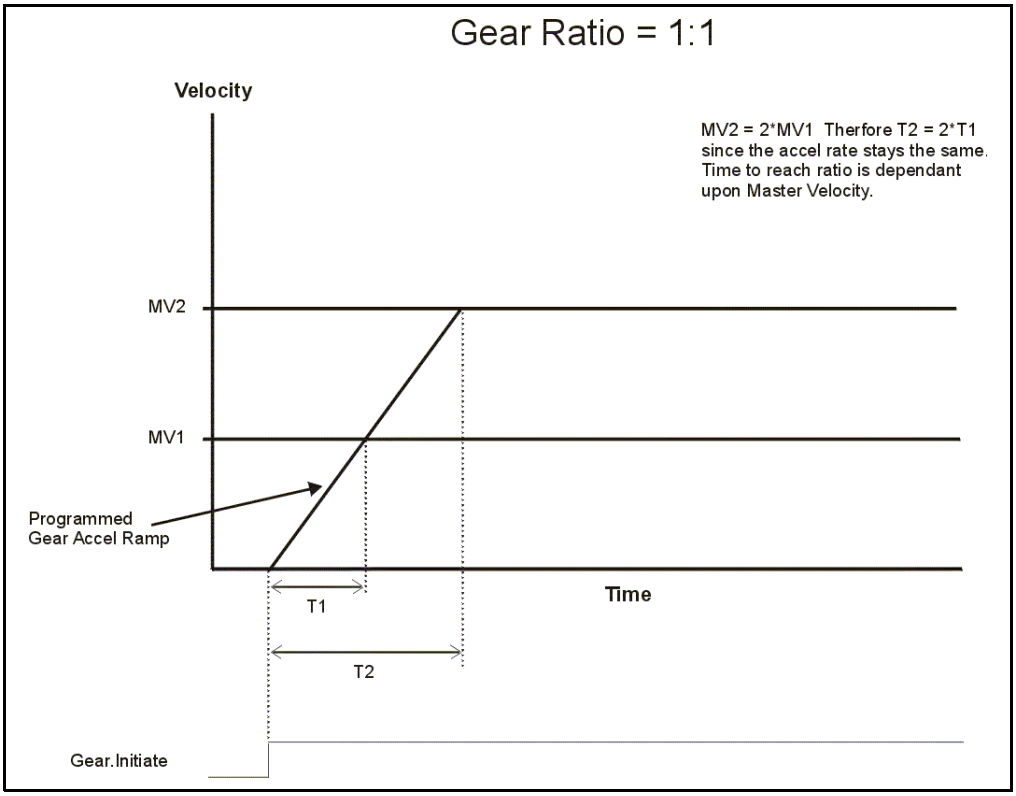


Figure 111: Gearing Acceleration Ramp Description

The Gear.Ratio/Scaling can be changed on the fly (while 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 (800<t<16000 microseconds).

Initiating Gearing Motion

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 program, the Gear.Stop instruction is used to stop the gearing motion.

Stopping Gearing Motion

The method used to stop gearing motion depends on how the gearing was initiated. If gearing was initiated using an Assignment, then simply deactivating the Gear.Active destination will cause gearing motion to stop. If gearing was initiated from within a program, then the Gear.Stop command must be used to stop gearing. If gearing motion is operating on Profile.1, then the On Profile.1 motion modifier must be used after the Gear.Stop instruction. See *Multiple Profiles* on page 103 for more information.

Distance Recovery when Gearing

If accel ramps are used, the gear will lag the master when Gear.AtVel is activated. The total lag distance will be stored in parameter Gear.RecoveryDist. This parameter value can be loaded in to Index.#.Dist and that Index is ran on the other profile. An alternative method is to enable the Distance Recovery feature on the Distance Recovery view. With Distance Recovery enabled.

3.38 Camming View

Electronic cams provide a non-linear motion function for a single axis. The basic motion can best be illustrated in Figure 112 of a mechanical cam and cam follower (or slave).

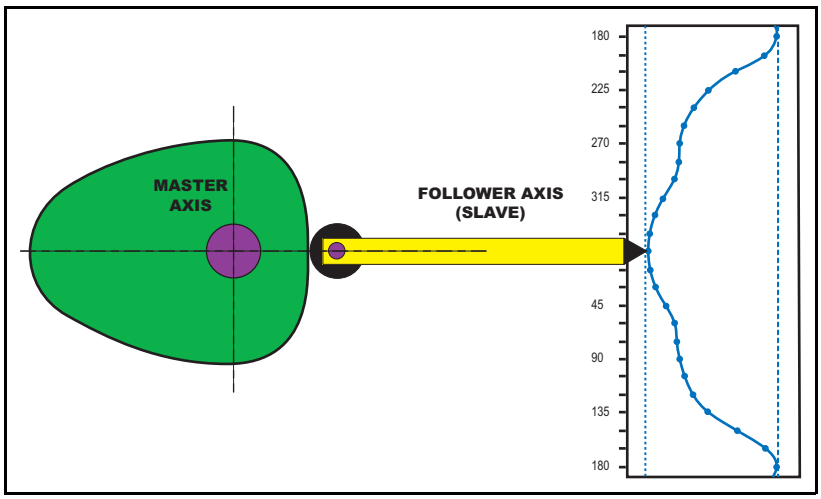


Figure 112: Mechanical Master and Slave Follower

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 principal 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 manor as jogs, home and indexes.

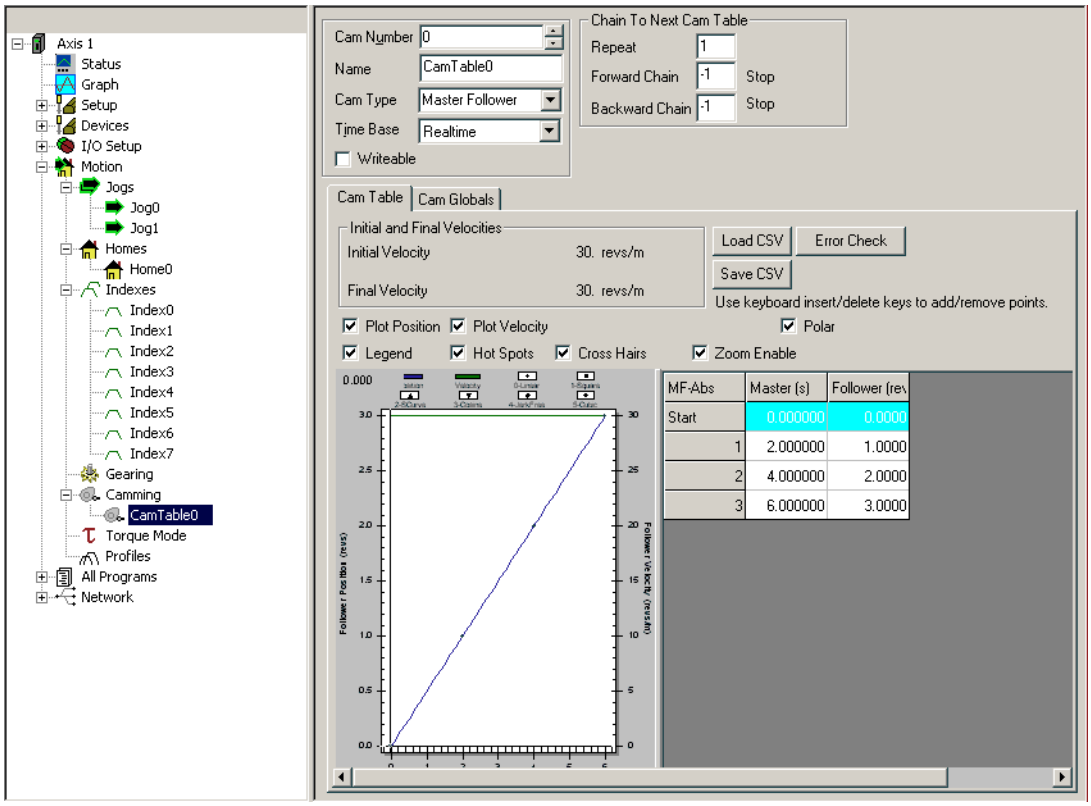


Figure 113: Camming View

Cam Number

Use the arrows of the spin 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.

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.

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.

3.38.1 Cam Table Plot Error

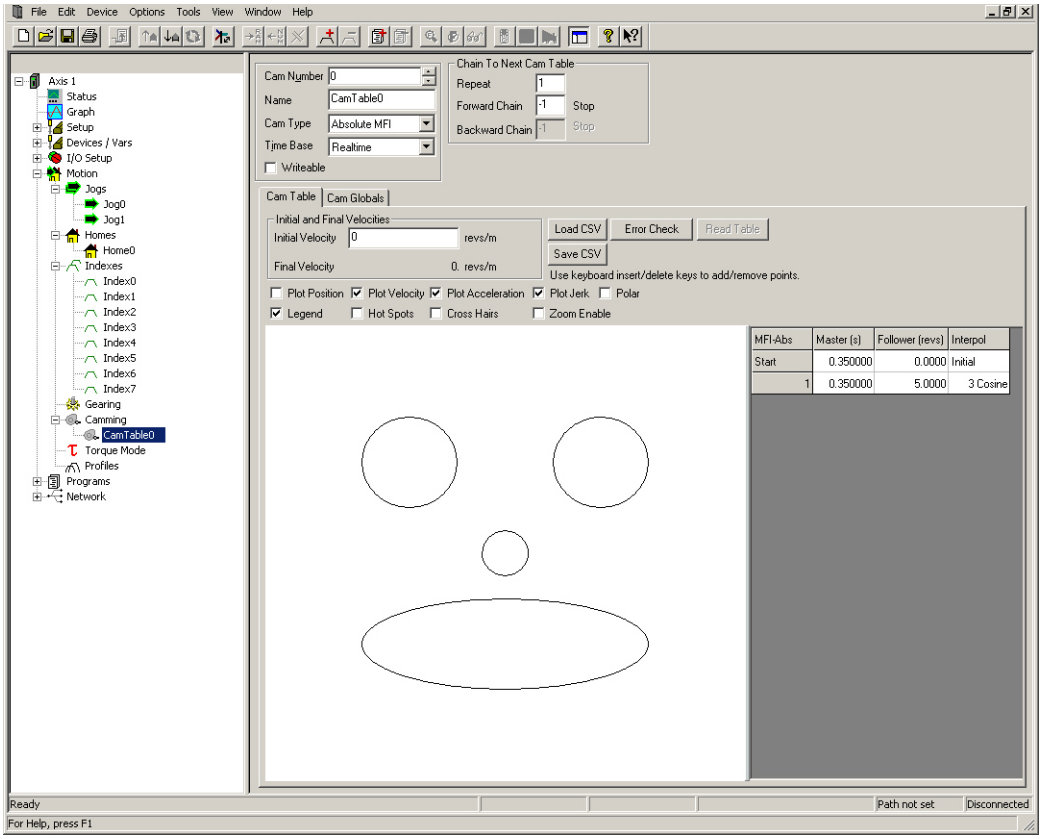


Figure 114: Cam Table View With a Smilely Face

When the Master and Follower parameter values are entered into the cam table the graph is updated to reflect these values. If

PowerTools is unable to plot the graph due to these values a smilely face will appear in the graph area of the view. To correct this error just change parameter values until the smilely face is cleared and the plot appears.

3.39 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.

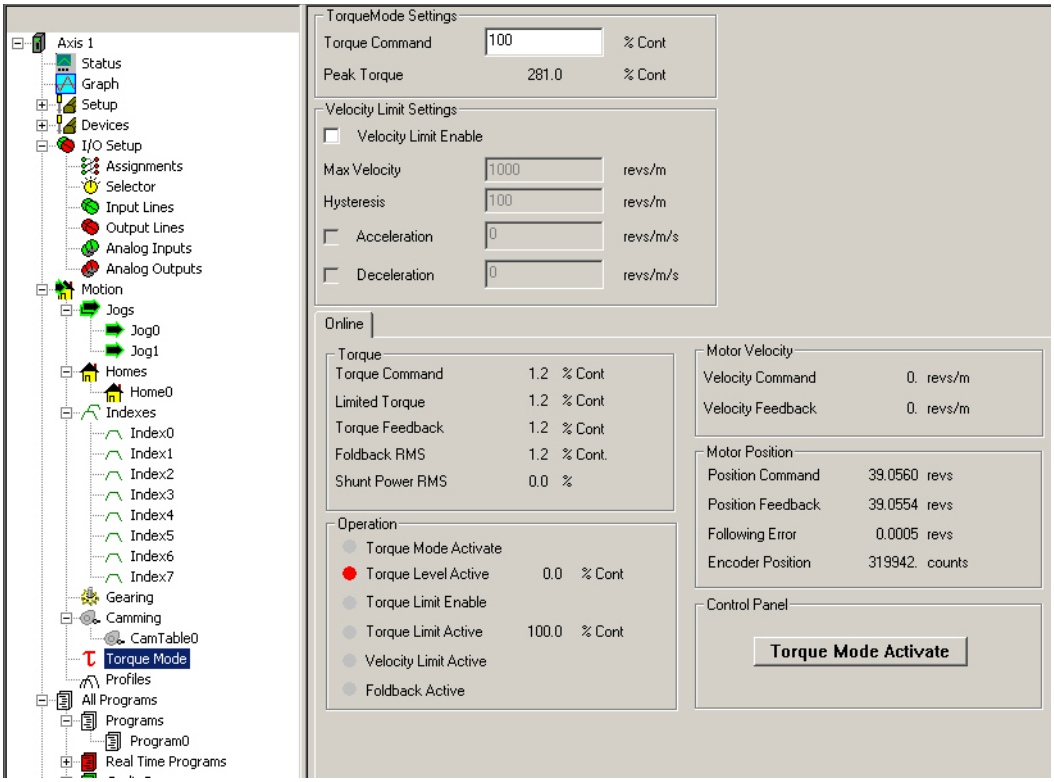


Figure 115: Torque Mode - Online

3.39.1 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.

3.39.2 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.

Hysteresis

When velocity is limited, this parameter (TorqueMode.TorqueModeHysteresis) is the value that the velocity must drop to before the drive returns to toque mode.

Acceleration Check Box

Select this check box (TorrqueMode.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 (TorrqueMode.DecelEnable) when an deceleration ramp is desired.

Deceleration

This parameter (TorqueMode.Decel) is the deceleration ramp for the velocity limit feature, when enabled.

3.40 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.

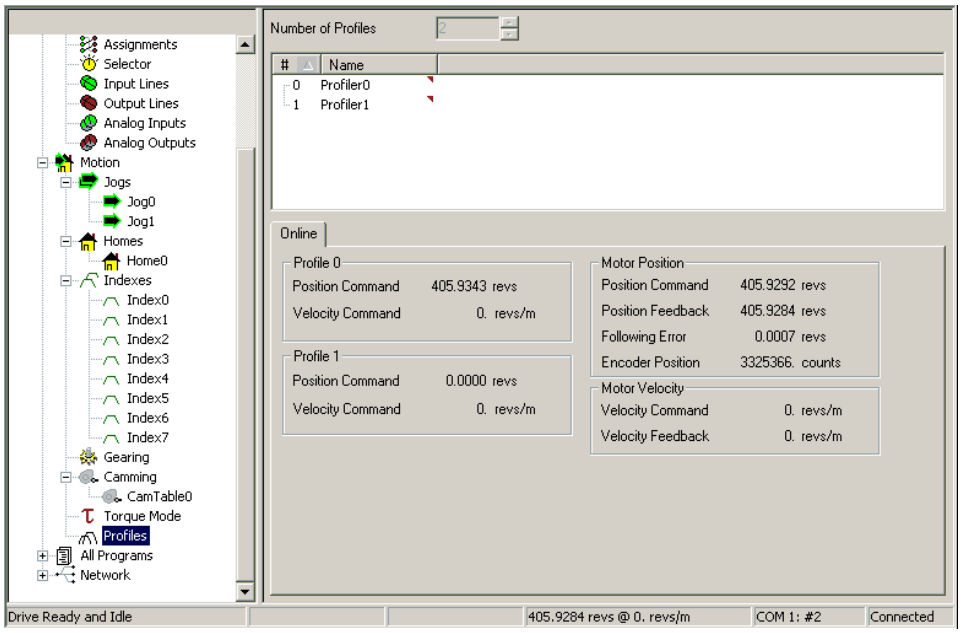
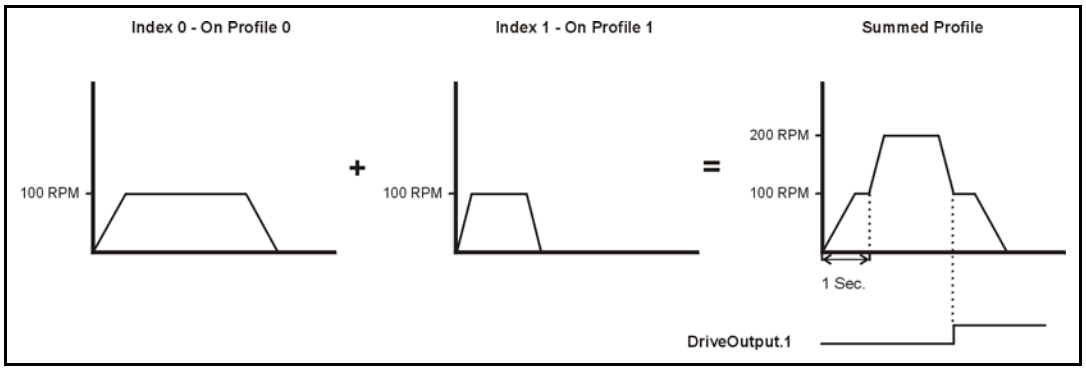


Figure 116: Profile View - Online

The following example program code runs Index 0 on Profile 0, Wait for 1 second, and then initiate Index 1 on Profile 1. The diagram shows how the two profiles look individually, and then shows how the motion looks after being summed by the two profiles.

```
Index.0.Initiate
Wait For Time 1.00 'Seconds
Index.1.Initiate On Profile.1
Wait For Index.1.CommandComplete
DriveOutput.1 = ON
```



If using the "On Profile" command on the same line of code as the "Using Capture" command, "Using Capture" should precede the "On Profile" command. Below are example lines of program code that initiate indexes on different profiles using captured data for the starting point.

Initiate Index 1 on Profile 0 using data stored in Capture object 2 as the starting point.

```
Index.1.Initiate Using Capture.2
```

Initiate Index 3 on Profile 1 using data stored in Capture object 1 as the starting point.

```
Index.3.Initiate Using Capture.1 On Profile.1
```

3.41 Stopping Motion

3.41.1 MotionStop from a Program

The MotionStop command will cause all motion to stop regardless of what type of motion it is, or where it was initiated from. Upon activation of the MotionStop, all motion will begin to decelerate to a stop using the standard Stop deceleration ramp. That ramp is defined using the StopDecel parameter. MotionStop is a level sensitive command meaning that as long as it is active, all motion will be stopped and prevented from running. When MotionStop is deactivated, all motion is permitted again. Any motion that is interrupted with the MotionStop command is cancelled, and will not complete when MotionStop deactivates.

The MotionStop command DOES NOT stop any programs. All programs that are active when the MotionStop is activated will continue to run as normal.

All motion stopped using the MotionStop command will stop using a realtime deceleration ramp (even if the timebase of the motion being stopped is synchronized). This can help in applications that use synchronized motion if the master stops and then the user wishes to break out

of the synch motion without performing an synchronized deceleration ramp.

Neither the CommandComplete signals from motion objects nor the ProgramComplete signals will activate if they have been stopped using the MotionStop command.

In the example below, Program 0 runs an infinite loop in which Index 5 runs and then waits for half a second and then repeats itself. When Input 2 activates, Index 5 will stop if in progress and the program will loop back to the Index.5.Initiate.

Example: 1

Program 0 – Running on Task 0

```
Do While TRUE
    Index.5.Initiate
    Wait For (Index.AnyCommandComplete OR MotionStop = ON)
    Wait For Time 0.50 ‘Seconds
Loop
```

Program 1 – Running on Task 1

```
Wait For DriveInput.2 = ON
MotionStop = ON
Wait For DriveInput.2 = OFF
MotionStop = OFF
```

3.41.2 MotionStop from an Assignment

The MotionStop as explained above can also be initiated from an Assignment. MotionStop can be found in the Ramps group of Destinations.

3.41.3 Profile.#.MotionStop from a Program

The Profile.MotionStop instruction is used to stop motion on an individual profile without stopping all motion. Upon activation of the Profile.MotionStop, any motion running on the specified profile will begin to decelerate using the StopDecel ramp down to zero velocity. The deceleration is performed in realtime regardless of the timebase of the active motion. This can be used in applications where motion is being run on both profiles simultaneously, and the user only wishes to stop one of the motion types. For example, an application that uses gearing to follow a master encoder and then uses indexes on the other profile to do correction profiles. The application may call for stopping all correction moves, but continuing the gearing motion. In this application, the user would perform a Profile.MotionStop on the profile doing the correction moves. The below example uses a separate program to control the Profile.Stop.

The Profile.MotionStop instruction does not stop the program that the motion is initiated from.

The Profile.MotionStop is level sensitive so that when it is activated, all motion on that profile will stop, and remain stopped, until the Profile.MotionStop is deactivated. If the Profile.MotionStop is activated, it will stop any motion in progress, and will also prevent any new motion from starting on that profile. No motion will be permitted on that profile until the profile being stopped has come to a complete stop. The motion that was stopped while in progress will not resume when the Profile.MotionStop is deactivated. Level sensitive motion that is initiated from the Assignments view (i.e. jogging, gearing) will not operate until the activate signal is reset and activated again.

The CommandComplete signal for the motion will NOT activate if the motion was interrupted using the Profile.MotionStop command regardless of motion type. In the example below, Program 0 would be stuck on the Wait For Index.AnyCommandComplete instruction if the Profile.MotionStop is used. To avoid this condition, “OR Profile.1.Stop” has been inserted after the Wait For Index.AnyCommandComplete.

Example: 1

Program 0 – Running on Task 0

```
Gear.Initiate On Profile.0
Do While TRUE
Wait For DriveInput.1 = ON
Index.1.Initiate On Profile.1
Wait For (Index.AnyCommandComplete OR Profile.1.MotionStop)
Loop
```

Program 1 – Running on Task 1

```
Do While TRUE
Wait For DriveInput.2 = ON
Profile.1.MotionStop = ON
Wait For DriveInput.2 = OFF
Profile.1.MotionStop = OFF
Loop
```

3.41.4 Profile.#.MotionStop from an Assignment

The Profile.#.MotionStop as explained above can also be initiated from an Assignment. Profile.#.MotionStop can be found in each Profiler group of destinations.

3.42 Network Group

For information on the DeviceNet, Ethernet and Profibus Views, please refer to the *Epsilon EP Drive Connectivity Reference Manual* (P/N 400518-04).

3.43 Modbus RTU/TCP View

The Modbus RTU/TCP View is used to assign Modbus addresses to individual parameters.

By selecting Modbus RTU/TCP in the Hierarchy Tree, the Modbus RTU/TCP View will appear on the right (see Figure 117). The right part of the window displays all of the drive parameters. The number of parameters that appear depends on the User Level.

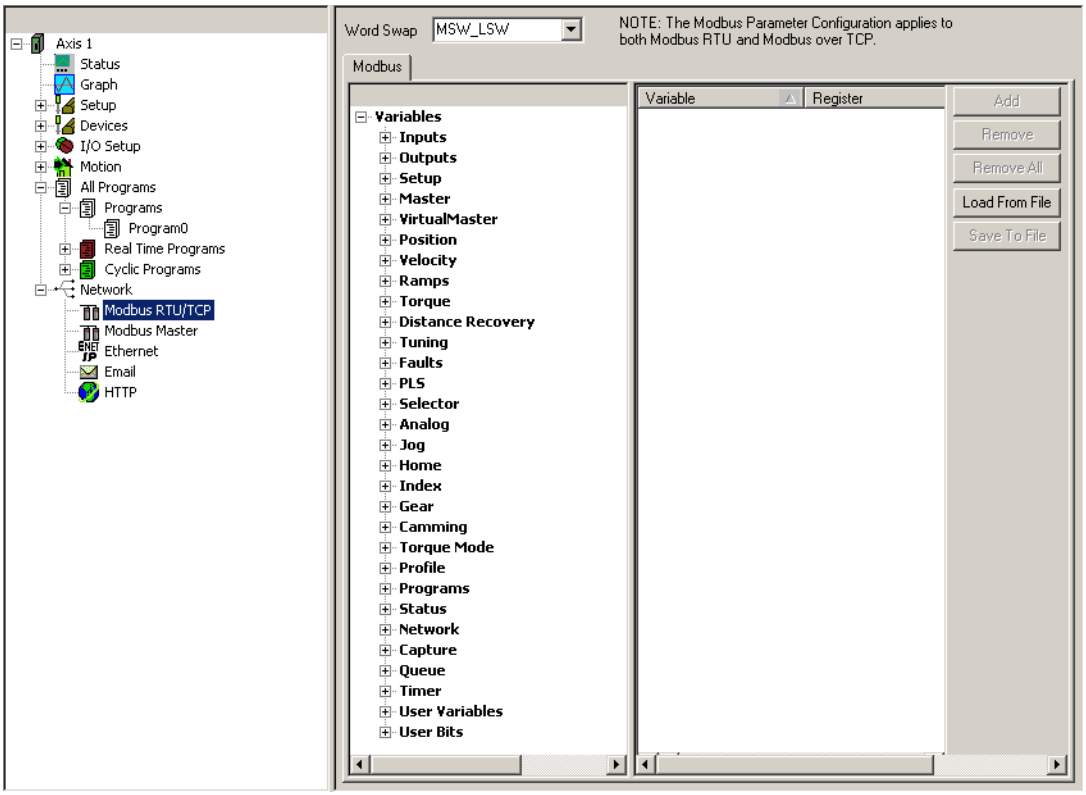


Figure 117: Modbus View

An external device such as a Human Machine Interface (HMI) or PLC can be used to monitor or edit individual drive parameters. The Epsilon EP-P drive uses a 32-bit Modbus RTU communications protocol.

In order to view or modify a parameter, a Modbus address must be assigned to the specific parameter. To do this, locate the parameter you wish to read/write to or from in the variables list in the middle of the view. Once you have found the proper parameter, click and hold the left mouse button over the parameter. While still holding the button on your mouse, drag the parameter into the Modbus window area on the right of the view. Now let go of the mouse button.

The New Assignment dialog box will appear. This will automatically assign the next available modbus address, or allow you to enter a different Modbus address. Click ***OK***, then you will be able to read or write the parameter at that address.

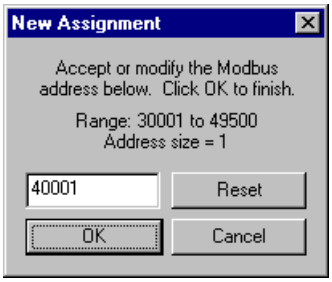


Figure 118: Modbus New Address Dialog Box

Address ranges are as follows:

Address Range	Accessibility	Type	Data Size
4xxxx	Read/Write	Register	32 bit word
3xxxx	Read Only	Register	32 bit word
1xxxx	Read Only	Input Bits	bit
0xxxx	Read/Write	Coil	bit

Any individual Modbus address can be deleted by selecting the parameter you wish to delete, and click on the ***Remove*** button. The address selected will be removed from the list. If you wish to delete all of the Modbus addresses that have been created, then simply click on the ***Remove All*** button. All of the addresses will disappear and the Modbus window will be empty.

Some Modbus addresses have been reserved and can not be assigned: 39980-39999 and 49501-49999.

NOTE

Some configuration software uses a 6 digit addressing base. The first digit is an indication of register or bit type. Thus 400001 in this software is equal to 40001 in Control Techniques Modbus RTU.

3.44 Modbus Master View

Modbus Master provides a method for the Epsilon EP drive to read or write directly to standard compliant Modbus slaves (such as extended I/O blocks). The process of using this function is to first define the communication path to the slave(s), then use the user program instructions to directly read or write either bits or words to the identified slave(s).

Configuring Modbus Master

The Modbus Master view, RTU Setup tab allows the user to configure the drive's serial (RS-485) port to operate as a Modbus Gateway and/or as a Modbus Master. Modbus Gateway check box or Modbus Master check box must be selected otherwise the port is configured as a Modbus slave. The RTU configuration comprises of two sets of parameters:

- The serial port configuration (baud, parity and stop bits)
- The timeouts and gaps for data transmission, see Figure 121 *Modbus RTU Timing Diagram* on page 108.

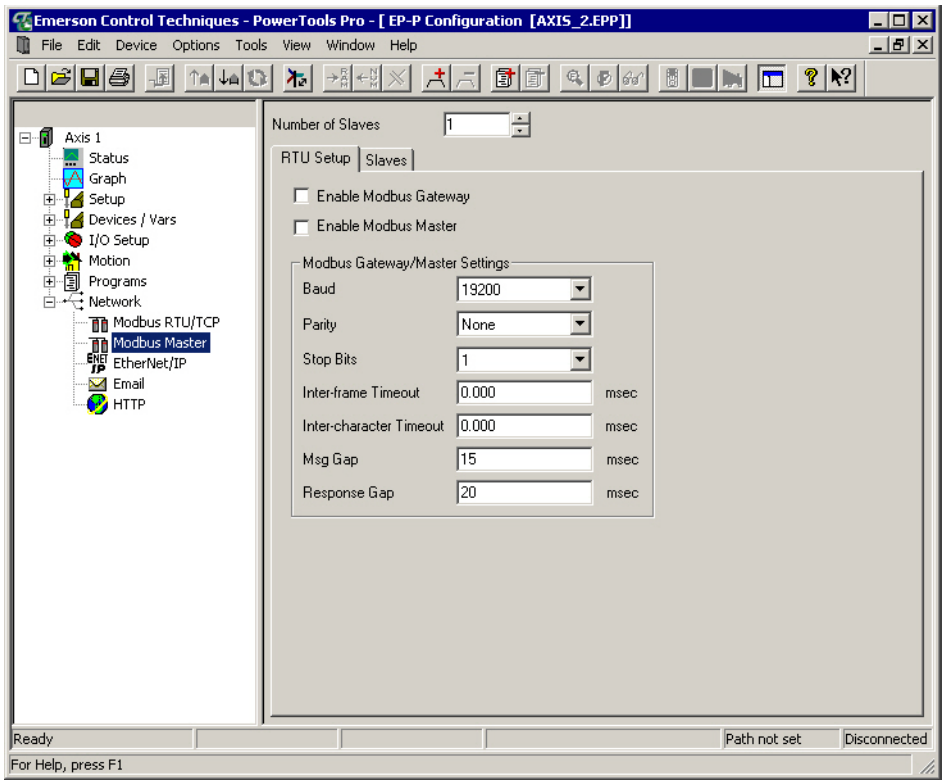


Figure 119: Modbus Master View

Identifying Slaves

Slaves need to be identified before they can be used. The Slaves tab on the Modbus Master view specifies the attributes of the slave(s). The quantity of slaves can be increased or decreased by changing the "number of slaves" field using the up and down arrows. This will add or remove slave entries.

Each slave entry identifies how the drive can reach the slave, its unit ID, data representation and Modbus specific timing attributes, see Figure 121 *Modbus RTU Timing Diagram* on page 108 for more details.

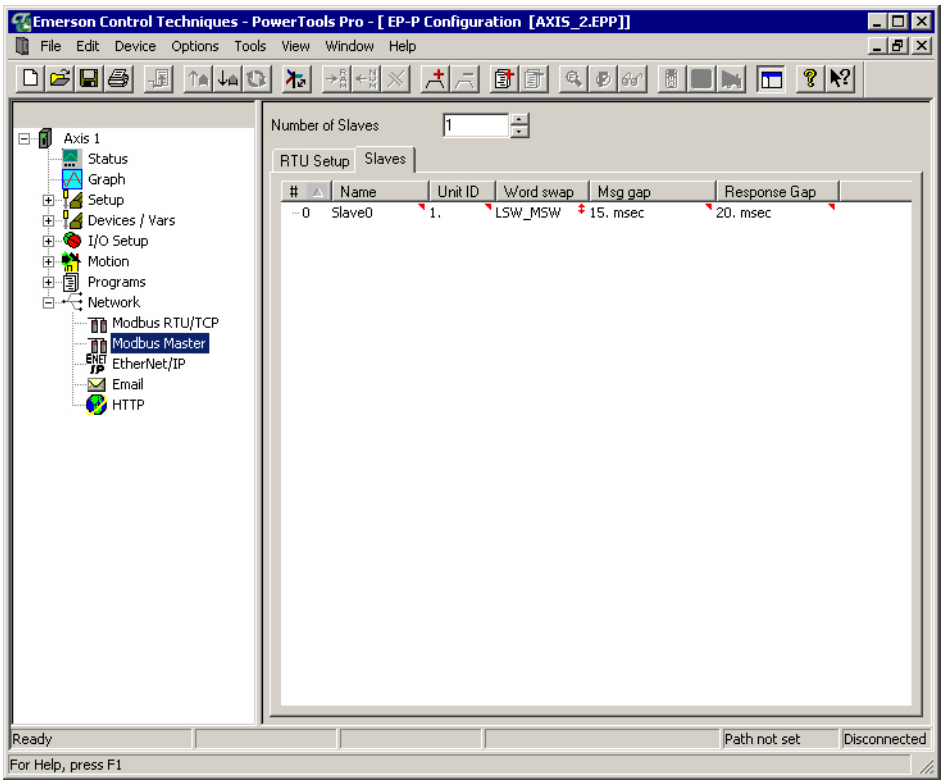


Figure 120: Modbus Master View - Slave Tab

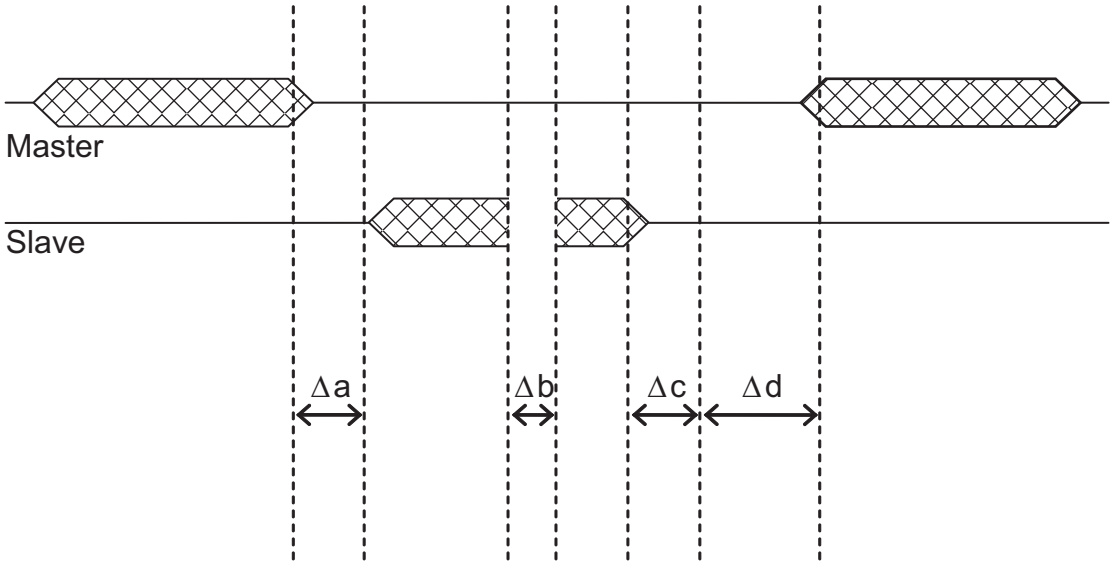


Figure 121: Modbus RTU Timing Diagram

Δa - Response Gap - Specifies the maximum timeout for the response packet. If a response packet is not detected within this period then the slave device status is set to no_response.

Δb - Inter Character - Specifies the maximum period for receiving each character of the Modbus packet. If a character is not received during this period, then the master considers the partial packet incomplete, and sets the slave device status to no_response.

Δc - Inter Frame - Specifies the period of no data transmission indicating the end of the Modbus packet. Modbus specifications generally specify this gap to be 3½ characters wide. If data is received during this period, then it is considered a part of the current data packet.

Δd - Message Gap - Specifies idle period between the last packet received and the next packet transmitted. This gap allows for extended quite time for multiple slaves on the same communications path; where slaves may be actively monitoring Modbus packets and may miss the standard Inter Frame gap.

3.45 DeviceNet View

For those drives that have the DeviceNet option, please refer to the *Epsilon EP Drive Connectivity Reference Manual*, P/N 400518-04, which can be found on the Control Techniques MME Power CD.

3.46 Profibus View

For those drives that have the Profibus option, please refer to the *Epsilon EP Drive Connectivity Reference Manual*, P/N 400518-04, which can be found on the Control Techniques MME Power CD.

3.47 Ethernet View

This view is used to navigate through the Ethernet related setup for the Epsilon EP-P drives. The Epsilon drive needs a minimum of an IP address and Subnet Mask configured before attacking it to an Ethernet hardware setup. For more information please refer to the *Epsilon EP Drive Connectivity Reference Manual*, P/N 400518-04, which can be found on the Control Techniques MME Power CD.

4 Programming

4.1 Programs

Programs are a series of indexes, homes and jogs that have been previously setup. You combine these with other programming steps to create a complex motion profile. Each motion program provides a series of movements in conjunction with other machine functions. The movements are used to perform a particular machine operation.

Multiple programs can be created using PowerTools Pro software and stored in the Epsilon EP-P drive. The drive is capable of storing up to 100 indexes, 99 motion programs, and a maximum of 1024 program steps in Flash Memory. The amount of available Flash Memory determines how many programs, program steps, indexes, etc. that the configuration can hold.

The number of available programs and average number of steps per program are directly related to each other. The memory is setup such that if you require 99 programs (maximum), each program can have an average of 10 program steps each. If the number of programs is reduced to a minimum, you could have as many as 1024 steps in a single program.

Expanding Programs in the Hierarchy Tree the user will notice there is 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 active 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.

4.2 Program Toolbar Buttons

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.

Undo Last Change

This button will undo the last change made to the program. PowerTools Pro 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 Pro 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.

Book Mark

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.

Next Book Mark

This button will position the cursor on the next available bookmark ahead of the cursor in the program.

Previous Book Mark

This button will position the cursor on the previous bookmark behind the cursor in the program.

Delete All Book Marks

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 Book Mark 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 Pro will attempt to give a detailed description of the error.

Red Dot Help can also be used to read the status of a program variable. Click the Red Dot Help button to activate Red Dot help, then click on the variable in the line of code in the program text. The line of code selected must not have a red dot because the error will take precedence.

After clicking on the variable a yellow popup window displays information about the variable.

- When online it will display the current values for the variable
- A short description of the variable is displayed
- It will display the initialization value set in the application by the view settings
- It will display the range for numerical data, selection options for selections, Boolean options for Booleans
- If the variable is read only

Drag In I/O

Clicking on this button will open the Select EPP Inputs/Outputs window. From this window, the user can drag Drive Input/Output lines of text into the program. This feature can be used to minimize the need to type in program statements. The Input or Output state (i.e. =On or =Off) can also be dragged into the program from this window.

Drag In Operands

This button will open the Select Math Operator window. From this window, the user can drag formula Operands (i.e. +, -, /, *) into the program formula.

Drag In Variables

This button will open the Select Variables From Tree 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 Epsilon EP-P drive. 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 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.

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 Real Time programs.

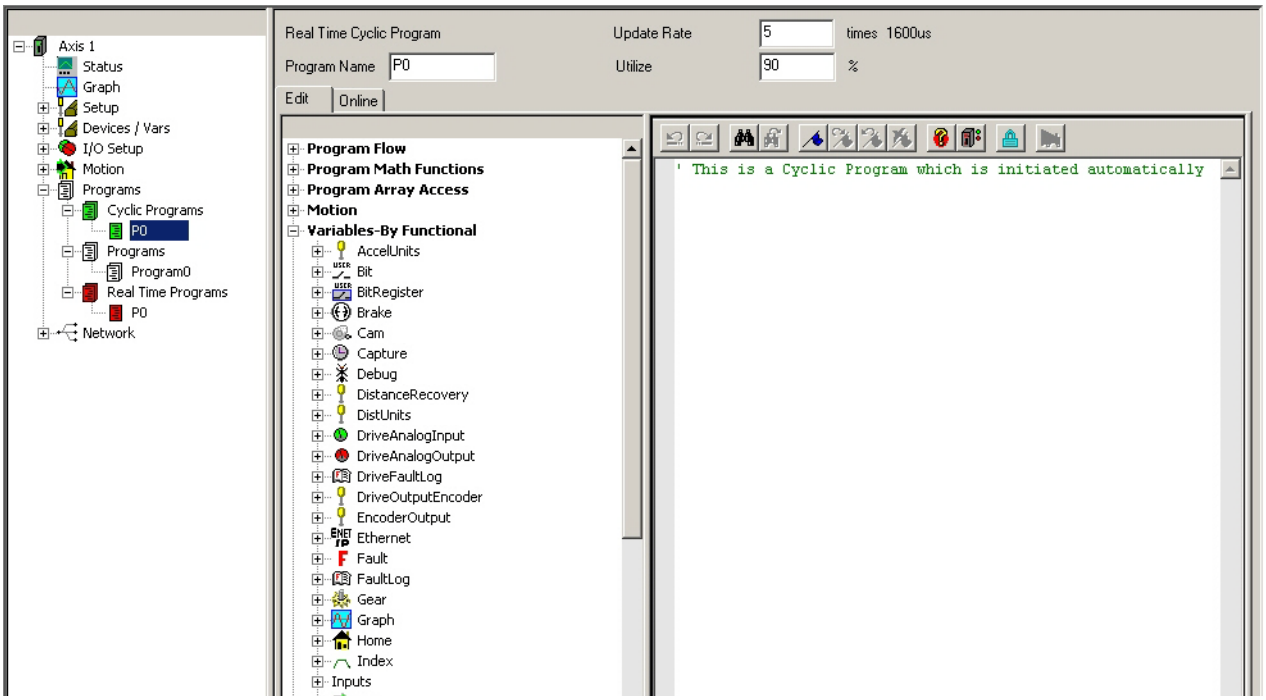


Figure 122: Cyclic Programs View

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 percent 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 processes 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 the utilize percentage and trajectory rate.

User Program View

Click on Programs\Program#, the program view will appear on the right (see Figure 123). The left side (pane) of this view contains the program instructions. The right side of the Programs view contains the Program Toolbar above the program code area.

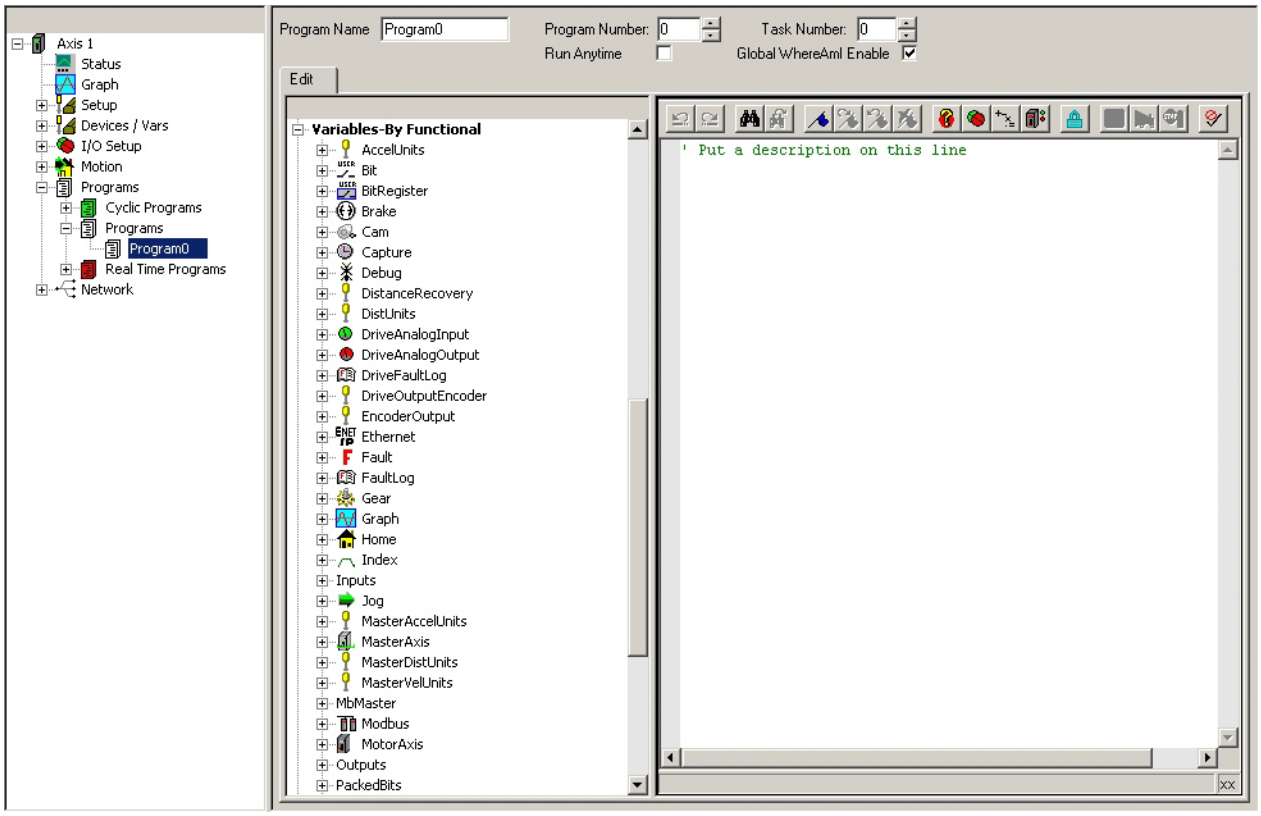


Figure 123: Programs View

Program Name

This parameter (Program.#.Name) is a 12 character string that the user can assign to an individual program. It allows the user to give a descriptive name to programs for ease of use.

Program Number

This spin box allows the user to create additional programs. Unlike the motion views, to get to Program1 you must use the arrows of the spin box.

Task Number

This parameter (Program.#.TaskNumber) allows programs to run simultaneously. To run one program (for example, Program1) while another program (Program0) is executing, Program0 will be set for Task Number 0 and select Program1, click on the up scroll arrow of the Program Number to Program 1 then click on the up arrow of the Task Number to number 1.

Use the up and down arrows next to the Task Number to change the Task Number. To create a new Task, simply click on the up arrow until PowerTools Pro asks if you wish to create a new Task. The maximum number is 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 cannot be run at the same time. If a given program calls another program, then the 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.

Run Anytime Enable Check Box (Programs only)

Some applications require the ability to run a program as soon as a fault occurs or continue running a program even through a fault condition. In order to do this, a program must be classified as "Run Anytime". To define a program to be able to run during a fault or while the drive is disabled, the "Run Anytime" check box must be enabled (selected) in the Program view or Program.#.RunAnytimeEnable can be activated in a program.

NOTE

Note: If a Program Fault occurs the program will stop executing regardless of the "Run Anytime" setting.

Global Where Am I Check Box (Programs only)

This parameter (Program.#.GlobalWhereAmIEnable) is used to control the functionality of the Global Where Am I arrow in a user program. If the user activates the Global Where Am I feature by clicking the ***Global Where Am I*** button on the PowerTools Pro 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 Pro 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. When clear, the Global Where Am I feature will not function within the specific program. Each program has its own Global Where Am I Enable check box. By default, all Global Where Am I Enable check boxes are selected (active).

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 insure all the operations are completed within the update rate.

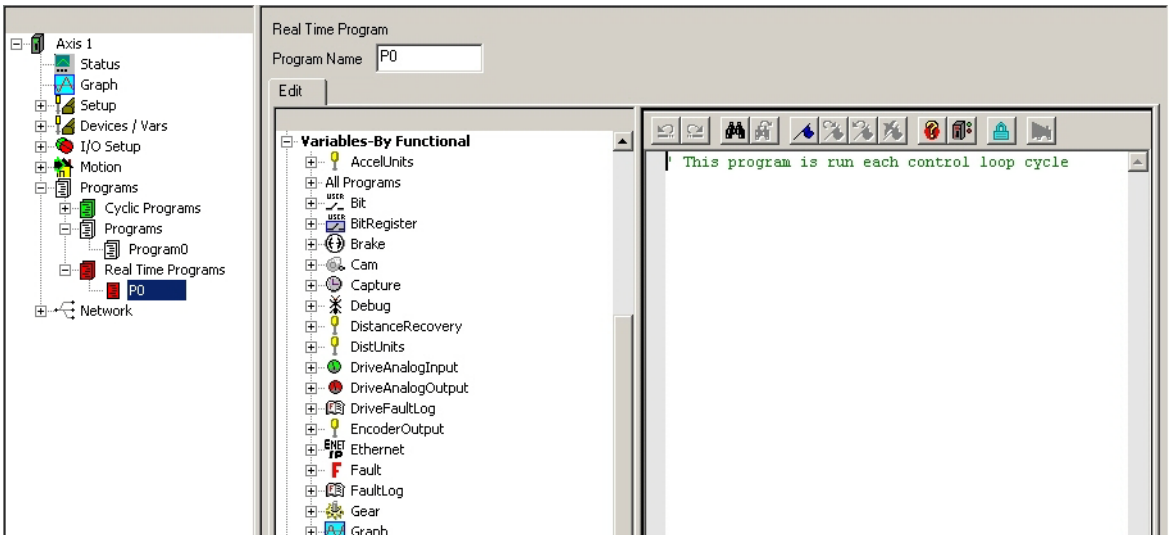


Figure 124: Real Time Programs View

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.

4.3 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 fault detection.

The Epsilon EP-P 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 processor stops all tasks and updates the control loop every 1600 microseconds (default, update time may be set by user). Inside the control loop update, the drive updates the motion trajectory, captured data, digital inputs and outputs, and other control parameters. Between each control loop update, the drive processes messages (i.e. Modbus, Ethernet, Faults, etc.) and then runs as much of the user programs as possible until the next control loop update begins. Each update, a different task is processed, and therefore how long it takes a given user program to complete depends how many tasks have been created.

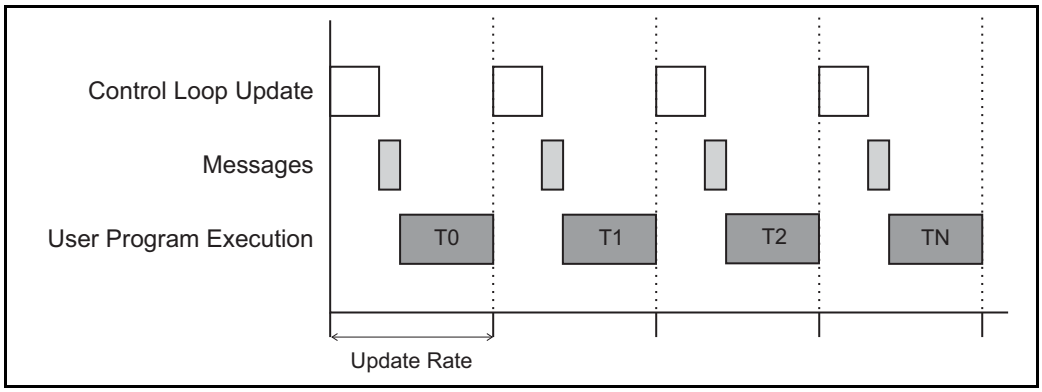


Figure 125: Time Slicing and Multiple Tasking Handling

The task assignment is done on the program view. Figure 123 shows the program view with the Task Number parameter. Use the up and down arrows next to the Task Number to change the value. To create a new Task, simply click on the up arrow until PowerTools Pro asks if you wish to create a new Task.

The Epsilon EP-P drive 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.

Timing Diagrams

In Figure 125 the update rate is set to 1600 microseconds (Found on the Setup View). The first routine to be processed in the update rate is the control loop update. Next, all messages will be handled. If no message has been sent from a Modbus master and no faults 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. After 1600 microseconds has passed, 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 is accurate as long as no program is blocked. The following figure shows examples of user programs and task numbers and how the drive processes them.

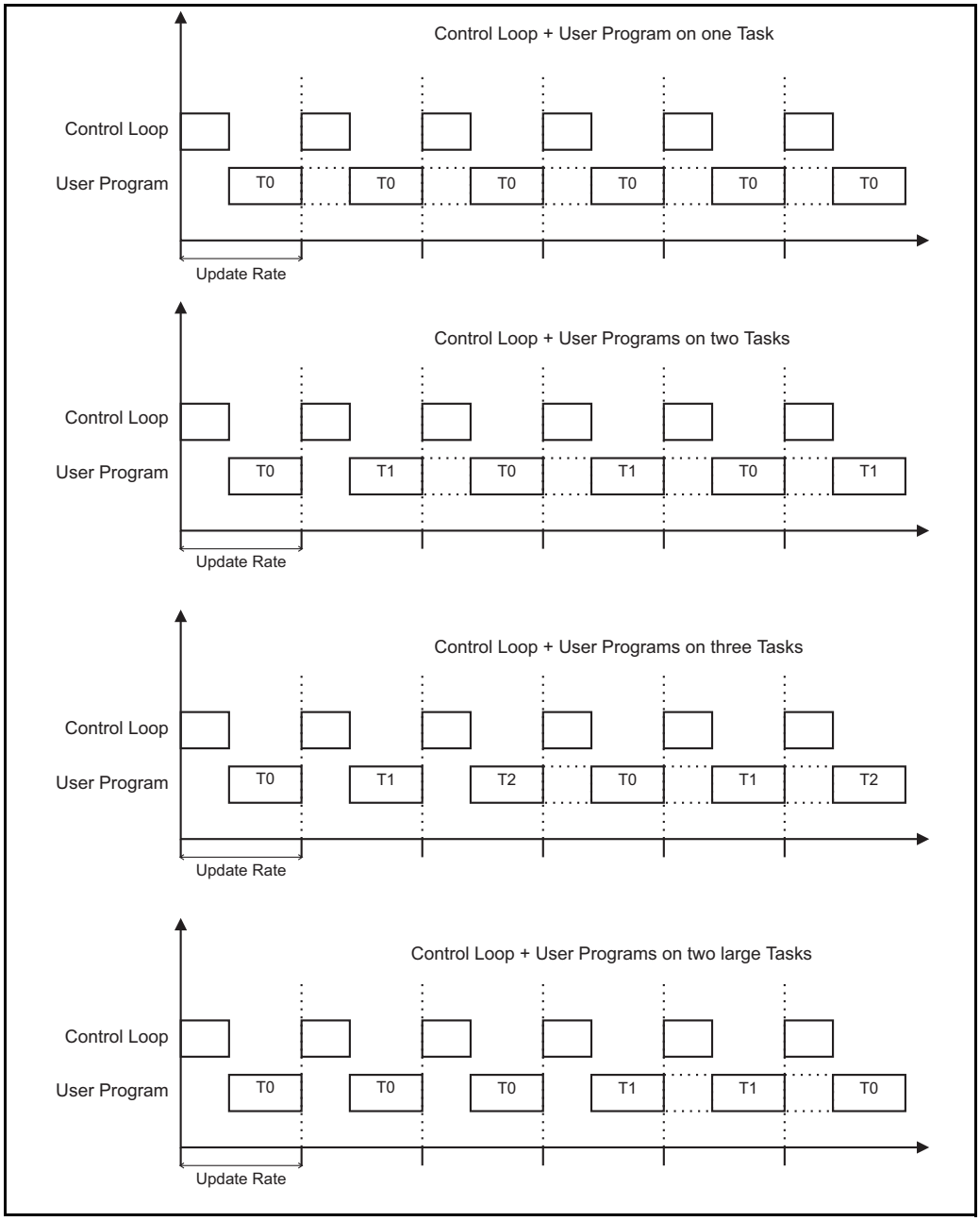


Figure 126: Diagram of User Programs with Multiple Tasks

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.

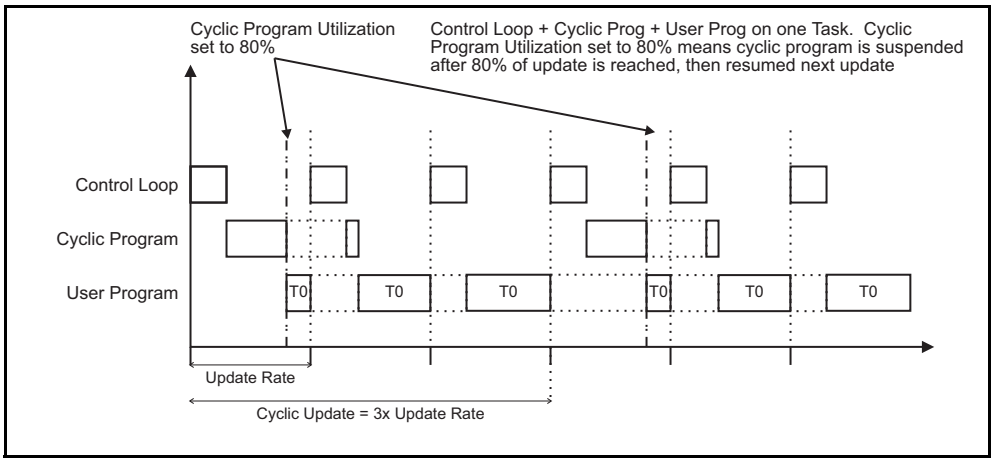


Figure 127: Diagram of User Program and a Cyclic Program

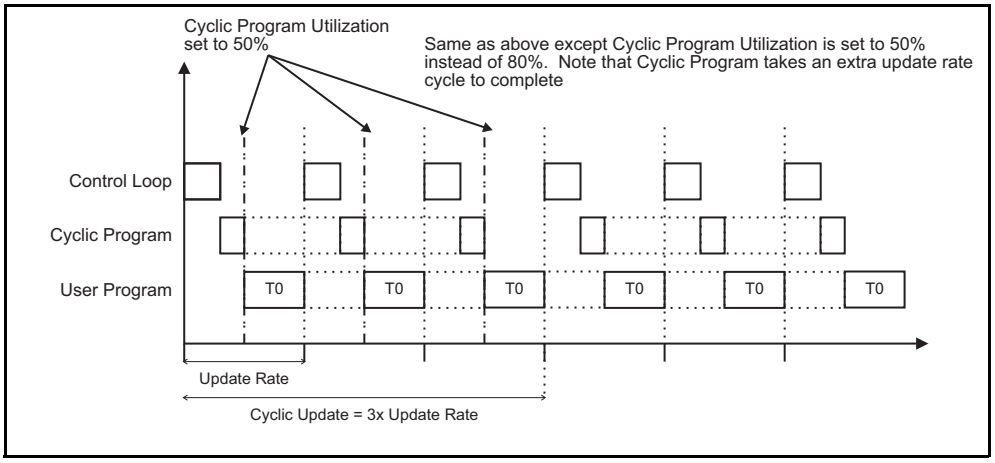


Figure 128: Diagram of User Program and a Cyclic Program

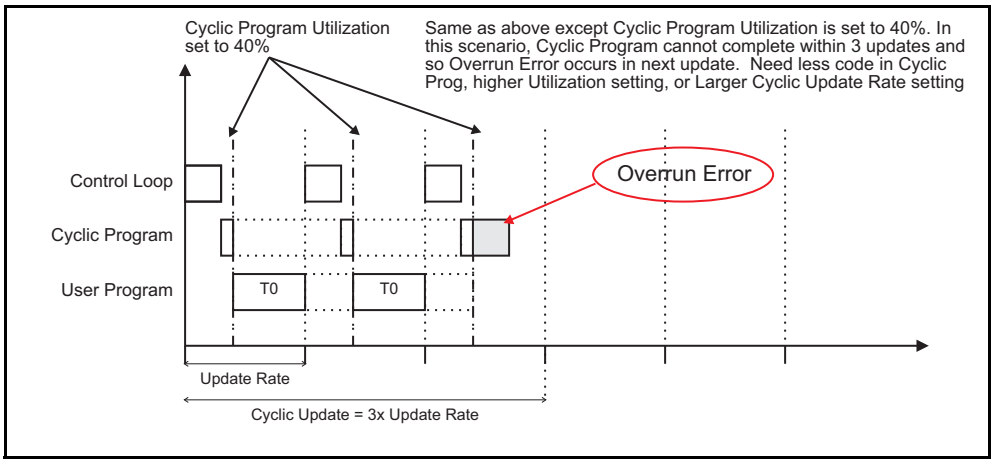


Figure 129: Diagram of User Program and a Cyclic Program

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.

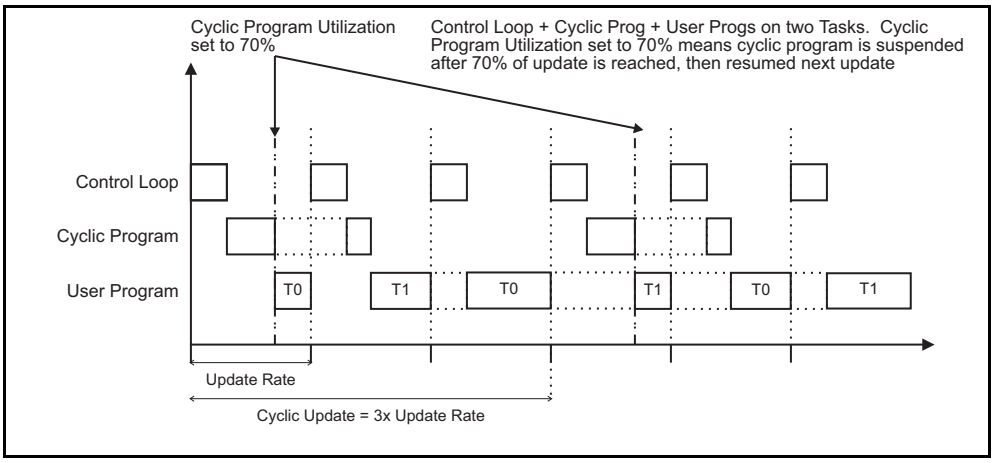


Figure 130: Diagram of User Programs on Two Tasks and a Cyclic Program

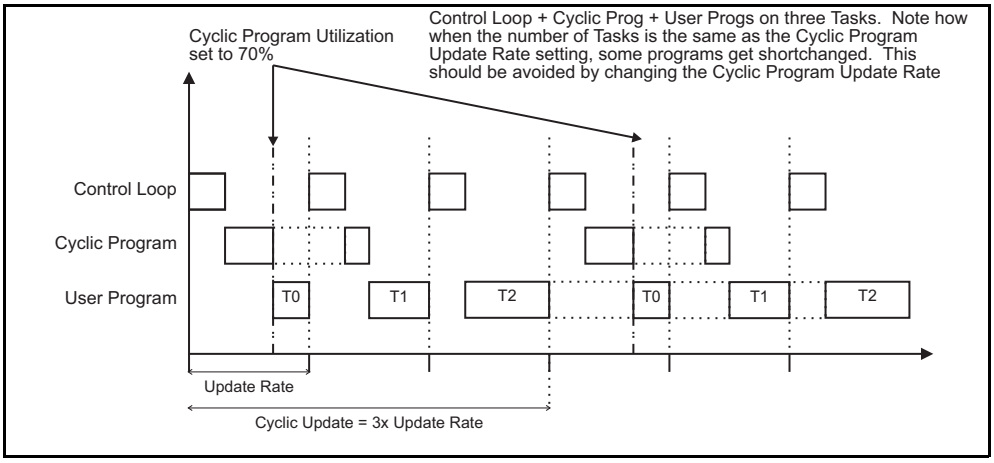


Figure 131: Diagram of User Programs on Three Tasks and a Cyclic Program

The 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.

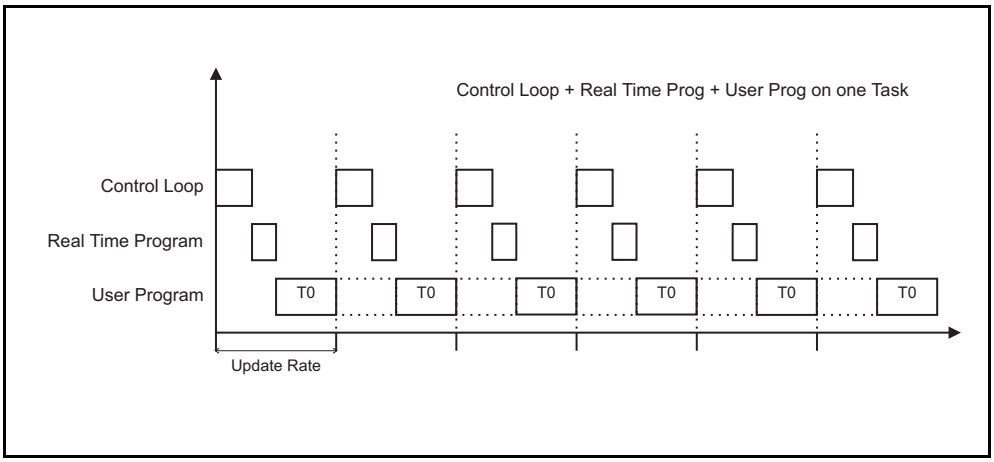


Figure 132: Diagram of User Program and a Real Time Program

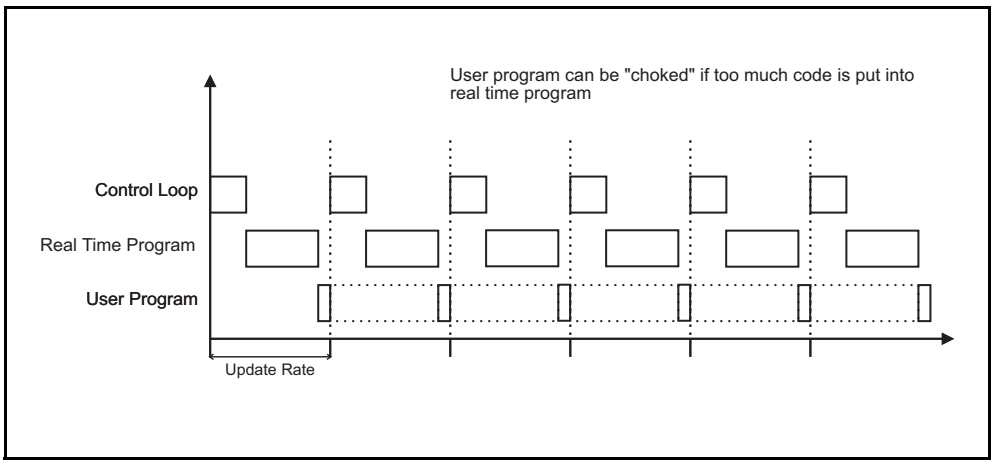


Figure 133: Diagram of User Program and a Real Time Program

When the Real Time program is to large and can not be completed in the update rate time the drive will fault.

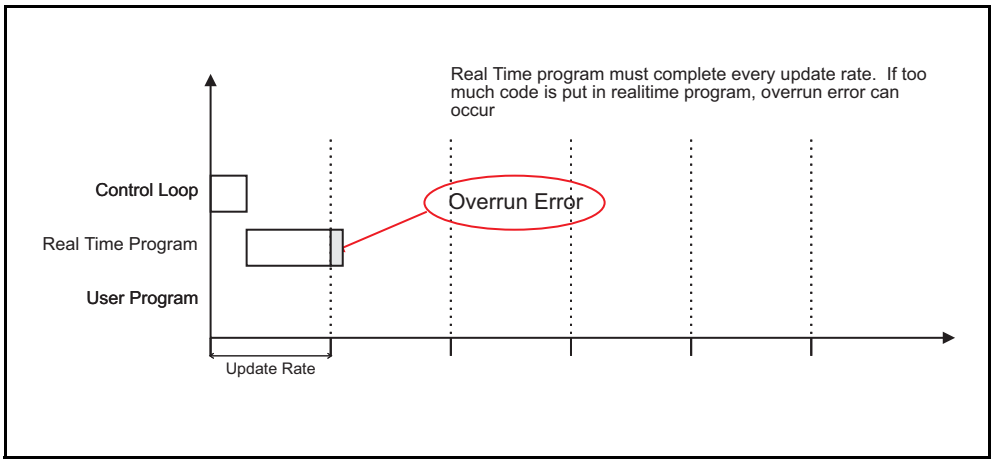


Figure 134: Diagram of User Program and a Real Time Program

Figure 135 shows how the Real Time program finishes and the two task user programs are processed.

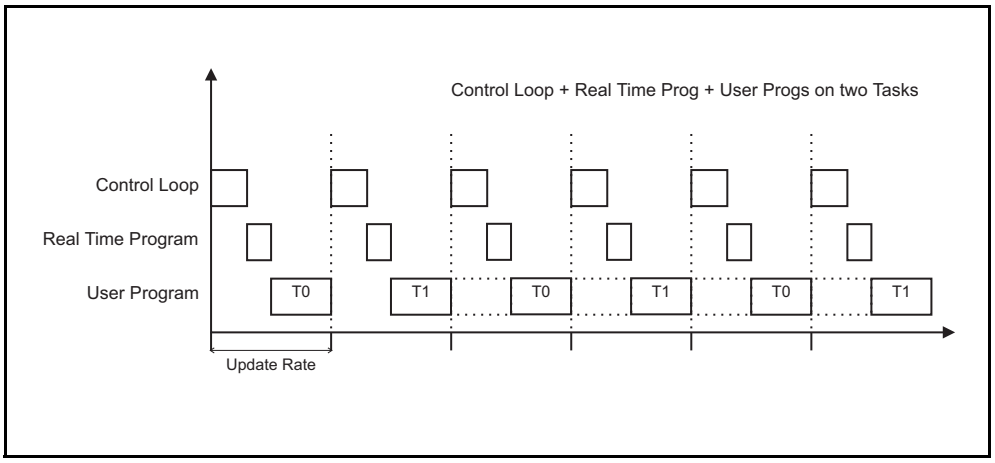


Figure 135: Diagram of User Programs on Two Tasks and a Real Time Program

The following three figures show how the drive processes Real Time, Cyclic, and User Programs with different number of tasks.

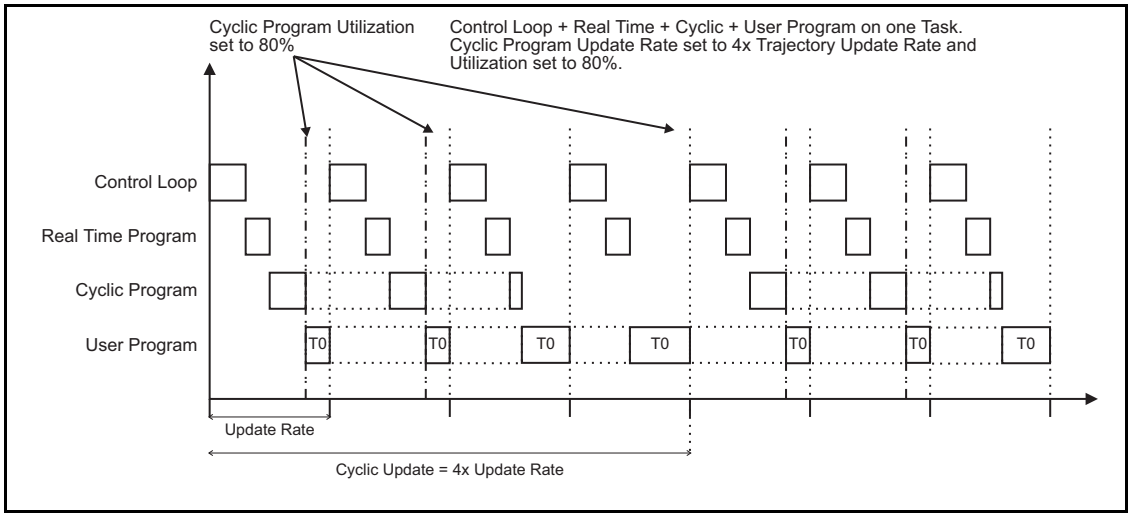


Figure 136: Diagram of User Program, a Real Time Program and a Cyclic Program

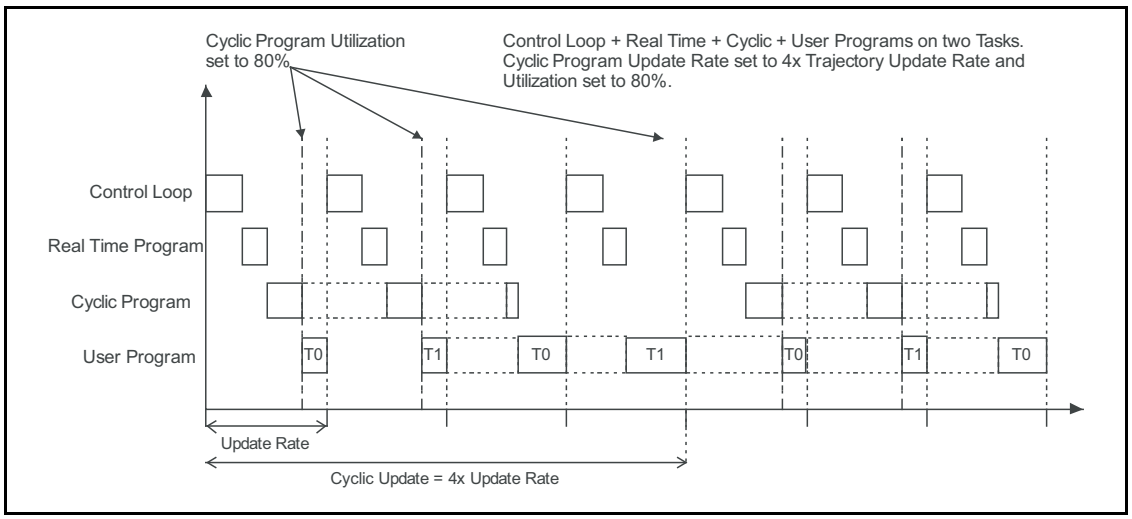


Figure 137: Diagram of User Program on Two Tasks, a Real Time Program and a Cyclic Program

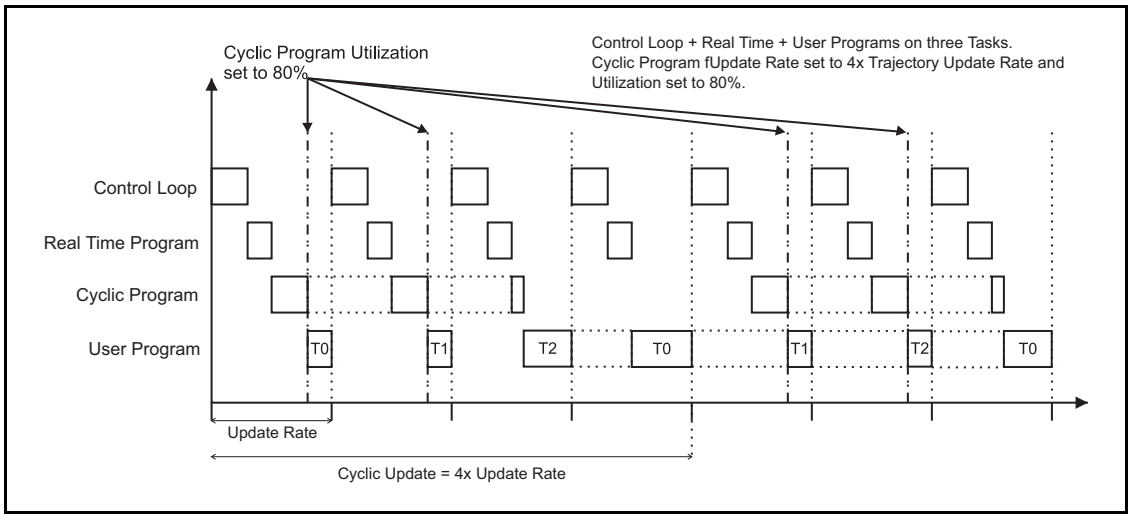


Figure 138: Diagram of User Programs on Three Tasks, a Real Time Program and a Cyclic Program

4.4 Program Instruction Types

4.4.1 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.

For the "For Count" instruction, break stops the instruction execution flow and continues execution after the Next instruction.

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 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

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 expression 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 number" 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 DriveInput.1=ON        'Repeat the three lines of code below
                                'as long as DriveInput.1 is ON.
    Index.1.Initiate            'Incremental,Dist=5.250in,Vel=10.0in/s
    Dwell For Time 1.000        'seconds
Loop
```

Example:

```
Do While (DriveInput.1=ON AND DriveInput.2=OFF)
                                'Repeat the three lines of code below
                                'as long as DriveInput.1 is ON and
                                'DriveInput.2=OFF.
    Index.1.Initiate            'Incremental,Dist=5.250in,Vel=10.0in/s
    Dwell For Time 1.000        'seconds
Loop
```

Example:

```
Do While (TRUE)                 'Repeat until the program is halted
    Index.1.Initiate            'Incremental,Dist=5.250in,Vel=10.0in/s
    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 DriveInput.1=ON Then          'The following two lines are executed if
                                 'DriveInput.1=ON
    DriveOutput.1=ON
    DriveOutput.2=OFF
Else                             'The following two lines are executed if
                                 'DriveInput.1=OFF
    DriveOutput.1=OFF
    DriveOutput.2=ON
Endif
```

Example:

```
If (DriveInput.5=ON) Then        'Set fast velocity if DriveInput.5 = ON
    Jog.0.Vel = 1.0 'in/s
Else                             'Set slow velocity if DriveInput.5 = OFF
    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 DriveInput.1=OFF Then
    End
Endif
```

Example:

```
If DriveInput.1=ON Then
    DriveOutput.1=ON
    End
Endif
```

For Count/Next

This instruction is used to execute section of code a specific number of times.

Example:

```
For Count = 1 to 5
    Index.1.Initiate             'Incremetal,Dist=5.250in,Vel=10.0in/s
    Dwell For Time 1.000         'seconds
Next
```

Example:

```
For Count = 1 To 10
    Wait For DriveInput.1 = ON
    Index.0.Initiate             'Incremetal,Dist=5.000in,Vel=50in/s
    Wait For Index.AnyCommandComplete
    DriveOutput.1=ON             'Turn DriveOutput.1 On
    Wait For Time 1.000          'seconds
    DriveOutput.1=OFF            'Turn DriveOutput.1 Off
Next
```

Formula

This program instruction can be used to enter a formula or assignment into a program. All 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 the section on User Level in the Setting Up Parameters chapter). 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
DriveOutput.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> = An expression 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> Greater 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> = An expression 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.

Example:

```
For var.var0 = 1 To 20 Step .5
Next
```

The Break instruction is allowed within the For<var> instruction loop.

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 RedDot 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 (DriveInput.1 = ON) Then
    GoTo RunIndex1:                ‘Go to RunIndex1 label
Else
    GoTo RunIndex2:                ‘Go to RunIndex2 label
EndIf
RunIndex1:                        ‘If Input.1 is on, resume here
Index.1.Initiate
GoTo EndLoop:                     ‘GoTo EndLoop label
RunIndex2:                        ‘If Input.1 is off, resume here
Index.2.Initiate
EndLoop:
Wait For Index.AnyCommandComplete
Loop
```

See the Label: instruction for additional examples.

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.

Examples:

```
If DriveInput.1=ON Then           ’Turn Outputs 1 On and 2 Off if Drive
                                  ’Input.1 is ON.
    DriveOutput.1=ON
    DriveOutput.2=OFF
Endif

If (DriveInput.1=ON AND DriveInput.2=OFF) Then
                                  ‘Turn Outputs 1 On and 2 Off if Drive
                                  ‘Input.1 is ON and DriveInput.2 is OFF
    DriveOutput.1=ON
    DriveOutput.2=OFF
Endif

If (DriveInput.2=ON) Then         ‘Jog+ when DriveInput.2=ON
    Jog.0.PlusInitiate            ‘Vel=20in/s
    Wait For DriveInput.2=OFF     ‘Stop when the input goes OFF
    Jog.Stop                      ‘Decelerate to a stop
```

```
Endif

If (DriveInput.3=ON) Then         ‘Jog- when DriveInput.3=ON
    Jog.0.MinusInitiate           ‘Vel=20in/s
    Wait For DriveInput.3=OFF     ‘Stop when the input goes OFF
    Jog.Stop                      ‘Decelerate to a 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 RedDot 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 (DriveInput.2 = ON) Then
    GoTo Start:                   ‘ Go to Start label if Input2 on
EndIf
DriveOutput.1 = ON
End
```

See GoTo instruction for additional examples.

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. In the following example if the value of var.var0 is one then case <1> will be executed. When the break instruction is encountered the program will continue with the instruction after the EndSwitch command.

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 (DriveInput=ON, AtVel, etc.) as well as comparisons (PosnFeedback > 1234, VelFeedback < 100, etc.) can be used in a Wait For instruction.

Example:

```
Wait For (DriveInput.1=ON AND DriveInput.2=OFF)
Index.0.Initiate
Wait For Index.AnyCommandComplete

If (DriveInput.2=ON) Then         ‘Jog+ when DriveInput.2=ON
    Jog.0.PlusInitiate            ‘Vel=20in/s
    Wait For DriveInput.2=OFF     ‘Stop when the input goes OFF
    Jog.Stop                      ‘Decelerate to a stop
Endif
If (DriveInput.3=ON) Then         ‘Jog- when DriveInput.3=ON
    Jog.0.MinusInitiate           ‘Vel=20in/s
    Wait For DriveInput.3=OFF     ‘Stop when the input goes OFF
```

```
    Jog.Stop                        'Decelerate to a stop
Endif
```

Example:

```
Wait For (MasterAxis.PosnFeedback > 1000.00)
DriveOutput.1 = ON
Wait For (VelFeedback > 50.00)
DriveOutput.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 an 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.

Examples:

```
Wait For Time 5.000               'seconds

Do While (TRUE)                   'Repeat until the program is halted
    Index.1.Initiate              'Incremental,Dist=25.250in,Vel=10.0in/s

    Wait For AtVel
                                  'Turn Output 1 ON for 1 second, after the 'index
                                  reaches its' target velocity
    DriveOutput.1=ON
    Wait For Time 1.000           'seconds
    DriveOutput.1=OFF
    Wait For Index.AnyCommandComplete
Loop
```

4.4.2 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.

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`

BitAnd

This operator may be used when it is desirable to AND each individual bit of a 32-bit parameter. This operator can be found by clicking on the ***Drag in Operands*** button to open the Select Math Operator window.

```
var.var2 = var.var0 bitand var.var1
```

Example: if var.var0 = 1000 and var.var1 = -1

```
var.var0 = 00000000000000000000001111101000b
var.var1 = 11111111111111111111111111111111b
var.var2 = 00000000000000000000001111101000b
```

BitOr

This operator may be used when it is desirable to OR each individual bit of a 32-bit parameter. This operator can be found by clicking on the ***Drag in Operands*** button to open the Select Math Operator window.

```
var.var2 = var.var0 bitor var.var1
```

Example: if var.var0 = 1000 and var.var1 = -10000

```
var.var0 = 00000000000000000000001111101000b
var.var1 = 11111111111111111101100011110000b
var.var2 = 11111111111111111101101111111000b
```

4.4.3 Program Array Access

Cam[ ].Master

This instruction allows the user to write to a cam tables master value at a given table number and element number.

Example:

Cam[0,3].Master =7

Cam[ ].Follower

This instruction allows the user to write to a cam tables master value at a given table number and element number.

Example:

Cam[1,2].Follower =3

Cam[ ].Interpolation

This instruction allows the user to write to a cam tables master value at a given table number and element number.

Example:

Cam[0,3].Interpolation =5

4.4.4 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 which notes that the dwell time is in units of seconds. The comment starts with the ‘ character.

Examples:

```
Do While (TRUE)
    Index.0.Initiate              ‘Incremetal,Dist=25.000in,Vel=25in/s
    Dwell For Time 1.000          ‘Seconds
Loop

Do While (TRUE)
    Index.0.Initiate              ‘Incremetal,Dist=25.000in,Vel=25in/s
    Dwell For Time 1.000          ‘Seconds
    Index.1.Initiate              ‘Incremental,Dist=15.000in,Vel=25in/s
    Dwell For Time 0.500          ‘Seconds
Loop
```

Dwell for Master Dist

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              'Synch,Incr,Dist=5.0 Inches,Vel=1 Inches/
MstrInch
    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:
```

CamInitate By Expression

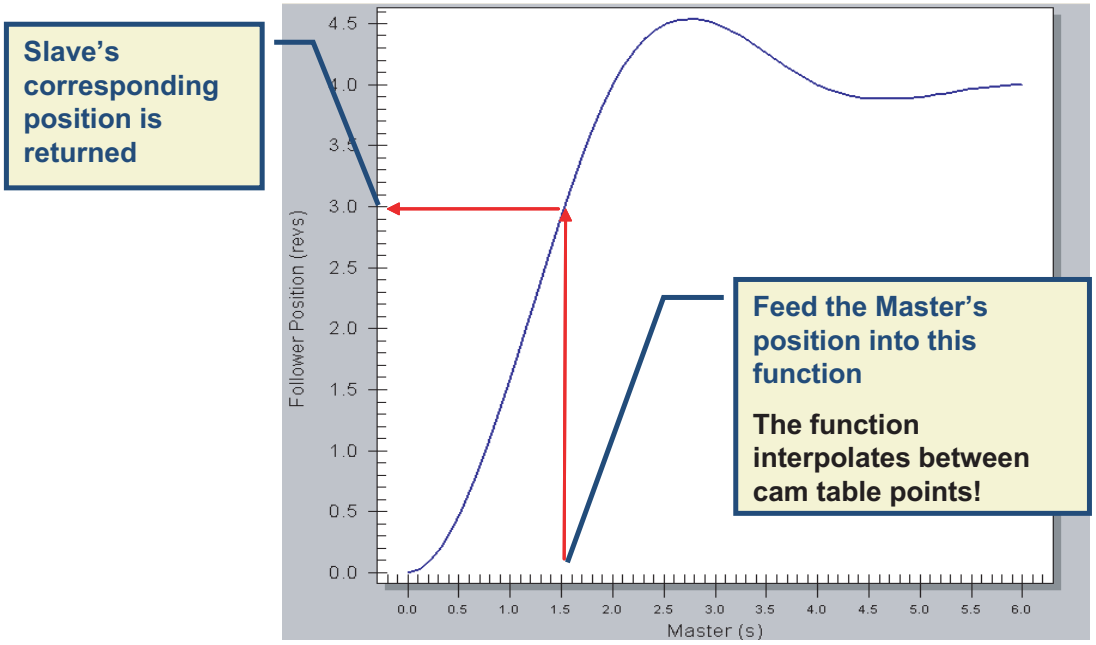
This motion instruction is used to initiate a cam. The cam is preset to include cam type, time base, direction, any chaining requirements, and a deceleration to a stop, optional. CamInitiate by Expression is used to initiate different Cams with the same line of code in a program.

The following example will initiate Cam Table 0, wait for complete, initiate Cam Table.1, wait for complete, index.2... etc.

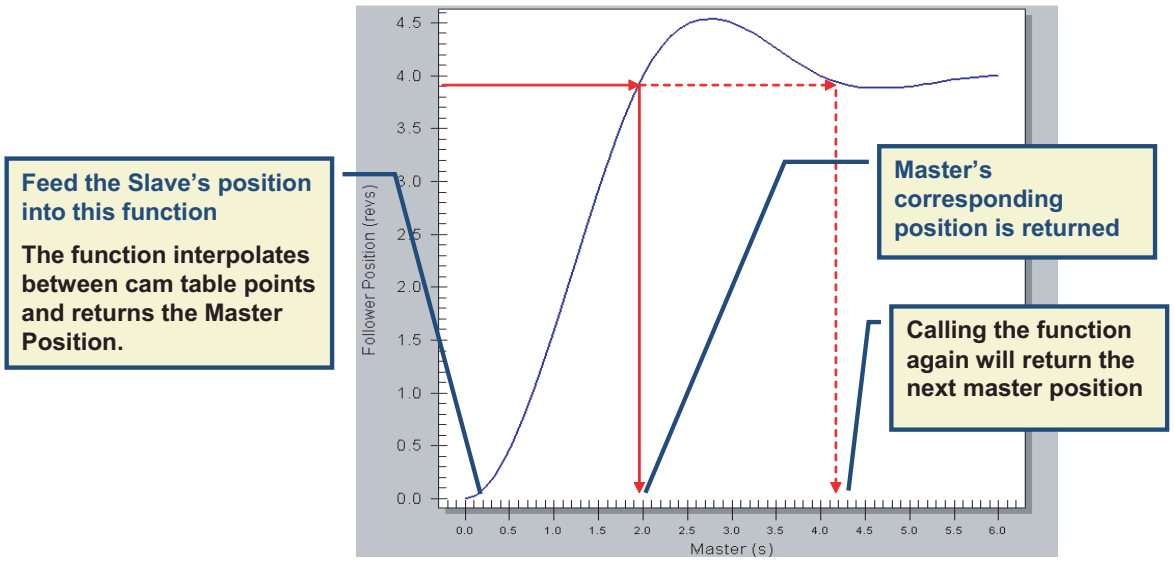
Example:

```
var.var0 = 0
a:
CamInitiate var.var0
Var.var0 = 1 + var.var0
wait for index.anycommandcomplete
goto a:
```

Set Cam Master Offset(...)



Set Cam Follower Offset(...)



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" to "Wait For InPosn". The In Position Window is configured in the Position view.

Example:

```
Index.0.Initiate                    'Incremental,Dist=5.000in,Vel=2.0in/s
Wait For Index.AnyCommandComplete

Index.37.Initiate                   'Absolute,Posn=120.60mm,Vel=50.2mm/s
Wait For InPosn
```

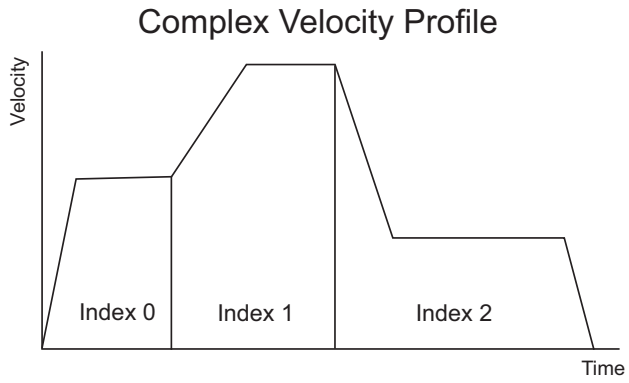
Index.CompoundInitiate

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.

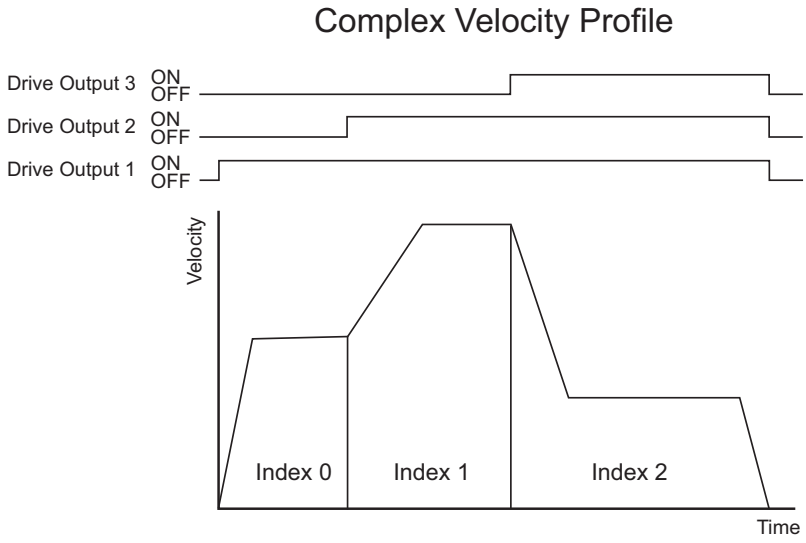
A comment is automatically inserted after the index instruction which shows key data about the particular index. The comment starts with the apostrophe ‘ character.

Examples:




```
Index.0.CompoundInitiate          ‘Incremental,Dist=5.000in,Vel=50in/s
Index.1.CompoundInitiate          ‘Incremental,Dist=20.000in,Vel=75in/s
Index.2.Initiate                  ‘Incremental,Dist=10.000in,Vel=30in/s
Wait For Index.AnyCommandComplete
```

Figure 139: Index Velocity Profile




```
Index.0.CompoundInitiate ‘Incremetal,Dist=5.000in,Vel=50in/s
DriveOutput.1=ON         ‘Turns ON immediately after Index.0 is started
Index.1.CompoundInitiate ‘Incremental,Dist=20.000in,Vel=75in/s
DriveOutput.2=ON         ‘Turns ON immediately after Index.1 is started
Index.2.Initiate         ‘Incremental,Dist=10.000in,Vel=30in/s
```

```
DriveOutput.3=ON         'Turns ON immediately after Index.2 is started
Wait For Index.AnyCommandComplete
DriveOutput.1=OFF        'Turns OFF after Index.2's command is completed
DriveOutput.2=OFF        'Turns OFF after Index.2's command is completed
DriveOutput.3=OFF        'Turns OFF after Index.2's command is completed
```

Figure 140: Index Velocity Profile with Drive Outputs

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.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 verus 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)
DriveOutput.1=ON
Index.1.BlendInitiate into (2)
DriveOutput.2=ON
Index.2.Initiate
DriveOutput.3=ON
Wait For Index.AnyCommandComplete
DriveOutput.1=OFF
DriveOutput.2=OFF
DriveOutput.3=OFF
```

Index.StopTracking

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.

Home.Initiate

This program instruction is used to initiate the home.

A comment is automatically inserted after the Home.Initiate instruction which 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.

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 DriveInput.2=ON         'Wait for "Jog –" input to turn on
Jog.0.MinusInitiate              'Vel=27.2in/s
Wait For DriveInput.2=OFF        'Wait for "Jog –" input to turn off
Jog.Stop                         'Decelerate to a stop

Do While(TRUE)                   'Repeat until the program is halted
    If (DriveInput.2=ON) Then    'Jog+ when DriveInput.2=ON
    Jog.0.PlusInitiate           'Vel=20in/s
    Wait For DriveInput.2=OFF    'Stop when the input goes OFF
    Jog.Stop                     'Decelerate to a stop
    Endif

    If (DriveInput.3=ON) Then    'Jog- when DriveInput.3=ON
```

```
        Jog.0.MinusInitiate           'Vel=20in/s
        Wait For DriveInput.3=OFF     'Stop when the input goes OFF
        Jog.Stop                      'Decelerate to a 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.Stop command.

Example:

Program 0 – Running on Task 0

```
Do While TRUE
    Index.1.Initiate
    Wait For Index.AnyCommandComplete
    Var.Counter = Var.Counter + 1
Loop
```

Program 1 – Running on Task 1

```
Wait For DriveInput.1 = ON
Program.0.Stop
```

NOTE

In the example above, when input 1 turns on, Program 0 will be stopped. If Index 1 was in progress when the program was stopped, the index will continue until it is complete.

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 115)

Gear Stop

Gear Stop will stop gearing motion that has been initiated from a program.

Example:

```
Gear.Initiate
Wait for DriveInput.2=ON
Gear.Stop
```

Gear.Initate

Gear Initiate will initiate gearing from a program. Gearing will remain active until the Gear.Stop command is used.

Example:

```
Gear.Initiate
Wait for DriveInput.2=ON
Gear.Stop
```

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).

4.4.5 Motion Modifiers

Timeline Control Instructions

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.

The Epsilon EP-P drive has program instructions to allow better control of the timeline. These instructions are the “Using Capture” and the “Using Last” instructions. Following is a description of each of the instructions:

On Profile

The On Profile instruction can be inserted after any motion type Initiate, Dwell For Time, or Dwell For MasterDist 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 view 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.

Examples:

```
Index.0.Initiate                  'Index 0 runs on Profile 0
Index.1.Initiate On Profile.1     'Index 1 runs on Profile 1

Gear.Initiate On Profile.1        'Gear operates on Profile 1
Wait For DriveInput.3 = ON
Gear.Stop On Profile.1            'Stop Gear running on Profile 1

Jog.0.PlusInitiate On Profile.1   'Jog 0 runs in positive direction on Profile 1
Index.0.Initiate                  'Index 0 runs on Profile 0
Wait For Index.AnyCommandComplete
Wait For DriveInput.2 = ON
Jog.Stop On Profile.1             'Stop Jog running on Profile 1
```

Using Capture

The Using Capture instruction can be inserted after any Jog Initiate, Index Initiate, Dwell For Time, and Dwell For MasterDist 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  ‘Index0,Incrmntl,Dist=5.0revs
```

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 Epsilon EP-P drive 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 Index1 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                    'Index0,Incrmntl,Dist=1.5revs
Dwell For Time 1.000 Using Last
Index.1.Initiate Using Last         'Index1,Incrmntl,Dist=3.5revs
```

4.4.6 Modbus Slave

ReadBit

Reads bit values (coils or inputs) from the slave into user bits.

```
ReadBit (<SlaveId>, <ModbusAddr>, <Qty>, <BitInstance>)
```

<slaveId> =The instance number of a defined slave. See Network Modbus Master Slave tab,

<ModbusAddr> = The starting modbus address of the slave coil or input.

Range	Description
00001 - 09999	one coil for each user bit
10001 - 19999	one input for each user bit

<Qty> = The number of coils or inputs to be read which equals the number of user bits

<BitInstance> = The starting bit instance number that the coil or input results will be written to

This is a block transfer command that can load several coil status bits or inputs into User Bits.

The program will not proceed to the next instruction until either the data transfer is complete or an error has occurred.

Example:

```
' Read from slave 2 - 3 coils (10, 11 and 12) and stores the result
' in user bits bit.1, bit.2 and bit.3
ReadBit (2, 10, 3, 1)
```

Example:

```
' Read from slave 1 - n(var.qty) coils starting at coil var.from
' and store the results in n(var.qty) user bits starting at
' bit instance var.to.
var.from = 39980
var.qty = 3
var.to = 1
ReadBit (1, var.from, var.qty, var.to)
```

ReadVar

This instruction reads values from the modbus slave to the user variables.

```
ReadVar (<SlaveId>, <ModbusAddr>, <Qty>, <VarInstance>)
```

<SlaveID> = The instance number of a defined slave. See Network Modbus Master Slave tab.

<ModbusAddr> = The starting modbus address of the slave holding or input register.

Range	Description
00001 - 09999	Each multiple of 16 coils are packed into one user variable. <Qty> defines the number of coils to read
10001 - 19999	Each multiple of 16 coils are packed into one user variable. <Qty> defines the number of coils to read
30001 - 39999	16 bit read only slave register is loaded into the lower 16bits of a user variable
40001 - 49999	16 bit writable slave data is loaded into the lower 16bits of a user variable
130001 - 139999	Two 16 bit read only slave registers are loaded into the lower 16bits of a user variable and the upper 16 bits of the user variable
140001 - 149999	Two 16 bit writable slave registers are loaded into the lower 16bits of a user variable and the upper 16 bits of the user variable
400001 - 409999	Unidrive 32bit values: Two 16bit slave holding registers are written from the 32bit user variable with word swap as specified by the slave configuration.

<Qty> = The number of slave values to be read.

<VarInstance> = The starting user variable instance number that the slave values will be written to. Decimal point of the user variable will not be altered.

This a block transfer command that can load several holding or input registers into user variables.

The program will not proceed to the next instruction until either the data transfer is complete or an error has occurred.

All operands can be numbers or expressions.

Example:

```
' Read from slave 2 - 3 input registers (39980, 39981 and 39982)
' and stores the result in user variables var.var1, var.var2 and
' var.var3
Recover (2, 39980, 3, 1)
```

Example:

```
' Read from slave 1 - n(var.qty) registers starting at register
' var.from and store the results in n(var.qty) user variables
' starting at var instance var.to
var.from = 39980
var.qty = 3
var.to = 1
ReadVar (1, var.from, var.qty, var.to)
```

WriteBit

This instruction writes bit values (coil) to the slave from user bits.

```
WriteBit (<SlaveId>, <ModbusAddr>, <Qty>, <BitInstance>)
```

<SlaveID> = The instance number of a defined slave. See Network Modbus Master Slave tab.

<ModbusAddr> = The starting modbus address of the slave coil.

Range	Description
00001 - 09999	one coil for each user bit

<Qty>= The number of coils to be written which equals the number of user bits

<BitInstance> = The starting user bit instance number that the coil values will be set to.

This is a block transfer command that can set several coil status bits from the User Bits.

The program will not proceed to the next instruction until either the data transfer is complete or an error has occurred.

All operands can be numbers or expressions.

Example:

```
' Write to slave 2 - 2 coils (23, and 24) the values set in
' in user bits bit.3, and bit.4
WriteBit (2, 23, 2, 3)
```

Example:

```
' Write to slave 1 - n(var.qty) coils starting at coil var.to
' the values stored in n(var.qty) user bits starting at
' bit instance var.from
var.to   = 39980
var.qty  = 3
var.from = 1
WriteBit (1, var.to, var.qty, var.from)
```

WriteVar

This instruction writes values (coils or holding registers) to the slave from user variables.

```
WriteVar (<SlaveId>, <ModbusAddr>, <Qty>, <VarInstance>)
```

<SlaveId> = The instance number of a defined slave. See Network Modbus Master Slave tab.

<ModbusAddr> = The starting modbus address of the slave holding register

Range	Description
00001 - 09999	Each multiple of 16 coils are written from the lower 16 bits of user variable. <Qty> defines the number of coils to write
40001 - 49999	16 bit writable slave data is loaded into the lower 16bits of a user variable
140001 - 149999	Two 16 bit writable slave registers are loaded into the lower 16bits of a user variable and the upper 16 bits of the user variable
400001 - 409999	Unidrive 32bit values: Two 16bit slave holding registers are written from the 32bit user variable with word swap as specified by the slave configuration.

"Range 00001 to 09999 - Each multiple of 16 coils are extracted from one user variable. <Qty> defines the number of coils to write

"Range 40001 to 49999 the lower 16bits of a user variable is written to the 16 bit writable slave register

"Range 140001 to 149999 the lower 16bits of a user variable is written to the slave and the next modbus register is written with the upper 16 bits of the user variable

<Qty> = The number of slave values to be written

<VarInstance> = The starting user variable instance number that the slave values will be set from. Decimal points will be ignored.

This a block transfer command that can set several holding or input registers from user variables.

The program will not proceed to the next instruction until either the data transfer is complete or an error has occurred.

All operands can be numbers or expressions.

Example:

```
' Writes to slave 2 - 3 registers (41121, 41122 and 41123)
' the values in user variables var.var3, var.var4 and var.var5
WriteVar (2, 41121, 3, 3)
```

Example:

```
' Writes to slave 1 - n(var.qty) registers starting at register
' var.to the values in n(var.qty) user variables starting
' at var instance var.from
var.to   = 41121
var.qty  = 3
var.from = 3
WriteVar (1, var.from, var.qty, var.to)
```

WriteValue

This instruction will writes a single value to the slave

```
WriteValue (<SlaveId>, <ModbusAddr>, <Qty>, <Value>)
```

<SlaveId> = The instance number of a defined slave. See Network Modbus Master Slave tab

<ModbusAddr> = The starting modbus address of the slave coil or holding register

Range	Description
00001 - 09999	Each multiple of 16 coils are packed into one user variable. <Qty> defines the number of coils to read
10001 - 19999	Each multiple of 16 coils are packed into one user variable. <Qty> defines the number of coils to read
40001 - 49999	16 bit writable slave data is loaded into the lower 16bits of a user variable
140001 - 149999	Two 16 bit writable slave registers are loaded into the lower 16bits of a user variable and the upper 16 bits of the user variable
400001 - 409999	Unidrive 32bit values: Two 16bit slave holding registers are written from the 32bit user variable with word swap as specified by the slave configuration.

<Qty> = The number of Modbus addresses of the specific type to write with the given value

<Value> = An expression representing a 32 bit value. Decimal point for the Value will be ignored

This is a block transfer command that can load up to 32 modbus coils or two Holding registers.

When multiple coils are written, the low 16 bits of <value> is written least significant to most significant bit. If more than 16 coils are written, <Value> is re-used.

The program will not proceed to the next instruction until either the data transfer is complete or an error has occurred.

All operands can be numbers or expressions.

Example:

```
' Write 1x 45 to register 39980 on MbMaster.1
WriteValue (1, 39980, 1, 45)
```

Example:

```
' Write 1x the content of var.var1 to register var.var0 on MbMaster.1
WriteValue (1, var.var0, 1, var.var1)
```

4.5 Adding and Deleting Programs

Programs can be added or removed from the user configuration in three ways. Programs may only be added or deleted while offline.

4.5.1 PowerTools Toolbar Button Method

Add Program Button

The Add Program button (shown below) will add a new program to the user configuration. Programs are added in sequential order. Clicking on the button will add a program and bring you to the program-editing view allowing you to enter program instructions.

Delete Program Button

The Delete Program button (shown below) will delete a program from the user configuration. The highest numbered program will automatically be deleted unless a different program is selected on the Programs heading screen. To delete a specific program, click on the Programs branch in the Hierarchy Tree. From this view, select the specific program you wish to delete, and then click on the Delete Program button.

4.5.2 Power Tools Menu Bar Method

Adding a Program

From the PowerTools Pro menu bar, select Edit/New/Program. A program will be added in sequential order and you will be brought to the program-editing view allowing you to enter program instructions.

Deleting a Program

Navigate to the Programs view on the Hierarchy Tree, and select the program you wish to delete. From the PowerTools Pro menu bar, select Edit/Delete/Program. The selected Program will be deleted from the configuration.

4.5.3 Right Click Method

Adding a Program

Navigate to the Programs view in the Hierarchy Tree. Position your mouse pointer on the right side of the view and right-click. A selection menu will appear allowing you to add a New Program or Delete a Program. Click on New Program and a program will be added in sequential order and you will be brought to the program-editing view allowing you to enter program instructions.

Deleting a Program

Navigate to the Programs view in the Hierarchy Tree. Select the program you wish to delete and then right-click on your mouse. A selection menu will appear allowing you to add a New Program or Delete a Program. Click on Delete Program and the selected program will be deleted from the configuration.

4.5.4 Run Anytime Programs

The programming environment has been designed to automatically stop all programs when a fault occurs (regardless of what type of fault). Some applications require the ability to run a program as soon as a fault occurs of continue running a program even through a fault condition. In order to do this, a program must be classified as “Run Anytime”. To define a program to be able to run during a fault or while the drive is disabled, the “Run Anytime” check box must be selected in the Program view. Figure 141 shows an example of the “Run Anytime” check box after it has been selected.

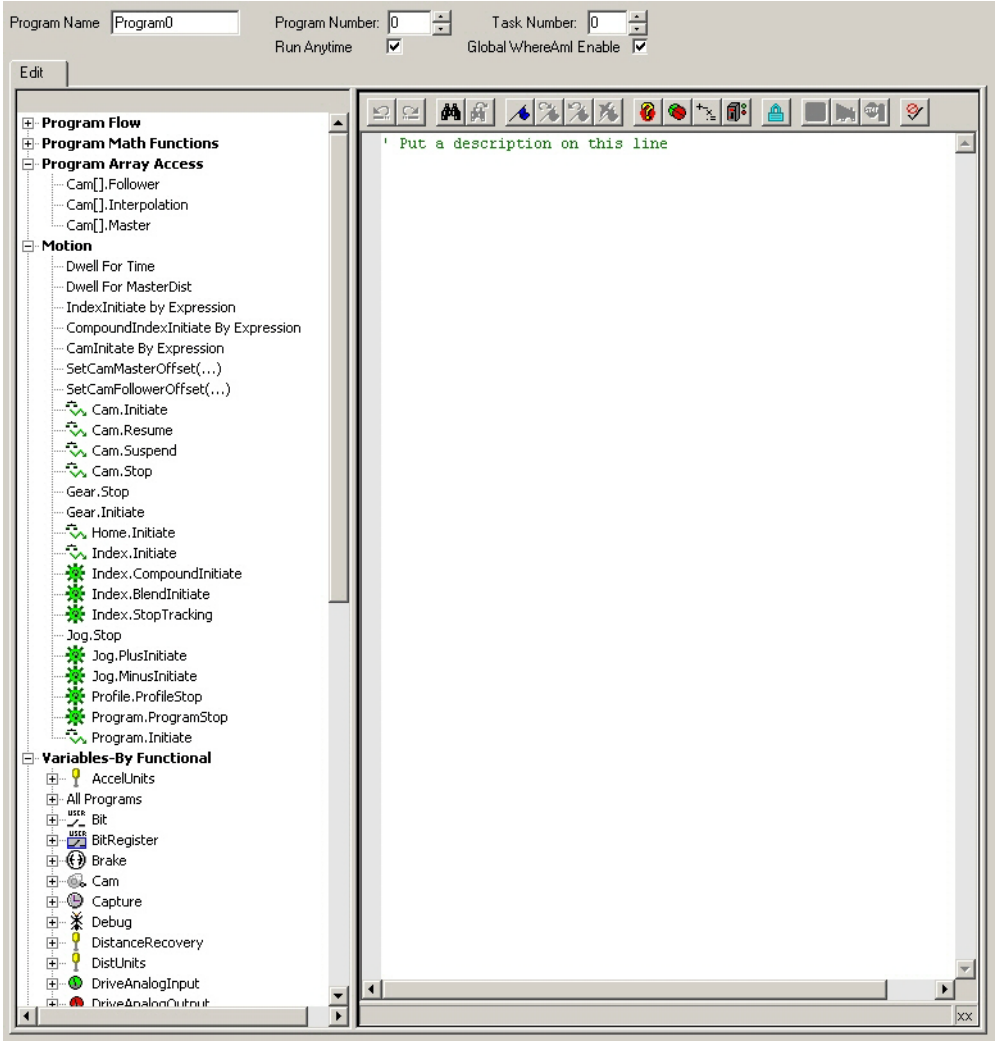


Figure 141: Program View with "Run Anytime" Check box Enabled

When a fault occurs, the drive will still be disabled, and no motion will be possible. For this reason, it may be necessary to reset the fault in the "Run Anytime" program prior to running 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 stop.

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 fault

Multiple programs may be configured as "Run Anytime" programs and can be called from a program the same as any other program. If a "Run Anytime" program calls another program which is not configured to run anytime while the drive is faulted or disabled, the task will be stopped.

Resetting Faults in "Run Anytime" Programs

To reset a fault from a "Run Anytime" program, use Fault.Reset = ON command in the user program. The Fault.Reset command does not clear all types of faults. Some faults require power to be cycled in order to clear the fault. For more information on the method used to clear individual faults, see the Diagnostics and Troubleshooting section.

After using the Fault.Reset command in a user program, use a Wait For Time 0.100'seconds command to give the drive time to clear the fault and re-enable the drive before initiating motion. If this is not done, the motion will be initiated before the drive is disabled. and the instruction will be ignored.

4.5.5 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  (XXXX)
```

The motion related instructions will only block the task for the remainder of the current update and the task will operate normally the next time it is processed. However, because the drive can currently only process one motion command at a time, a buffered motion command could cause the program to be blocked for a longer period of time.

For instance, if a program initiates Index0 and the next program instruction initiates Index1, the program will be blocked until Index0 is complete. This is because Index1 cannot start until Index0 is finished.

Blocked here until Index0 is complete →

```
Index.0.Initiate
Index.1.Initiate
DriveOutput.1=ON
```

A Dwell instruction is also a motion instruction and can block the program in the same way. The Dwell cannot start until Index0 is complete, and therefore the program (or task) is blocked until Index0 is finished.

Blocked here until Index0 is complete →

```
Index.0.Initiate
Dwell For Time 1.000 'Seconds
ModuleOutput.1=ON
```

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 142 shows the same time-slicing diagram as the previous figure, 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.

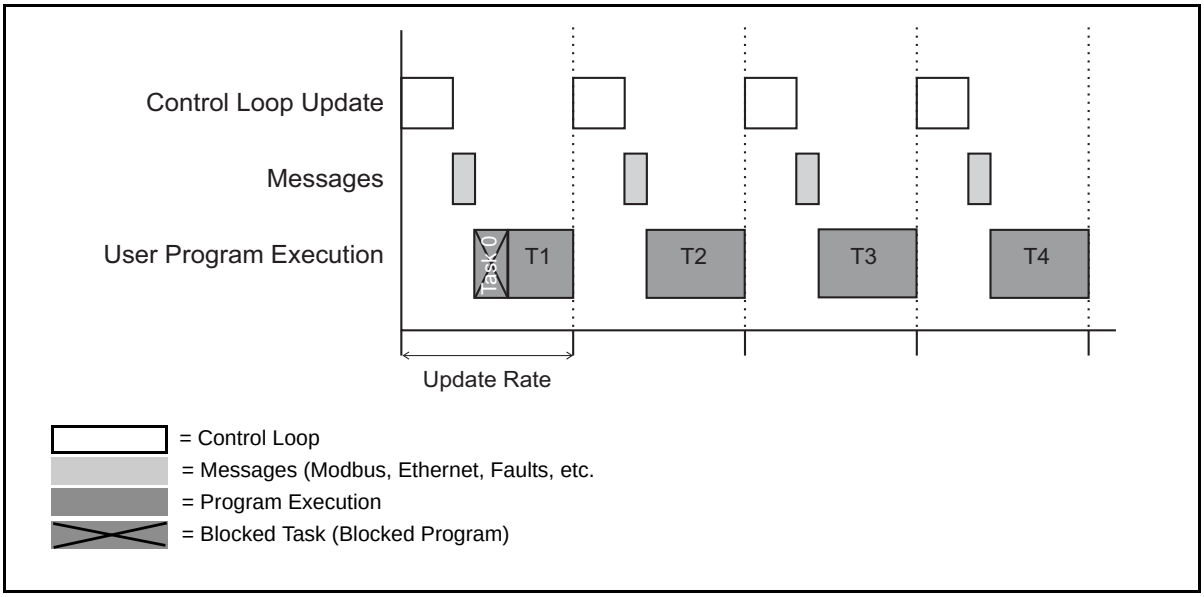


Figure 142: Block Time Slice

The time taken to process the blocked task and pass on to the next available task is between 50 and 100 microseconds.

Figure 143 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.

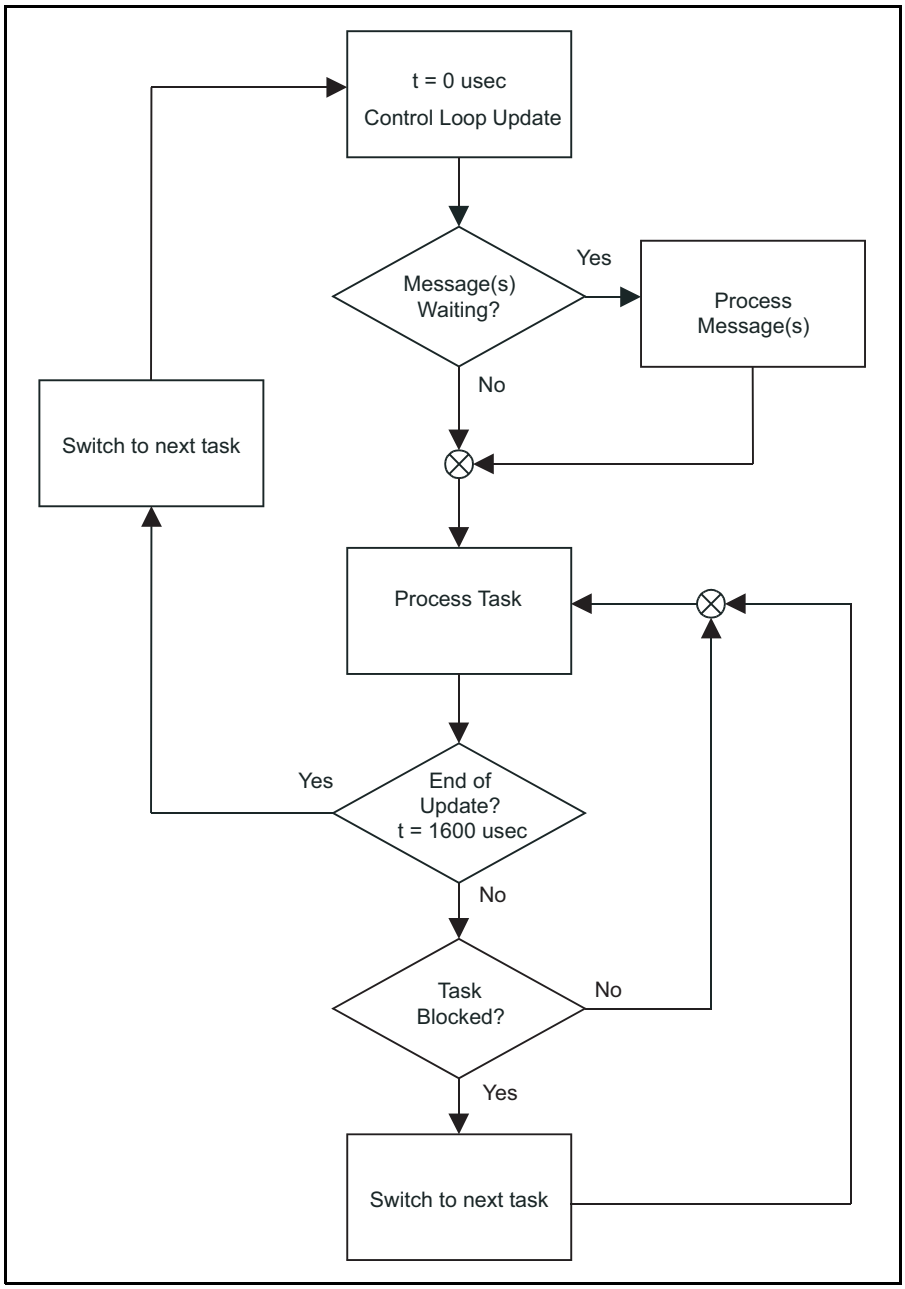


Figure 143: Time Slicing Flow Chart

4.6 Example Programs

Out and Return - Simple

Description: Move out to an absolute position and return

```
Index.2.Initiate                    'Absolute,Posn=10.000in,Vel=5.0in/s
Index.1.Initiate                    'Absolute,Posn=0.000in,Vel=10.0in/s
Wait For Index.AnyCommandComplete
```

Out and Return – More Complex

Description: Home, Wait For an input, Move out to an absolute position, set an output, dwell for 1 second, clear the output, return to home position, repeat the out and return sequence until the stop input halts the program.

```
Home.0.Initiate                     'Sensor,Offset=0.000in,Vel=-10.0in/s
DriveOutput.1=ON                    'Set the "At Position 1" output

Do While (TRUE)                     'Repeat until the program is halted
   Wait For DriveInput.2=ON         'Wait for the "Go" Input
```

```
    DriveOutput.1=OFF             'Clear the "At Position 1" output
    Index.2.Initiate              'Absolute,Dist=10.000in,Vel=5.0in/s
    Wait For InPosn
    DriveOutput.2=ON              'Set the "At Position 2" output
    Wait For Time 1.000           'Seconds
    DriveOutput.2=OFF             'Clear the "At Position 2" output
    Index.1.Initiate              'Absolute,Dist=0.000in,Vel=10.0in/s
    Wait For InPosn
    DriveOutput.1=ON              'Set the "At Position 1" output
Loop
```

Punch a Hole in a Web a Specified Distance Beyond a Registration Mark

Description: Index a web to a position 2 inches beyond a registration mark. Then fire a solenoid to punch a hole in the web. Wait for a sensor to indicate that the punch is in the down position. Retract the solenoid. Wait until it is sensed in the up position.

```
Do While (TRUE)                   'Repeat until the program is halted
    Index.0.Initiate              'Registration,Offet=2.0in,Dist=20in,Vel=20in/s
    Wait For InPosn
    DriveOutput.1=ON              'Fire the punch solenoid
    Wait For DriveInput.2=ON      'Wait for the "down" indicator
    DriveOutput.1=OFF             'Retract the punch solenoid
    Wait For DriveInput.3=ON      'Wait for the "up" indicator
Loop
```

Registration Index to Place a Product on a Conveyor After Each Lug

Registration Index (synchronized) to find the front edge of product, wait for input from a lug sensor and repeat.

```
Do While (TRUE)                   'Repeat until the program is halted
    Index.0.Initiate              'Sync,Registration,Offset=0.500,
                                  'Dist=50.000,Vel=1.000in/in
                                  'Registration move to product sensor.
                                  'Go to head of next product.
    Wait For Index.AnyCommandComplete
    Wait For DriveInput.2=ON      'Wait for lug sensor on master conveyor.
Loop
```

Elevator (Accumulator) with 100 Stop Positions

Home, when an input goes on move down to the next position. When the bottom position is reached, move back to home when the input goes on.

```
Home.0.Initiate                   'Sensor,Offset=0.00mm,Vel=100mm/s

Do While (TRUE)                   'Repeat until the program is halted
    For Count = 2 To 100          'Step to positions 2 - 100
                                  Wait For DriveInput.2=ON'Wait for "Go" input
                                  Index.2.Initiate'Incremetal,Dist=2.00mm,Vel=100mm/s
                                  Wait For InPosn
    Next

    Wait For DriveInput.2=ON      'Wait for "Go" input
    Index.1.Initiate              'Absolute,Dist=0.00mm,Vel=1000mm/s
    Wait For InPosn
Loop
```

Simple Jogging within a Program

Jog+ when DriveInput.2 goes ON and stop when it goes off. Jog- when DriveInput.3 goes ON and stop when it goes off. This could also be accomplished using the Jog input functions when there is no program running.

```
Do While (TRUE)                   'Repeat until the program is halted
    If(DriveInput.2=ON) Then      'Jog+ when DriveInput.2=ON
                                  Jog.0.PlusInitiate'Vel=20in/s
    Wait For DriveInput.2=OFF     'Stop jogging when DriveIput.2 goes OFF
    Jog.Stop                      'Decelerate to a stop
    Endif
    If (DriveInput.3=ON) Then     'Jog- when DriveInput.3=ON
    Jog.0.MinusInitiate           'Vel=20in/s
    Wait For DriveInput.3=OFF     'Stop jogging when DriveInput.3 goes OFF
```

```
        Jog.Stop                        'Decelerate to a stop
        Endif
    Loop
```

Rotary Table with "Calibrated" Stop Positions

Home the axis, wait for an input and then index to 3 different stop positions (absolute positions), wait for an input between indexes. The InPosn output function could be assigned to an output to indicate when the axis has completed the index and the following error is less than a specified amount. Since the indexes are to absolute positions they can be adjusted to "calibrate" the stop positions to account for mechanical non-linearity in the particular rotary table. A rollover position of 360.00 degrees would be entered into the setup view so that the system would take the shortest path (across the rollover) during the last move.

```
    Home.0.Initiate                     'Sensor,Offset=0.0deg/s,Vel=-1000deg/s

    Do While (TRUE)                     'Repeat until the program is halted
        Wait For DriveInput.1=ON        'DriveInput.1 is the "Go" input
        Index.2.Initiate                'Absolute,Dist=120.07deg,Vel=1000deg/s
        Wait For InPosn
        Wait For DriveInput.1=ON
        Index.3.Initiate                'Absolute,Dist=239.95deg,Vel=1000deg/s
        Wait For InPosn
        Wait For DriveInput.1=ON
        Index.1.Initiate                'Absolute,Dist=0.03deg,Vel=1000deg/s
        Wait For InPosn
    Loop
```

Flying Cutoff/Shear

Flying cutoff or flying shear application to perform synchronized out and return indexes which repeat every 100 inches of master travel.

Part Length = 100 inches

Knife Travel Distance = 20 inches

PLS.0 is used to initiate Index.0 every 100 inches. PLS.0 has an "ON" point at 0.000 inches and an "OFF" point at 90.000 inches. PLS.0 has a rollover position of 100.000 inches. The rollover position is used to set the part length. The source for PLS.0 is the master axis. The PLS is configured in the PLS view. The PLS output does not necessarily need to be connected to an output line on the drive because it is used within the program to initiate an index.

An Index PLS is used to fire the cutoff knife. The Index PLS is connected to Output #1 on the drive. The Index PLS is configured in the index setup view. The Index PLS for Index.1 has an "ON" point 2.000 inches into the index and an "OFF" point 18 inches into the index.

```
    Home.0.Initiate                     'Sensor,Offset=1.000in,Vel=-5.0 in/s

        MasterAxis.DefineHome=ON        'Set the master position to 0.0
        PLS.0.Enable=ON                 'Turn on PLS.0

    Do While (TRUE)                     'Repeat until the program is halted
        Wait For DriveInput.1=ON        'Input 1 is used as a "hold" input.
        If PLS.0.Status = ON Then       'If the PLS is already on you are too late.
        DriveOutput.4=ON                'Set a "Too Late" output
        End                             'Drop out of the program
        Endif
        Wait For PLS.0.Status=ON        'Start the Index when PLS.0 goes on
                                        '(every 100 inches).
        Index.1.Initiate                'Incremental,Sync,Dist=20.0in,Vel=1.0in/in
        Index.0.Initiate                'Absolute,Sync,Dist=0.0in,Vel=2.0in/in
        Wait For Index.AnyCommandComplete
    Loop
```

Synchronized Jog with Manual Phase Adjustment

The motor controls a lugged conveyor belt which is synchronized to another lugged conveyor belt. Jog.0 is configured as a "Synchronized" jog using the setup software. The program first homes the follower and then waits for an input from a sensor on the master axis lugs. When the input comes on the follower starts the synchronized jog. If the home is setup correctly the follower will be in perfect phase when it gets up to speed. If the follower gets out of phase with the master the operator can manually bring the it back into phase using "Advance" and "Retard" inputs. The program adjusts the phase of the follower axis by adjusting the jog velocity (Jog.0.Vel) when the operator hits one of the phasing inputs.

```
    Home.0.Initiate                     'Sensor,Offset=2.25in,Vel=10in/s
```

```
Jog.0.Vel=1.000                  ‘follower inches/master inch

Wait For DriveInput.1=ON         ‘Start when a master lug is detected
Jog.0.PlusInitiate               ‘Sync,Vel=1.000in/in

Do While (TRUE)                  ‘Repeat until the program is halted
    If (DriveInput.2=ON) Then    ‘Phase Advance when DriveInput.2=ON
    Jog.0.Vel=1.100              ‘follower inches/master inch
    Wait For DriveInput.2=OFF
    Jog.0.Vel=1.000              ‘follower inches/master inch
    Endif

    If (DriveInput.3=ON) Then    ‘Phase Retard when DriveIput.3=ON
    Jog.0.Vel=0.900              ‘follower inches/master inch
    Wait For DriveInput.3=OFF
    Jog.0.Vel=1.000              ‘follower inches/master inch
    Endif

Loop
```

Auger Filler with Inputs to Adjust the Fill Amount

Incremental indexes are used to squirt a specified amount of food product into a box. Inputs are used to adjust the index distance. It would be much simpler to adjust the index distance with an OIT-3165 operator interface panel, but inputs could be used as described below.

```
DriveOutput.3=OFF
DriveOutput.4=OFF

Do While (TRUE)                  ‘Repeat until the program is halted
    If (DriveInput.2=ON) Then    ‘Fill a box if the “Go” input is on.
    Index.1.Initiate             ‘Incremental,Dist=16.00oz,Vel=16.0oz/s
    Wait For Index.AnyCommandComplete
    Endif

‘Increase the fill amount once every time DriveInput.3 is pressed
    If((DriveInput.3=ON) AND (DriveOutput.3=OFF)) Then
    Index.1.Dist = Index.1.Dist + 0.10‘ounces
    DriveOutput.3=ON

‘DriveOutput.3 is used to make sure that the distance is
‘incremented only once each time DriveInput.3 is pressed.
    If (Index.1.Dist > 20) Then
    DriveOutput.1=ON             ‘Long index output
    Else
    DriveOutput.1=OFF
    Endif
    Endif

    If((DriveInput.3=OFF) AND (DriveOutput.3=ON)) Then
    DriveOutput.3=OFF
    Endif

‘Decrease the fill amount once every time DriveInput.4 is pressed.
    If ((DriveInput.4=ON) AND (DriveOutput.4=OFF)) Then
    Index.1.Dist = Index.1.Dist - 0.10 ‘ounces
    DriveOutput.4=ON

‘DriveOutput.4 is used to make sure that the distance is
‘incremented only once each time DriveInput.4 is pressed.
    If (Index.1.Dist < 12) Then
    DriveOutput.2=ON             ‘Short index output
    Else
    DriveOutput.2=OFF
    Endif
    Endif
```

```
    If((DriveInput.4=OFF) AND (DriveOutput.4=ON)) Then
    DriveOutput.4=OFF
    Endif
Loop
```

Sequence Learn and Playback

This example consists of three programs. The first program is used to learn 3 positions using “Jog+”, “Jog-“, “Jog Fast” and “Learn” inputs. The second program is called several times by the first program. The third program steps through the learned positions

Learn Program (Program 0)

```
Home.0.Initiate                  ‘Sensor,Offset=0.000in,Vel=-10in/s

Index.1.Initiate                 ‘Move to position 1
Wait For Index.AnyCommandComplete

Call Program.1                   ‘Program 1 allows the axis to be jogged
                                 ‘into position
If (DriveInput.1=ON) Then        ‘Learn the new position if the “Learn”
                                 ‘input is on
   Index.1.Dist = PosnCommand    ‘Read the Position Command into Index.1’s
                                 ‘absolute position.
Endif
Wait For DriveInput.1=OFF        ‘Wait until the “Learn” input goes off
Wait For DriveInput.2=OFF        ‘Wait until the “Skip” input goes off

Index.2.Initiate                 ‘Move to position 2
Wait For Index.AnyCommandComplete

Call Program.1                   ‘Program 1 allows the axis to be jogged
                                 ‘into position
If (DriveInput.1=ON) Then        ‘Learn the new position if the “Learn”
                                 ‘input is on
   Index.2.Dist = PosnCommand    ‘Read the Position Command into Index.2’s
                                 ‘absolute position.
Endif
Wait For DriveInput.1=OFF        ‘Wait until the “Learn” input goes off
Wait For DriveInput.2=OFF        ‘Wait until the “Skip” input goes off

Index.3.Initiate                 ‘Move to position 3
Wait For Index.AnyCommandComplete

Call Program.1                   ‘Program 1 allows the axis to be jogged
                                 ‘into position
If (DriveInput.1=ON) Then        ‘Learn the new position if the “Learn”
                                 ‘input is on
   Index.3.Dist = PosnCommand    ‘Read the Position Command into Index.3’s
                                 ‘absolute position.
Endif
Wait For DriveInput.1=OFF        ‘Wait until the “Learn” input goes off
Wait For DriveInput.2=OFF        ‘Wait until the “Skip” input goes off
```

Subroutine for Jogging the Axis into the Desired Position (Program 1)

```
‘Allow jogging until either the “Learn” input (DriveInput.1)
‘or the “Skip” input (DriveInput.2) goes ON.

Do While ((DriveInput.1=OFF) AND (DriveInput.2=OFF))
   If (DriveInput.3=ON) Then     ‘Jog+ if the Jog+ input is on
   Jog.0.PlusInitiate            ‘Vel=0.1in/s

   Do While (DriveInput.3=ON)
   If (DriveInput.5=ON) Then     ‘DriveInput.5 = “Jog Fast”
   Jog.0.Vel = 1.0               ‘in/s
   Else
   Jog.0.Vel = 0.1               ‘in/s
```

```
        Endif
        Loop

        Jog.Stop                      'Stop jogging when the Jog+ input goes off.
        Endif

        If (DriveInput.4=ON) Then     'Jog- if the Jog- input is on
        Jog.0.MinusInitiate           'Vel=0.1in/s

        Do While (DriveInput.4=ON)
        If (DriveInput.5=ON) Then     'DriveInput.5 = "Jog Fast"
        Jog.0.Vel = 1.0               'in/s
        Else
        Jog.0.Vel = 0.1               'in/s
        Endif
        Loop

        Jog.Stop                      'Stop jogging when the Jog+ input goes off.
        Endif
    Loop
```

Playback Program (Program 2)

```
    Home.0.Initiate                   'Sensor,Offset=0.000in,Vel=-10in/s

    Do While (TRUE)                   'Repeat until the program is halted
        Index.1.Initiate              'Absolute,Posn=1.000in,Vel=5in/s
        Wait For InPosn
        DriveOutput.1=ON              'Turn on DriveOutput.1 for 1 second
        Wait For Time 1.000           'seconds
        DriveOutput.1=OFF

        Index.2.Initiate              'Absolute,Posn=20.000in,Vel=7in/s
        Wait For InPosn
        DriveOutput.1=ON              'Turn on DriveOutput.1 for 1 second
        Wait For Time 1.000           'seconds
        DriveOutput.1=OFF

        Index.3.Initiate              'Absolute,Posn=5.250in,Vel=10in/s
        Wait For InPosn
        DriveOutput.1=ON              'Turn on DriveOutput.1 for 1 second
        Wait For Time 1.000           'seconds
        DriveOutput.1=OFF
    Loop
```

5 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 Position Valid
AbsolutePosnValid

This source is activated when either the DefineHome destination is activated, or any home routine is successfully completed (sensor or marker found). This source is deactivated if the drive is rebooted, an encoder fault occurs, the drive is powered down, a home is re-initiate, or the UndefinedHome destination activates.

Accelerating
Accelerating

This source is active when the drive 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.

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 drive 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#

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 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).

Brake Activate
Brake.Activate

This destination, when activated, engages the brake. This is simply used to manually engage the brake outside of the normal brake operation. This is level sensitive.

Brake Disengaged
Brake.Disengaged

This source is used to control the motor holding brake. When it is "off" the brake is mechanically engaged. When the brake is engaged, the diagnostic display on the front of the drive will display a "b". The drive outputs are limited to 150 mA capacity, therefore, a suppressed relay is required to control the brake coil. Model BRM-1 may be used.

Brake Release
Brake.Release

This destination will release the brake under all conditions, even when Brake.Activate is engaged. When this input function is active, the Brake.Disengaged output function (source) will be activated. This is used as a manual brake override.This is level sensitive.

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 is 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 CamPlus is selected the cam only moves in the forward direction and only chains in the forward direction. If in CamPlus 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 CamPlus 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

Accepts Motion Modifiers	Action
= 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 cam 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.

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.

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
Captured.#.CapturedMasterPosition

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.

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.

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 drive 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.

Commutation Angle Correction
CommutationAngleCorrection

The difference between the electrical angle as determined at power up from the U, V, and W commutation tracks and the electrical angle as determined from the marker pulse or UVW transitions. This value will be zero until the marker pulse is detected or UVW transition is detected.

Commutation Track Angle
CommutationTrackAngle

This parameter is derived directly from the state of commutation tracks and the Encoder U Electrical Angle parameter.

Commutation Voltage
CommutationVoltage

This parameter is used to determine commutation angle accuracy. When queried it returns the value of the direct axis voltage. The value is given as a percentage of 1/2 the bus voltage.

Decelerating
Decelerating

This source is active when the drive 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.

Define Home Position
DefineHomePosn

This is the value to which the position command will be set when the Define Home Destination is activated, This is used in applications that do not use a home routine, but is required to define a known reference point.

Acceleration
DistanceRecovery.Accel

This parameter in user units is the acceleration rate for the distance recovery index.

Deceleration
DistanceRecovery.Decel

This parameter in user units is the deceleration rate for the distance recovery index.

Enable Distance Recovery
DistanceRecovery.DistanceRecoveryEnable

Select the check box 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 the section on the User Units View in the Setting Up Parameters chapter.

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 the section on the User Units View in the Setting Up Parameters chapter.

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.

Enable Channel Check Box
DriveAnalogInput.ChannelEnable

By default, the analog input channel is enabled meaning that the drive is reading the A/D value read by the analog circuit. If the check box is clear, the channel is not enabled and the configuration parameters for the analog input is unavailable and therefore has no effect.

To enable the analog input, simply select the Enable Channel check box, and the configuration parameters will become available to edit. With the channel enabled, the trajectory loop update will transfer data from the drive into DriveAnalogInput.RawValue as volts and into DriveAnalogInput.ValueIn scaled.

Destination Variable
DriveAnalogInput.CustomDestinationVariable

PowerTools Pro displays a text box that the user can enter any available parameter using the dot notation format for the variable on the Analog Inputs View.

When the Popup Variables... button is pressed a window will open containing the list of variables that can be dragged into the Destination Variable text box.

Destination Variable is used to allow Position Tracker or Correction index types to perform an index to the scaled analog value. Set the Destination Variable to an index distance parameter. When the index type is set to Position Tracker, the Analog to Position scaling is used to scale the index's absolute position. When the index type is set to Correction, the Analog to Distance scaling is used to scale the index's incremental distance.

Bandwidth
DriveAnalogInput.LowPassFilterFrequency

This parameter sets the low-pass filter cutoff frequency applied to the analog input. Signals exceeding this frequency will be filtered at a rate of 20 db per decade.

Maximum Value
DriveAnalogInput.MaxUserValue

Enter the maximum value in analog user units that the maximum analog voltage should correspond to. Set Maximums Group.

Maximum Voltage
DriveAnalogInput.MaxVoltageValue

Enter the maximum voltage that will be seen on the analog input terminals. The user can enter the value in this field by hand, or set the analog source to it's maximum value, and click on the "Set Max Voltage to Measured" button next to the text box on the Analog Inputs view. This will read the current value on the analog channel and enter it into the Maximum Voltage text box.

Minimum Value
DriveAnalogInput.MinUserValue

Enter the minimum value in analog user units that the minimum analog voltage should correspond to.

Minimum Voltage
DriveAnalogInput.MinVoltageValue

Enter the minimum voltage that will be seen on the analog input terminals. The user can enter the value in this field by hand, or set the analog source to it's minimum value by clicking on the "Set Min Voltage to Measured" button next to the text box. This will read the current value on the analog channel and enter it into the Minimum Voltage text box. This parameter is used

for user units scaling.

A/D Voltage
DriveAnalogInput.RawValue

This is the raw analog input value in Volts.

Set Max Voltage To Measured
DriveAnalogInput.SetMax

Activate to set the maximum voltage to the current reading of the Analog Input channel.

Set Min Voltage To Measured
DriveAnalogInput.SetMin

Activate to set the minimum voltage to the current reading of the Analog Input channel.

Enable Channel
DriveAnalogOutput.#.ChannelEnable

By default, the analog output channel is not enabled meaning that a value is not being sent to the analog circuit. When the channel is not enabled (cleared), the configuration parameters for that analog output is unavailable and therefore has no effect. To enable the output, simply select the Enable Channel check box, and the configuration parameters will become available to edit.

If the user wishes to control the Analog Output through other means, it is necessary to clear the Enable Channel check box.

Custom Variable
DriveAnalogOutput.#.CustomSourceVariable

The Custom Variable parameter is only available once the user has selected Custom Variable from the Source list box. The field is used to define what parameter will control the Analog Output. This means that the selected variable parameter will directly determine the value of the analog output based on the Max and Min scaling values entered on this view. When the Popup Variables... button is pressed a window will open containing the list of variables that can be dragged into the Custom Variable text box.

Drive Analog Output Feedback
DriveAnalogOutput.#.Feedback

Displays the Output voltage from one of the two analog outputs found on the drive analog/sync connector.

Analog Output - Maximum Output
DriveAnalogOutput.#.MaxOutputValue

The analog output is a linear interpolation of the selected module variable between the minimum and maximum specified end points. Each end point is specified as the user value and the corresponding output value at that point. The number of decimal places for both values is taken from the selected source variable. MaxOutputValue is the maximum analog output value which corresponds to the maximum user value.

Analog Output - Maximum Value
DriveAnalogOutput.#.MaxUserValue

The analog output is a linear interpolation of the selected module variable between the minimum and maximum specified end points. Each end point is specified as the user value and the corresponding output value at that point. The number of decimal places for both values is taken from the selected module variable. MaxUserValue is the maximum user unit value which corresponds to the maximum analog output value.

Analog Output - Minimum Output
DriveAnalogOutput.#.MinOutputValue

The analog output is a linear interpolation of the selected module variable between the minimum and maximum specified end points. Each end point is specified as the user value and the corresponding output value at that point. The number of decimal places for both values is taken from the selected module variable. MinOutputValue is the minimum analog output value which corresponds to the minimum user value.

Analog Output - Minimum Value
DriveAnalogOutput.#.MinUserValue

The analog output is a linear interpolation of the selected module variable between the minimum and maximum specified end points. Each end point is specified as the user value and the corresponding output value at that point. The number of decimal places for both values is taken from the selected module variable. MinUserValue is the minimum user unit value which corresponds to the minimum analog output value.

Drive Enable Status
DriveEnableStatus

This source is active when the drive is enabled.

Power Up (counts)
DriveFaultLog.#.PowerUpCounts

This indicates during which power up the fault occurred in.

Time (minutes)
DriveFaultLog.#.Time

This indicates the time into the power up that the fault occurred.

Drive Input Debounced
DriveInput.#.Debounced

This displays the state of the input after the debounce is taken into account.

Drive Input Debounce Time
DriveInput.#.DebounceTime

The Drive Input Debounce Time parameter is the minimum time a digital input must be steady in order to be recognized by the drive. This feature helps prevent false triggering in applications in electrically noisy environments.

Drive Input Force
DriveInput.#.Force

Input can be forced either On or Off. This parameter is the state to which the input will be forced when the ForceEnable bit is activated.

Drive Input Force Enable
DriveInput.#.ForceEnable

If DriveInput.#.ForceEnable parameter is activated, then the state of the DriveInput.#.Force bit will override the current input state.

Drive Input Name
DriveInput.#.Name

This is a text string up to 12 characters that can be assigned to a given input. It allows the user to use application specific terminology in naming digital inputs.

Drive Input Raw
DriveInput.#.Raw

This displays the raw state of the digital input without debounce or forcing to override the raw status.

Drive Input Status
DriveInput.#.Status

This source is the state of the input after debounce and forcing are taken into account.

Drive Output Force
DriveOutput.#.Force

A drive output can be forced either On or Off with this parameter. If the ForceEnable bit is activated, the DriveOutput.#.State will be set to this value.

Drive Output Force Enable
DriveOutput.#.ForceEnable

If DriveOutput.#.ForceEnable parameter is activated, then the state of the DriveOutput.#.Force bit will override the current output state.

Drive Output Name
DriveOutput.#.Name

This is a text string up to 12 characters that can be assigned to a given output. It allows the user to use application specific terminology in naming digital outputs.

Drive Output State
DriveOutput.#State

This destination sets the current state of an output line.

Drive Output Encoder Scaling
DriveOutputEncoder.Scaling

This parameter allows scaling of the drive encoder output resolution in increments of one line per revolution. Allowable range is from one line per revolution up to the actual density of the encoder in the motor. If the Encoder output scaling is set greater than the motor encoder density the output scaling will be equal to the motor encoder density.

Drive Output Encoder Scaling Enable
DriveOutputEncoder.ScalingEnable

When on, this parameter enables the use of the drive encoder output scaling feature.

Drive Serial Number
DriveSerialNumber

This displays the serial number of the Drive.

Dual Loop Enable
DualLoopEnable

This Destination is used to initiate, turns on, the Dual Loop feature. The system will remain in this mode as long as this Destination is active. DualLoopEnable can be assigned or initiate by a user program.

Dual Loop Encoder Ratio - Motor Encoder
DualLoopMotorEncoder

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.

Motor - Position Feedback
DualLoopMotorPosnFdbk

This is the actual motor position in user units. The difference between the PosnCommand and the PosnFeedback is the FollowingError.

Motor - Velocity Feedback
DualLoopMotorVelFdbk

This is the feedback (or actual) velocity. It is calculated using the change in position of the motor encoder. It will always return the actual motor velocity - even in synchronized applications in which the master axis is halted during a move.

Dual Loop Encoder Ratio - Position Encoder
DualLoopPositionEncoder

This value along with the Motor Encoder value define the mechanical ratio of motor turns to position encoder turns. Typically these values will have units of encoder counts since this will give the highest degree of accuracy. The user must calculate the mechanical ratio as accurately as possible for best results.

Dual Loop Position Feedback Polarity
DualLoopPosnFdbkPolarity

Position Feedback Polarity defines the direction of the position encoder that corresponds to a positive position change. This value is somewhat arbitrary, since the position encoder can be mounted to spin CW or CCW as the material moves "forward". If this value is not set correctly in Dual Loop mode the motor will rotate in the opposite direction as desired.

Output Source
EncoderOutput.Source

This determines which signal is sent out the Sync Output connector on the drive. If Motor Encoder (default) is selected, the encoder signals from the motor that the drive is controlling will be sent out the 15-pin Analog/Sync Output connector (J5). If Drive Encoder Input is selected, then the signals coming in the drives 9-pin Sync Input connector on the drive will be sent out the Analog/Sync. Output connector. When Virtual Master is selected the virtual master signal generated by the drive will be sent out the Analog/Sync. Output connector.

Current Data Rate
Ethernet.EthernetConfiguration.DataRate

The data rate is a read-only parameter that indicates the bit rate of communications over the Ethernet Hardware. Supported Data rated include 10 Mbps Full Duplex and 100 Mbps Full Duplex. The EP-P defaults to auto detect of the network data rate.

Modbus Port Number
Ethernet.EthernetConfiguration.ModbusPortNumber

This parameter is the Modbus TCP port number. Default is 502.

Explicit Parameter - Attribute
Ethernet.EthernetExplicit.#.Attribute

The Ethernet/IP Attribute is a sub-classification for a parameter or bit. The attribute is grouped directly under the broader category of class.

Ethernet Explicit Command Complete
Ethernet.EthernetExplicit.#.CommandComplete

This Source becomes active when the explicit message process is complete. An Explicit message is complete when it is sent to another Ethernet/IP device, and the device responds.

NOTE

This parameter does not indicate a successful explicit message. The Ethernet.EthernetExplicit.#.CommandComplete bit turns "off" when the corresponding Ethernet.EthernetExplicit.#.Initiate is activated and then turns "on" when the command is complete.

Ethernet Explicit Command In Progress
Ethernet.EthernetExplicit.#.CommandInProgress

This Source activates when the corresponding Ethernet.EthernetExplicit.#.Initiate becomes active and deactivates when the corresponding Ethernet.EthernetExplicit.#.CommandComplete becomes active.

Ethernet Explicit Initiate
Ethernet.EthernetExplicit.#.Initiate

This Destination starts the explicit messaging process using the data in the corresponding explicit message.

Explicit Parameter – Instance
Ethernet.EthernetExplicit.#.Instance

The Ethernet/IP Instance is a sub-classification for a parameter or bit. The instance is grouped directly under the broader category of attribute and allows for multiple occurrences of a parameter.

For example: Class = Index, Attribute = Index Dwell. Instance1 = Index 0, Instance2 = Index 1.

Each Instance is assigned a unique number under its respective attribute.

Ethernet Explicit Message Failed
Ethernet.EthernetExplicit.#.MessageFailed

This Source becomes active when the explicit message fails to either send a valid command to another IP address, or when the explicit message is not accepted from another address. The Ethernet.EthernetExplicit.#.MessageFailed source is cleared when the corresponding Ethernet.EthernetExplicit.#.Initiate is activated.

Explicit Message - Operation
Ethernet.EthernetExplicit.#.Operation

The operation can be one of two values, "GET" or "SET". The "GET" command defines an explicit message that will be

querying another device for data. The "SET" command defines an explicit message that will be sending data to another device.

As with all Destinations in the Epsilon EP-P, Explicit.Instance.Initiate may be initiated via a program, assignment, or a network connection. When this parameter is changed from within a program or over a network connection, the following values correspond to the terms GET and SET.

GET = 0

SET = 1

Explicit Signed Data Check Box
Ethernet.EthernetExplicit.#.SignedData

The Signed Data check box is used for explicit message read commands. When data for the read command comes in, it is stored in the parameter Ethernet.EthernetExplciit.#.Value. The signed data check box determines whether this data is consumed as a 32 bit signed, or 32 bit unsigned value.

Explicit Message - Value
Ethernet.EthernetExplicit.#.Value

This parameter gives an initial value to "SET" data that will be sent from the FM-3E or FM-4E out onto the network. This parameter is also used to store the data coming in from an explicit message from a "GET" command.

Module Status
Ethernet.EthernetStatus.EthernetModuleStatus

Module Status is an Ethernet/IP protocol specific parameter that gives feedback data of the Epsilon EP-P drive. Module Status indicates one of the following conditions: Standby, Operational. Minor Flt, or Major Flt.

Standby - Module is not configured

Operational - Module operating correctly

Minor Flt - A recoverable fault occurred. This fault does not require a power reset.

Major Flt - A non-recoverable fault occurred. The module needs to have it's 24 V power cycled to reset.

Network Status
Ethernet.EthernetStatus.EthernetNetworkStatus

Network Status is an Ethernet/IP protocol specific parameter that gives feedback data on the Ethernet/IP network as it relates to the corresponding Epsilon EP-P drive. Network Status indicates one of the following conditions: No IP, No Conn, Conn est, Conn Timeout, or Dup IP.

No IP - The Epsilon EP-P does not have an IP address configured

No Conn - The Epsilon EP-P has no established connections

Conn est - The Epsilon EP-P has at least one established connection

Conn Timeout - At least one of the established connections has timed out.

Dup IP - The Epsilon EP-P has detected that it's IP address is already in use on the network.

Ethernet Status Link Status
Ethernet.EthernetStatus.LinkState

The Link State is a parameter that indicates whether or not the Ethernet 802.3 communications interface is connected to an active network. Link State as displayed on the LCD keypad indicates one of the following conditions: Active link or Inactive link.

Active Link - The Epsilon EP-P is physically connected to a network.

Inactive Link - The Epsilon EP-P is not physically attached to an operating Ethernet network.

Ethernet Status Rcv Counter
Ethernet.EthernetStatus.RcvCounter

The Receive Counter keeps track of all IP data packets received by the Epsilon EP-P drive. This counter is useful when commissioning the EP-P drive to verify incoming communications.

Ethernet Status Xmit Counter
Ethernet.EthernetStatus.XmitCounter

The Transmit Counter keeps track of all IP data packets sent from the Epsilon EP-P drive. This counter is useful when commissioning the EP-P drive to verify outgoing communications.

Commutation Hardware Fault Enable
Fault.CommutationHardwareFaultEnable

When this check box is selected or this parameter is enabled, faults occurring from the commutation tracks U, V, and W will be recognized as "E" faults in the drive. When this check box is clear, no fault will occur due to the commutation tracks. This functionality can be useful to diagnose the nature of the "E" fault. If the check box is clear and "E" faults are still occurring, then the encoder lines (A, A/, B, B/, C, C/) are the most likely source of the "E" fault.

Active Fault
Fault.#.Active

The specified fault is active. See the help index for more information on faults and recovery from them.

Clear Non-Critical Fault Counts
Fault.ClearNonCriticalFaultCounts

Set the value to TRUE to reset all of the module fault counts to zero. The read back value is whatever was last set (TRUE or FALSE). It is not necessary to set it FALSE before setting it TRUE – the act of writing TRUE initiates the operation. Another way to clear Non-Critical Fault Counts is to click on the Clear Non-Critical Fault Counts button on the Faults view/ Faults Counts Tab when online with the drive.

Fault Counts
Fault.#.Counts

The drive stores the total number of times the specific fault has occurred since it was manufactured.

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.

Faults Bitmap 1
Fault.FaultBitmap1

This parameter is a 32-bit register which holds drive fault status bits. Following is a list of drive faults and their associated bit numbers:

0 = Encoder state fault

1 = Encoder hardware fault

3 = Drive power module fault

4 = Low DC bus fault

5 = High DC bus fault

8 = ISR overrun fault

9 = Drive trajectory fault

10 = Sync fault

19 = Drive over speed fault

20 = Drive invalid configuration fault

21 = Drive power up self test fault

22 = NVM Invalid fault

24 = Drive RMS shunt power fault

25 = Motor Overtemperature fault

26 = Drive Over Temp fault

30 = Auto Tune fault

All other bits are not used. A "1" in these bit locations indicates the specific fault is active, and a "0" is inactive.

Faults Bitmap 2
Fault.FaultBitmap2

This parameter is a 32-bit register holds the drive fault status bits. Following is a list of drive faults and their associated bit numbers:

0 = Watchdog Timer fault #2

1 = Invalid Configuration fault #2

2 = NVM Invalid fault #2

3 = Power Up Test fault #2

4 = Following Error fault
5 = Travel Limit Plus fault
6 = Travel Limit Minus fault
7 = Program fault
8 = No Program Loaded fault
9 = DeviceNet Connection Timeout fault
10 = DeviceNet BusOff Interrupt fault
11 = DeviceNet Duplicate Mac ID fault
12 = Trajectory fault #2
13 = Profibus Parameterization fault
14 = Profibus Watchdog fault
15 = Profibus Configuration fault
18 = Cam fault
All other bits are not used. A "1" in these bit locations indicates the specific fault is active, and a "0" is inactive.

Faulted
Fault.Faulted

Any fault will activate this event.

Reset Faults
Fault.Reset

Resets faults that do not require a power down. This event is "or"ed with the reset button on the drive.

Fault Log - Fault Type
FaultLog.#.FaultType

This is the fault identifier for the current fault log entry. It is a read only parameter for the logged fault.
It may hold a value for any fault supported by the drive. They are the same as the faults listed on the Assignments View for Faults.

Code	Fault Description
00	EncoderStates
01	EncoderHardware
03	PowerModule
04	LowDCBus
05	HighDCBus
08	IsrOverrun
09	TrajectoryFault
0A	Synchronization
10	WatchdogTimer
13	OverSpeed
14	InvalidConfiguration
15	PowerUpSelfTest
18	RMSShuntPower
19	MotorOverTemperature
1A	OverTemperature
1E	Auto-Tune
20	WatchdogTimer
21	InvalidConfiguration
22	NVMInvalid
23	PowerUpTest
24	FollowingError
25	TraveLimitPlus
26	TraveLimitMinus
27	ProgramFault

Code	Fault Description
28	NoProgram
29	DevicenetConnTimeout
2A	DevicenetBusOffInt
2B	DevicenetDupMacId
2C	TrajectoryFault
2D	ProfibusParameterizationFlt
2E	ProfibusWatchdogFault
2F	ProfibusConfigurationFault

Fault Log - Power Up Count
FaultLog.#.PowerUpCount

This is the drive power up count at the time that the fault occurred.

Fault Log - Power Up Time
FaultLog.#.PowerUpTime

This is the drive power up time when the fault occurred.

Fault Log - Sub Fault
FaultLog.#.SubFault

The sub-fault value at the time the fault occurred.

Fault Log - Valid Entry
FaultLog.#.ValidEntry

Flag to indicate that the log entry is valid. The flag is cleared for all fault entries when the fault log is cleared.

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. When the feedhold destination is deactivated the motor will accelerate to full velocity using the value of the FeedholdDecelTime.

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.

Foldback Active
FoldbackActive

This source (output function) is active when the drive is limiting motor current. If the Foldback RMS exceeds 100 percent of the continuous rating, the current foldback circuit will limit the current delivered to the motor to 80 percent of the continuous rating.

Foldback RMS
FoldbackRMS

This read-only parameter accurately models the thermal heating and cooling of the drive. When this parameter reaches 100 percent, current foldback will be activated.

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.

Friction
Friction

This parameter is characterized in terms of the rate of friction increase per 100 motor RPM. If estimated, always use a conservative (less than or equal to actual) estimate. If the friction is completely unknown, a value of zero should be used. A typical value used here is less than one percent.

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 view. When gearing from a program, the Gear.Initiate and Gear.Stop instruction 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. 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.

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 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.

Gear Stop
Gear.Stop

Gear.Stop will stop gearing motion that has been initiated from a program using Gear.Initate.

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.

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.

Acceleration
Home.#.Accel

This parameter sets the average Acceleration rate used during the home, units are specified on the User Units page.

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.

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.

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.

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.

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.

Deceleration
Home.#.Decel

The Deceleration ramp parameter is used during all the home moves specified in user units.

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.

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.

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.

Limit Distance Hit
Home.#.LimitDistHit

This source is activated when the home sensor is not found before the Home Limit Distance is traveled.

Name
Home.#.Name

The user can specify a name up to 12 alphanumeric characters that allows a descriptive name for the 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.

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 then marker.

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.

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.

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.

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.

If entering a timed based index at a non zero velocity, the Max. Velocity must be greater than the entry velocity to prevent Index.ProfileLimited. Once Index.ProfileLimited is set, the index will be treated as a non-timed based index.

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.

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 Pro software.

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
Index.#.AnalogLimitType

Select the analog parameter (i.e. Torque Command) to compare to the AnalogLimitValue. Satisfying the comparison triggers the index registration event.

Limit Value
Index.#.AnalogLimitValue

This preset value is compared to the selected AnalogLimitType to determine the registration point.

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.

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.

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.

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.

Deceleration
Index.#.Decel

This parameter is the Average Deceleration rate used during the index. Units are specified on the User Units page.

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.

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 Units view.

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.

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.

Name
Index.#.Name

The user can specify an Index name of up to 12 characters.

Enable 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.

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.

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.

PLS Status
Index.#.PLSStatus

Controlled by the PlsOnDist and PlsOffDist 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.

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.

Registration Type
Index.#.RegistrationType

This selects either sensor or analog as the registration mark for a registration 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.

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.

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.

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.

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.

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
Inertia

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 a 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, in seconds, that commanded motion must complete and the absolute value of the following error is less than the InPosnWindow parameter 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.

In Position Window
InPosnWindow

The absolute value of the following error must be less that this value at the completion of a move for the InPosnTime before InPosn will activate.

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.

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.

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.

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.

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.

Acceleration
Jog.#.Accel

This parameter is the average acceleration ramp for the specific 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.

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.

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.

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.

Deceleration
Jog.#.Decel

This parameter is the average deceleration ramp for the specific 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.

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.

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.

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.

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.

Low Pass Filter Frequency Enable
LowPassFilterEnable

This parameter enables a low pass filter applied to the output of the velocity command before the torque compensator. The low pass filter is

only active in Pulse and Velocity modes, not in Torque mode.

Low Pass Filter Frequency
LowPassFilterFrequency

When enabled this parameter defines the low pass filter cut-off frequency. Signals exceeding this frequency will be filtered at a rate of 40 dB per decade. The default value is 600 Hz. The low pass filter is only active in Pulse and Velocity modes, not in Torque mode.

Modbus Master Baud Rate
MbMaster.CommPort.Baudrate

When the drive's serial port is operating as the Modbus master (either MbMaster.CommPort.EnableMbGateway or MbMaster.CommPort.EnableMbMaster is enabled) this parameter sets the data rate for the port.

When operating as a Modbus slave (for an HMI or PowerTools connectivity), the data rate is controlled by Modbus.Baudrate.

Enable Modbus Gateway
MbMaster.CommPort.EnableMbGateway

This parameter when enabled allows the drive to pass through Modbus messages received in the Ethernet port through to the serial port that do not match the Target Drive Address (Modbus.ModbusID) of the drive.

Enable Modbus Master
MbMaster.CommPort.EnableMaster

When this parameter is enabled the drive operates the serial port as a Modbus master. Slave devices can be read from or written to using user program - Modbus commands.

Force Slave Mode
MbMaster.CommPort.ForceSlaveMode

When this parameter is active it forces the drive to operate the serial port in Modbus slave mode. This will override both Modbus bridging mode (MbMaster.CommPort.EnableMbGateway) and Modbus master operations (MbMaster.CommPort.EnableMbMaster.

Inter-character Timeout
MbMaster.CommPort.IntercharacterTimeout

Defines a broken frame if data stream is interrupted with this duration of silence, but less than Inter-frame Timeout time. See ∆b in Figure 121 *Modbus RTU Timing Diagram* on page 108.

Inter-frame Timeout
MbMaster.CommPort.InterframeTimeout

This parameter specifies the period of no-data transmission indicating the end of a Modbus packet. See ∆c in Figure 121 *Modbus RTU Timing Diagram* on page 108.

Msg Gap
MbMaster.CommPort.MsgGap

This parameter defines the minimum amount of time to wait following a slave response before sending a new query to a different slave. See ∆d in Figure 121 *Modbus RTU Timing Diagram* on page 108.

Parity
MbMaster.CommPort.ParitySetting

When the device's serial port is operating as a Modbus master (either MbMaster.CommPort.EnableMbGateway or MbMaster.CommPort.EnableMbMaster is enabled) this value sets the data parity for the port.

When operating as a Modbus slave (for HMI or PowerTools connectivity), the data parity defaults to none.

Response Gap
MbMaster.CommPort.ResponseGap

This parameter is the maximum time to wait before the slave has timed out. See ∆a in Figure 121 *Modbus RTU Timing Diagram* on page 108.

Stop Bits
MbMaster.CommPort.StopBits

When the device's serial port is operating as a modbus master (either MbMaster.CommPort.EnableMbGateway or MbMaster.CommPort.EnableMbMaster is enabled) this parameter value sets the data stop bits for the port.

Operating as a modbus slave (for HMI or PowerTools connectivity), the data stop bits defaults to 2.

Transmission Counts
MbMaster.CommPort.TransCount

Number of transmissions requested. (writable so it can be zeroed).

Transmission Time
MbMaster.CommPort.TransTime

The duration of the last transmission from request to receive.

ErrorAllSlavesCount
MbMaster.ErrorAllSlavesCount

Indicates the number of communication errors for ALL slaves.

Error Slave Count
MbMaster.#.ErrorSlaveCount

Indicates the number of failed attempts to communicate with the slave device. This parameter can be cleared by setting it to zero in a user program.

Error Slave Detected
MbMaster.#.ErrorSlaveDetected

Indicates weather the last transmission to the slave failed.

Error Status
MbMaster.#.ErrorStatus

The status of the last command processed by the specified slave. The slave status is updated as follows:

- Cleared at start of the command
- At end of command it is set non-zero if there was any errors
- Errors less than 100 are Modbus exception responses .

Error Status Number		Error Description
01	Illegal Function	The slave does not support the request
02	Illegal Data Address	The data address is not supported by the slave
03	Illegal Data Value	The value (range) is not supported by the slave
04	Slave Device Failure	An unrecoverable error occurred while the slave was a attempting to perform the request
05	Acknowledge	The slave accepted the request but expects a
06	Busy	The slave is busy processing a previous request

Errors greater than 100 are internal errors whose meaning may vary with each firmware revision.

Modbus ID
MbMaster.#.ID

The Modbus slaves Modbus ID number.

Message Gap
MbMaster.#.MsgGap

This parameter defines the minimum amount of time to wait following a slave response before sending a new query to a different slave. See Δd in Figure 121 *Modbus RTU Timing Diagram* on page 108.

If the value of MbMaster.CommPort.MsgGap is larger, then its value will take precedence.

Modbus Master Path Type
MbMaster.#.PathType

This parameter defines the connection method to reach the slave. The user can choose between Ethernet (Modbus TCP/IP) or Serial (Modbus RTU).

When Ethernet is selected the master attempts to communicate to the slave device using Ethernet.

When selecting Serial, the master attempts to communicate to the slave using the serial port.

NOTE

The drive's serial port must be configured as a modbus master before attempting to communicate to the slave via serial communications.

Receive Count
MbMaster.#.RecvCount

Indicates the number of responses from the slave device. This parameter can be cleared by setting it to zero in a user program.

Response Gap
MbMaster.#.ResponseGap

This parameter defines the maximum amount of time to wait for a response before declaring a timeout. See Δa in Figure 121 *Modbus RTU Timing Diagram* on page 108.

If the value of MbMaster.CommPort.ResponseGap is larger, then that value will take precedence.

Transmission Count
MbMaster.#.TransCount

This parameter indicates the number of transmissions to the slave device. This parameter can be cleared by setting it to zero in a user program.

Transmission Time
MbMaster.#.TransTime

This parameter indicates (in milliseconds) the time to complete the last Modbus command.

Word Swap
MbMaster.#.WordSwap

Word Swap is used to change the order that the user will send or receive the 32-bit parameters over the network. The default Word Swap for Modbus Master is LSW-MSW.

BaudRate
Modbus.BaudRate

Modbus baudrate for the Epsilon EP-P drive serial communication.

Modbus Id
Modbus.ModbusId

Modbus ID # for the Epsilon EP-P drive.

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 axis can have a length up to 12 characters. This can be used to help differentiate setup files.

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. When set to Plus or Minus the PLS Status will activate when the position of the PLS source is > or = to the on position, + or < the off position. See PLS View on page 56 for more information on how the PLS feature works.

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.

With the PLS.#.Direction set to "Both". The PLS.#.Status will activate when traveling in the positive direction and the position feedback reaches the OnPosn. As the motor continues in the same direction, the PLS.#.Status will deactivate when the source reaches or exceeds the OffPosn. If motor travel changes to the negative direction after passing the OffPosn, 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 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 the PLS.#.Direction is set to "Both". When traveling in the positive direction and the source 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 Pro 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 position feedback 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 RotaryRolloverPosition 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 four different sources: MotorPosnFeedback, MotorPosnCommand, MasterPosnFeedback, or FreeRunTime. This parameter determines which signal the PLS uses to reference its On Position and Off Position in order to determine the PLS.#.Status parameter.

PLS Status
PLS.#.Status

This source is active when the position of the PLS source is greater than or equal to the On Position and less than the Off Position.

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 is the commanded position based on information entered into the application. This parameter does not take following error into account. See also PosnFeedback and FollowingError. Units are in user units.

Position Error Integral Time Constant
PosnErrorIntegralTimeConst

Position Error Integral parameter is a control term, which can be used in Pulse mode to compensate for the continuous torque required to hold a vertical load against gravity or to minimize following error.

The user configures this control term using the “Position Error Integral Time Constant” parameter. This parameter determines how quickly the drive will correct for in-position following error. The time constant is in milliseconds and defines how long it will take to decrease the following error to 37 percent of the original value. In certain circumstances the value actually used by the drive will be greater than the value specified here.

Min Time Constant = 1000/Response

For example, with “Response” set to 50, the minimum time constant value is 1000/50 = 20 msec.

Position Feedback
PosnFeedback

Feedback position is the actual motor position in user units. PosnCommand minus the PosnFeedback is the FollowingError

Position Feedback In Counts
PosnFeedbackInCounts

Motor encoder position in encoder counts since power up. This position reflects the feedback position of the motor and is not scaled into user units. It can be used to confirm the exact position of the motor in applications where precise positioning is required.

Position Error Integral Enable
PosnIntegralEnable

This parameter is used to enable the position error integral compensation. See also Position Error Integral Time Constant.

Power Stage Enabled
PowerStageEnabled

This source (output function) is active when the drive's power stage is enabled.

Power Up Count
PowerUpCount

Number of times the drive has been powered up since it was manufactured.

Power Up Time
PowerUpTime

Time elapsed since the drive was last powered-up. The units is hours.

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.

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.

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 on before the actual motor motion is stopped.

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.

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.

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, found on the Ramps view. 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 as the decel, to finish its 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.

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.

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.

Name
Program.#.Name

This can be a 12 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.

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.

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.

Data In
Queue.#.DataIn

Data is loaded into the queue using the Queue.#.DataIn instruction in a program. When Data In is set equal to a 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.

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 source activates when the Comparator Select parameter is equal to the sum of the data entered into the queue, plus the queue offset. Queue Exit deactivates when the 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.

Offset
Queue.#.QueueOffset

The Queue Offset is the value that is added to the Queue Data and then compared to the Comparator 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.

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 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.

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.

Read Bit
ReadBit

This instruction reads bit values (coils or inputs) from the slave into user bits.

Response
Response

The Response parameter adjusts the velocity loop bandwidth with a range of 1 to 500 Hz. In general, it affects how quickly the drive will respond to commands, load disturbances and velocity corrections. A good value to start with (the default) is 50 Hz.

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.

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.

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.

Shunt Active
ShuntActive

This source is active when the drive's internal shunt is active (conducting current).

Shunt Power RMS
ShuntPowerRMS

This parameter models the thermal heating and cooling of the drive internal shunt. This parameter indicates the percent of shunt capacity utilization and is based on the Heat Sink RMS value. When this value reaches 100 percent the drive will generate an RMS Shunt Power Fault.

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 machines 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 Activate
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 Plus Activate
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.

Start Up
StartUp

This source can be used to trigger an event to occur on startup (when the Epsilon EP-P drive 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 Epsilon EP-P drive 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 Epsilon EP-P drive 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.

Torque Command
TorqueCommand

This parameter is the torque command value before it is limited. The torque command may be limited by the motor's rating, the amplifier's rating, the Torque Limit (if the Torque Limit Enable destination is active) or Current Foldback. Units for this parameter are defined in the Torque Group on the User Units view.

Limited Torque Command
TorqueCommandLimited

This is the actual torque commanded to the motor. This value is the result after the TorqueCommand is limited by the current foldback or the TorqueLimit value (if enabled).

Torque Level
TorqueLevel

This parameter is compared to the TorqueCommand. If the absolute value of the TorqueCommand is greater than or equal to the TorqueLevel the TorqueLevelActive source is activated. This parameter is specified in Torque User Units.

Torque Level Active
TorqueLevelActive

This source is used to indicate that the absolute value of the TorqueCommand is greater than or equal to the TorqueLevel setting.

Torque Limit
TorqueLimit

This is the level to which the TorqueCommand will be limited when the TorqueLimitEnable input function is active.

Torque Limit Active
TorqueLimitActive

Active when the TorqueCommand is greater than the TorqueLimit and the TorqueLimitEnable input function is active.

Torque Limit
TorqueLimitEnable

This destination is used to enable the TorqueLimit. This is level sensitive.

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 view, or through a user program. It can be used to temporarily disable the travel limit fault capability of the Epsilon EP-P drive. When TravelLimitDisable is activated, the 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.

Decimal Places
TorqueUnits.Decimal

This parameter is the decimal point location for user torque units.

Units Name
TorqueUnits.Name

The User can specify a torque unit name of up to 12 characters. Default is % Cont.

Percent Continuous
TorqueUnits.PercentContinuousCurrent

This parameter is the denominator of torque scaling factor. This is an amount of continuous current in percent that is equal to the TorqueUnits.Torque parameter.

Torque
TorqueUnits.Torque

This parameter is the numerator of the torque scaling factor. This is an amount of torque in Torque User Units that is equivalent to one unit of PercentContinuousCurrent (denominator of scaling factor). This scaling factor is used to relate actual current or torque to user units.

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 Epsilon EP-P drive 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.

Decimal Places
VelocityUnits.Decimal

This parameter is used to select the number of decimal places used in velocity units scaling. Using a high number of decimal places will improve velocity resolution, but will also limit the maximum velocity. This parameter is selectable between 0 and 6 decimal places. The number of decimal places set in this parameter determines the number of decimal places used in all velocity parameters throughout the PowerTools Pro software.

Scaling
VelocityUnits.DistVelScale

Velocity units can be scaled to different from distance units, i.e. user distance units are inches and velocity units are feet per minute, instead of inches per minute. To do this, simply enter 12 to set 1 foot equal to 12 inches (1 velocity unit = 12 distance units). Range is 1 to 1000, integers only.

Units Name
VelocityUnits.Name

If the user wants the velocity units to have a different distance scaling than the distance units a name can be entered here up to 12 characters, e.g. user distance units are inches and velocity units are feet per minute.

Scaling
VelocityUnits.ScalingEnable

This parameter enables separate velocity and distance user units, name and scaling. If disabled, the velocity units, name and scaling will be defined by the Distance Group.

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 Pro 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.

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.

ScalingVirtualMaster.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.

$$\text{Scaling} = \frac{\text{Virtual MasterCharacteristicLength}}{\text{VirtualMaster.CharacteristicDistance}}$$

Command Complete
VirtualMaster.CommandComplete

This parameter indicating when a Virtual Master index or jog is complete.

Command In Progress
VirtualMaster.CommandInProgress

This parameter is an output event indicating a Virtual Master index or jog is currently being executed.

Counts
VirtualMaster.Counts

This read only parameter is the number of virtual counts transmitted. It is zeroed on power up only.

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.

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 Deactivate
VirtualMaster.FeedRateDeactivate

When this parameter is activated, from an assignment or a program, virtual master feedrate override is deactivated meaning that the virtual master counts will be 100%.

FeedRate Decel/Accel
VituralMaster.FeedRateDecelerationTime

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 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.

Initiate
VirtualMaster.Initiate

This initiates a Virtual Master index that generates virtual master pulses equivalent to VirtualMaster.Dist. The pulse rate will be defined by the accel rate, velocity and decel rate. VirtualMaster.Initiate can be assigned or activated by a user program.

Marker Count
VirtualMaster.MarkerCount

The encoder marker pulses are simulated and this parameter (VirtualMaster.MarkerCount) specifies the number of VirtualMaster.Counts per encoder marker pulse generated.

Minus Activate
VirtualMaster.MinusActivate

When active, this executes a Virtual Master jog in the minus direction that generates virtual master pulses as long as the VirtualMaster.MinusActive is set. The pulse rate will be defined by the accel rate, velocity and decel rate. VirtualMaster.MinusActivate can be assigned or activated by a user program

Plus activate
VirtualMaster.PlusActivate

When this parameter is active, this executes a Virtual Master jog in the plus direction that generates virtual master pulses as long as the VirtualMaster.PlusActive is set. The pulse rate will be defined by the accel rate, velocity and decel rate. VirtualMaster.PlusActivate can be assigned or activated by a user program.

Position Count
VirtualMaster.PosnCmdInCounts

This parameter is the number of virtual counts transmitted. Write to this parameter to zero it's 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.

Posn Command
VirtualMaster.Stop

This parameter when activated stops the Virtual Master using the VirtualMaster.Decel rate.

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).

6 Tuning Procedures

The drive uses closed loop controllers to control the position and velocity of the attached motor. These position and velocity controllers and the associated tuning parameters are in effect when the drive is in velocity or pulse mode and have no effect when the drive is in Torque mode.

Many closed loop controllers require tuning using individual user-specified proportional, integral and derivative (PID) gains which require skilled “tweaking” to optimize. The combination of these gains along with the drive gain, motor gain, and motor inertia, define the system bandwidth. The overall system bandwidth is usually unknown at the end of the tweaking process. The drive closes the control loops for the user using a state-space pole placement technique. Using this method, the drive’s position control can be simply and accurately tuned. The overall system’s bandwidth can be defined by a single user-specified value (Response).

The drive’s default settings are designed to work in applications with up to a 10:1 load to motor inertia mismatch. Most applications can operate with this default setting.

Some applications may have performance requirements which are not attainable with the factory settings. For these applications a set of measurable parameters can be specified which will set up the internal control functions to optimize the drive performance. The parameters include Inertia Ratio, Friction, Response and Line Voltage. All the values needed for optimization are “real world” values that can be determined by calculation or some method of dynamic measurement.

6.1 PID vs. State-Space

The power of the state-space control algorithm is that there is no guessing and no “fine tuning” as needed with user-specified PID methods. PID methods work well in controlled situations but tend to be difficult to setup in applications where all the effects of the system are not compensated for in the PID loop. The results are that the system response is compromised to avoid instability.

The drive state-space control algorithm uses a number of internally calculated gains that represent the wide variety of effects present in a servo system. This method gives a more accurate representation of the system and maximizes the performance by minimizing the compromises.

You need only to setup the system and enter three parameters to describe the load and the application needs. Once the entries are made the tuning is complete - no guessing and no “tweaking”. The drive uses these entries plus motor and amplifier information to setup the internal digital gain values. These values are used in the control loops to accurately set up a stable, repeatable and highly responsive system.

6.2 Motor Tuning and Motor Auto-Tuning

The unloaded motor parameters are set by the motor .ddf file or by using the motor Auto-tuning. This is a separate process that is separate from this system tuning. The motor tuning should be set prior to tuning the system.

6.3 Tuning Procedure

Once the initial setup has been completed, you can run the system to determine if the level of tuning is adequate for the application. A drive can be tuned basically to four levels.

- No Tuning
- Basic Level
- Intermediate Level
- Fully Optimized Level

Each level is slightly more involved than the previous one requiring you to enter more information. If your system needs optimization, we recommend that you start with the Basic Level, then determine if further tuning is needed based on axis performance.

The setup procedures explained here assume that you are using PowerTools Pro software.

6.4 Initial Settings

Set the drive tuning parameters as follows:

- Inertia Ratio = 0
- Friction = 0
- Response = 50
- Feedforwards = Disabled

6.5 Tuning Steps

If your Inertia Ratio is greater than 10 times the motor inertia go directly to the Intermediate Level tuning.

6.5.1 No Tuning

No tuning will be required in most applications where the load inertia is 10 times the motor inertia or less.

6.5.2 Basic Level

Adjust Response to obtain the best performance.

6.5.3 Intermediate Level

1. Calculate or estimate the load inertia. It is always better to estimate low.
2. Disable the drive.
3. Enter the inertia value calculated into the Inertia Ratio parameter.
4. Leave all other tuning parameters at the initial values.
5. Enable the drive and run the system.
6. Adjust Response to obtain the best performance.

6.5.4 Fully Optimized Level

1. Determine the actual system parameters.
2. Disable the drive.
3. Enter the parameters.
4. Enable the drive and run the system.
5. Adjust Response to obtain the best performance.

General Tuning Hints

General Tuning Procedure:

1. Calculate inertia of the system

The inertia of the system up to the motor shaft should be calculated using CT-Size software or some other inertia calculating software. Under perfect mechanical conditions, entering this number into the “Inertia” parameter will produce a well-matched system tuning. Because most systems include mechanics that are less than ideal, a number less than the inertia parameter will need to be used to avoid bandwidth issues or “buzzing” of the motor. See Figures 144 and 145.

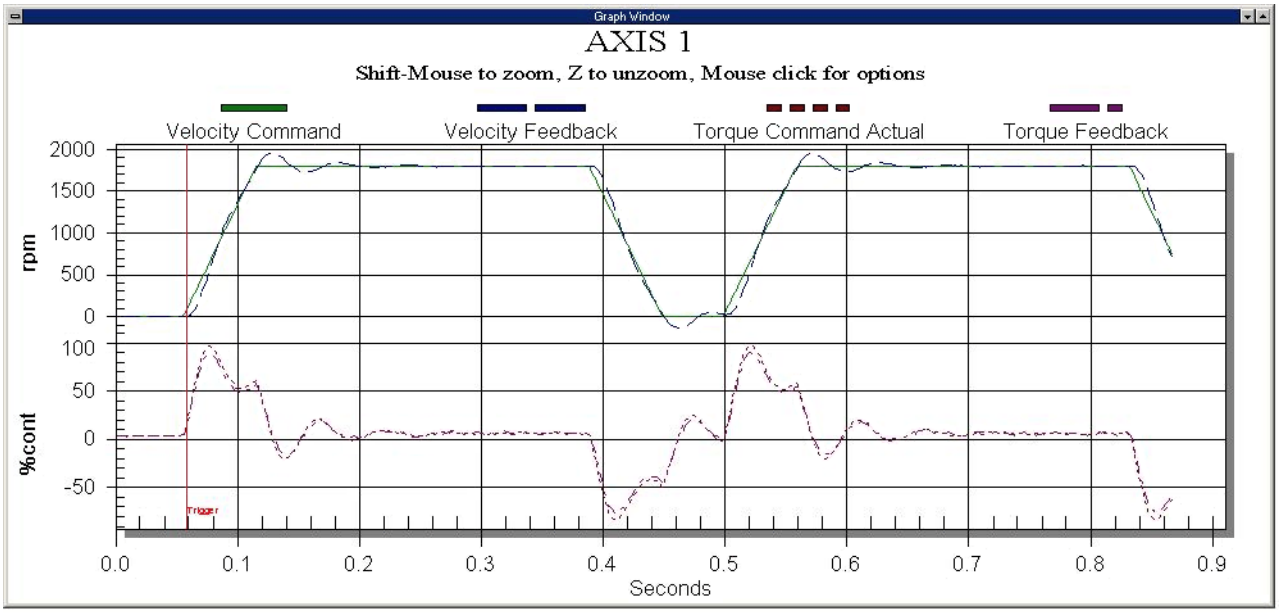


Figure 144: Default Inertia Setting (0)

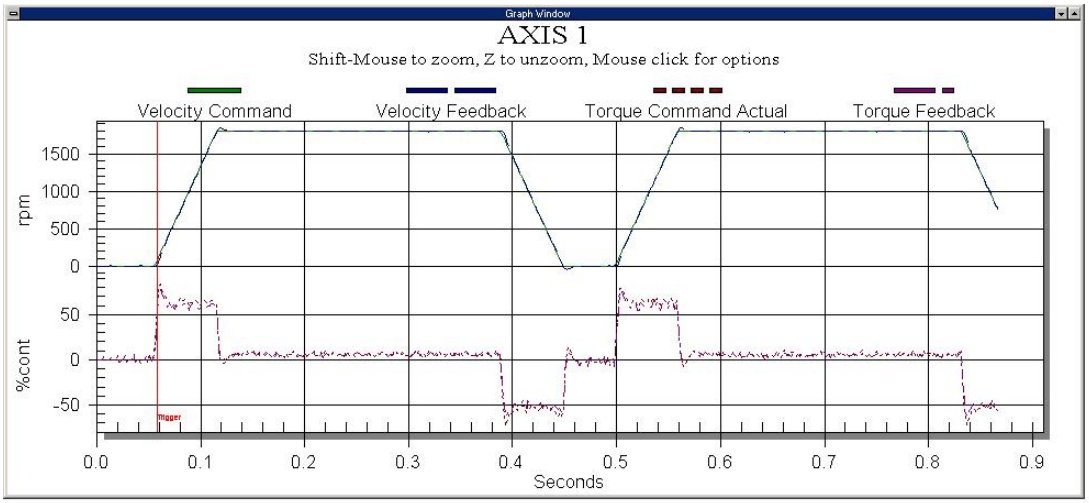


Figure 145: Inertia Setting (5)

2. Increase the response parameter

The Response is normally the final adjustment when tuning. For best performance the Response should be lower with a higher inertia mismatch (>10:1) and higher with a lower inertia mismatch.

If your system has some torsional compliance, such as with a jaw type coupling with a rubber spider, or if there is a long drive shaft, the Response should be decreased. The highest recommended Response with High Performance Gains enabled is 100 Hz.

Also, enabling the Low Pass Filter helps diminish the resonant frequency of torsionally compliant loads. In such cases, using the Low Pass Filter usually allows for higher Response values. The optimum Low Pass Filter frequency is at the frequency of the resonance.

Feedforwards can be enabled if the performance requirements are very demanding. However when using them, make sure the Inertia Ratio and Friction values are an accurate representation of the load. Otherwise, the system performance will actually be degraded or stability will suffer. Enabling the Feedforward makes the system less tolerant of inertia or friction variations during operation.

6.6 Tuning Parameters

Inertia Ratio

Inertia Ratio specifies the load to motor inertia ratio and has a range of 0.0 to 50.0. A value of 1.0 specifies that load inertia equals the motor inertia (1:1 load to motor inertia). The drives can control up to a 10:1 inertia mismatch with the default Inertia Ratio value of 0.0. Inertial mismatches of over 50:1 are possible if response is reduced.

The Inertia Ratio value is used to set the internal gains in the velocity and position loops, including feedforward compensation if enabled.

To calculate the Inertia Ratio value, divide the load inertia reflected to the motor by the motor inertia of the motor. Include the motor brake as a load where applicable. The resulting value should be entered as the Inertia Ratio parameter.

$$IR = \frac{RLI}{MRI}$$

Where:

IR = Inertia Ratio

RLI = Reflected Load Inertia (lb-in-sec^2)

MI = Motor Inertia (lb-in- sec^2)

If the exact inertia is unknown, a conservative approximate value should be used. If you enter an inertia value higher than the actual inertia, the resultant motor response will tend to be more oscillatory.

If you enter an inertia value lower than the actual inertia, but is between 10 and 90 percent of the actual, the drive will tend to be more sluggish than optimum but will usually operate satisfactorily. If the value you enter is less than 10 percent of the actual inertia, the drive will have a low frequency oscillation at speed.

Friction

In the drive, this is a viscous friction parameter, characterized in terms of the rate of friction increase per 100 motor RPM. The range is 0.00 to 100.00 in units of percent continuous torque of the specified motor/drive combination. The Friction value can either be estimated or measured.

If estimated, always use a conservative (less than or equal to actual) estimate. If the friction is completely unknown, a value of zero should be used. A typical value used here would be less than one percent.

If the value entered is higher than the actual, system oscillation is likely. If the value entered is lower than the actual a more sluggish response is likely but generally results in good operation.

Response

The Response adjusts the velocity and position loop bandwidths with a range of 1 to 500 Hz. In general, it affects how quickly the drive will respond to commands, load disturbances and velocity corrections. The Response bandwidth is set to the Response value. It reflects both the velocity command and the load disturbance correction bandwidth. Increasing the Response value will increase loop stiffness. The maximum Response level recommended is approximately 100 Hz.

If the Inertia Ratio and Friction values are exactly correct, changing the Response will not affect the damping (percent of overshoot and number of ringout cycles) to velocity command changes or load disturbance corrections but will affect their cycle frequency. The response level generally should be decreased as the load to motor inertia ratio increases.

Feedforwards

Feedforward gains are essentially open loop gains that generate torque commands based on the commanded velocity, accel/decel and the known load parameters (Inertia Ratio and Friction). Using the feedforwards reduces velocity error during steady state and reduces overshoot during ramping. This is because the Feedforwards do not wait for error to build up to generate current commands.

Feedforwards should be disabled unless the absolute maximum performance is required from the system. Using them reduces the forgiveness of the servo loop and can create instability if the actual inertia and/or friction of the machine varies greatly during operation or if the Inertia Ratio or Friction parameters are not correct.

The internal feedforward velocity and acceleration gains are calculated by using the Inertia Ratio and Friction parameters. The feedforward acceleration gain is calculated from the Inertia Ratio parameter and the feedforward velocity gain is calculated from the Friction parameter.

When Feedforwards are enabled, the accuracy of the Inertia Ratio and Friction parameters is very important. If the Inertia Ratio parameter is larger than the actual inertia, the result would be a significant velocity overshoot during ramping. If the Inertia parameter is smaller than the actual inertia, velocity error during ramping will be reduced but not eliminated. If the Friction parameter is greater than the actual friction, it may result in velocity error or instability. If the Friction parameter is less than the actual friction, velocity error will be reduced, but not eliminated.

Low Pass Filter Group

The Low Pass Filter will reduce machine resonance due to mechanical coupling and other flexible drive/load components by filtering the command generated by the velocity loop. A check box on the Tuning view enables a low pass filter applied to the output of the velocity command before the torque compensator. The low pass filter frequency parameter defines the low pass filter cut-off frequency. Signals exceeding this frequency will be filtered at a rate of 40 dB per decade. The default value is 600Hz.

6.7 Determining Tuning Parameter Values

For optimum performance you will need to enter the actual system parameters into the drive. This section discusses the methods which will most accurately determine those parameters.

NOTE

If you have an application which exerts a constant unidirectional loading throughout the travel such as in a vertical axis, the inertia tests must be performed in both directions to cancel out the unidirectional loading effect.

Initial Test Settings

When running the tests outlined in this section, the motor and drive must be operational so you will need to enter starting values.

If your application has less than a 10:1 inertia mismatch, the default parameter settings will be acceptable. If the inertial mismatch is greater than 10:1, use the following table for initial parameter settings.

Parameter	Setting
Friction	0.00
Inertia Ratio	1/3 to 1/2 Actual
Response	500/Inertia Ratio
Feedforwards	Enabled

Determining Friction

This parameter represents friction that increases proportionally as motor velocity increases. The viscous friction of your system can be determined by reading the percent of continuous torque required to operate the loaded motor at two different speeds.

Consider the following before determining the Friction:

- The most consistent readings can usually be obtained at motor speeds higher than 500 RPM but lower test speeds can be used if necessary.
- If your application has travel limits, it may be helpful to use an external position controller to prevent the axis from exceeding the machine limits. Set up a trapezoidal profile as shown.
- In the procedure below, the Torque Command and Velocity Feedback parameters can be measured using the drive's analog outputs, PowerTools software.
- With vertical loads the test readings must be taken while traveling in the same direction.
- The use of the PowerTools graph feature or an oscilloscope may be needed for systems with limited travel moves to measure the rapidly changing torque and velocity signals.
- If your system's friction changes with operating temperature, perform this procedure at normal operating temperature.

Procedure for Determining Friction:

1. Run the motor at the low test speed (at least 500 RPM).
2. While at speed, note the Torque Command Actual value (TCL).

NOTE

If the friction loading of your system varies when operating at constant speed, due to a load or spring load that changes as the motor rotates, use the lowest value measured.

3. Repeat Step 1 using a velocity at least two times the low speed.
4. While at speed, note the Torque Command Actual value (TCH).
5. Use the following formula to calculate the friction:

$$FV = (100)\frac{(T_{CH} - T_{CL})}{RPM_H - RPM_L}$$

Where:

T_{CH} = Torque Command Limited value at higher speed
T_{CL} = Torque Command Limited at lower speed
RPM_H = Higher RPM (velocity)
RPM_L = Lower RPM (velocity)
FV = Friction value

The figure below shows the relationship of Torque Command to the Velocity Feedback. There is increased torque during the Accel ramp (Ta), constant torque (Tc) during the constant velocity portion of the ramp and decreased torque (Td) during the decel ramp.

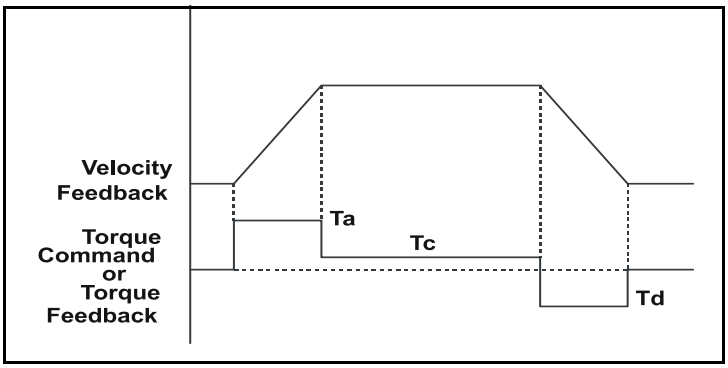


Figure 146: Trapezoidal Velocity Waveform with Torque Waveform

Determining Inertia Ratio

Actual system Inertia Ratio is determined by accelerating and decelerating the load with a known ramp while measuring the torque required.

Consider the following before determining the inertia:

- The accel and decel ramp should be aggressive enough to require at least 20 percent of continuous motor torque. The higher the torque used during the ramp, the more accurate the final result will be.
- With ramps that take less than 1/2 second to accelerate, use PowerTools graph to measure the Torque Feedback.
- With ramps that take 1/2 second or longer to accelerate, read the Torque Command parameter on the Torque view, while online, or with the Watch Window.
- To best determine the inertia, use both acceleration and deceleration torque values. The difference allows you to drop the constant friction out of the final calculation.
- If your application exerts a constant “unidirectional loading” throughout the travel such as in a vertical axis, the inertia test profiles must be performed in both directions to cancel out the unidirectional loading effect.
- The Torque Command Limited and Velocity Feedback parameters can be measured using the drive’s Analog Outputs or PowerTools Pro graph.

These will be needed for systems with limited travel moves and rapidly changing signals of torque and velocity.

Inertia Measurement Procedure:

NOTE

The test profile will need to be run a number of times in order to get a good sample of data.

1. Enable the drives and run the test profiles.
2. Note the Torque Command Limited value during acceleration and deceleration.
3. Use the appropriate formula below to calculate the inertia.

For horizontal loads or counterbalanced vertical loads use the following formula:

$$IR = \frac{(R \bullet Vm\,(Ta + Td))}{2000} - 1$$

Where:

IR = Inertia Ratio
R = ramp in ms/kRPM
Ta = (unsigned) percent continuous torque required during acceleration ramping (0 - 300)
Td = (unsigned) percent continuous torque required during deceleration ramping (0 - 300)
Vm = MotorKt * System Continuous Current / Motor Inertia in Kg-cm^2

For un-counter balanced vertical loads use the following formula:

$$IR = \frac{(R \bullet Vm\,(Tau + Tud + Tad + Tdd))}{4000} - 1$$

Where:

IR = Inertia Ratio
R = ramp in ms/kRPM
Vm = MotorKt * System Continuous Current / Motor Inertia in Kg-cm^2
Tau = (unsigned) percent continuous torque required during acceleration ramping while moving up (against the constant force)
Tdu = (unsigned) percent continuous torque required during deceleration ramping while moving up (against the constant force)
Tad = (unsigned) percent continuous torque required during acceleration ramping while moving down (aided by the constant force)
Tdd = (unsigned) percent continuous torque required during deceleration ramping while moving down (aided by the constant force)

7 Diagnostics and Troubleshooting

7.1 Diagnostic Display

The diagnostic segment display on the front of the Epsilon EP-P drive shows drive status and fault codes. When a fault condition occurs, the drive will display the fault code, overriding the status code.

The decimal point is “On” when the drives are enabled, and the stop input is not active. This indicates that the drives are ready to run and any motion command will cause motion. Motion commands will not cause motion unless you are Ready (R) and the decimal point is “On”.

7.1.1 Status Codes

Display Indication	Status	Description
R	Ready	The system is functioning normally and is ready to execute a motion command.
b	Brake Engaged (Output "Off")	Motor brake is mechanically engaged. This character will only appear if the Brake output function is assigned to an output line. See Brake Operation section for detailed description of Brake Output function.
d	Disabled	Power Stage is disabled.
C	RMS Foldback	Motor torque is limited to 80 percent.
c	Stall Foldback	Drive output current is limited to 80 percent of drive stall current.
.	Ready to Run	Drive enabled. No Stop input.
h	Homing	Home cycle is executing. Other motion commands do not funtion.
X	Indexing	Index is executing. Other motion commands do not function.
\	Decelerating from Stop or Travel Limit Decel	Deceleration ramp after the Stop or Travel Limit function is activated. The ramp is displayed while decelerating, and then display will go back to normal after completing the decel ramp.
J	Jogging	Jog function in progress. Other motion commands do not function.
K	Camming	Camming function in progress. Other motion commands do not function.

Display Indication	Status	Description
P	Program	Program is executing.
P	Pulse Mode	Pulse Mode Operation is executing.
T	Torque Mode	Analog Torque Mode Operation
V	Velocity Mode	Analog Velocity Mode Operation

7.1.2 Fault Codes

A number of diagnostic and fault detection circuits are incorporated to protect the drive. Some faults, such as High DC Bus and Motor Over Temperature can be reset with the Reset button on the front of the drive or the Reset input function. Other faults, such as Encoder Faults, can only be reset by cycling power “Off” (wait until the diagnostics display turns “Off”), then power “On”.

The drive accurately tracks motor position during fault conditions. For example, if there is a Low DC Bus fault where the power stage is disabled, the drive will continue to track the motor’s position provided the logic power is not interrupted.

The +/- Travel Limit faults are automatically cleared when the fault condition is removed.

Segment Display	Fault	Action to Reset	Bridge Disabled
X	Flash Invalid	Reprogram the Flash	Yes
T	Drive Power Up Test	Cycle Logic Power	Yes
N	Drive NVM Invalid	Reset Button or Input Line	Yes
U	Invalid Configuration	Reset Button or Input Line	Yes
Z	Power Module	Reset Button or Input Line	Yes
H	High DC Bus	Reset Button or Input Line	Yes
L	Low DC Bus	Reset Button or Input Line	Yes

Segment Display	Fault	Action to Reset	Bridge Disabled
P	Encoder State	Cycle Power	Yes
E	Encoder Hardware	Cycle Power	Yes
M	Motor Overtemp	Reset Button or Input Line	Yes
S	RMS Shunt Power	Reset Button or Input Line	Yes
O	Overspeed	Reset Button or Input Line	Yes
F	Max Following Error	Reset Button or Input Line	Yes
L	Travel Limit +/-	Auto	No
W	All "On"	Normally on for one second during power up	Yes
U	Run Time Faults	Reset Button or Input Line	Yes
A	Drive Overtemp	Allow the drive to cool down, Cycle Logic Power	Yes

Fault Descriptions

Flash Invalid

This fault indicates that the firmware checksum has failed. Use the Tools Program Flash menu item from PowerTools to reprogram/upgrade the firmware stored in flash memory. If this problem persists, call Control Techniques. A common cause would be an interrupted F/W Flash upgrade (cable disconnected during an upgrade process).

Power Up Test

This fault indicates that the power-up self-test has failed. This fault cannot be reset with the reset command or reset button.

NVM Invalid

At power-up the drive tests the integrity of the non-volatile memory. This fault is generated if the contents of the non-volatile memory are invalid.

7 Power Module

This fault is generated when a power stage over-temperature, over-current or loss of power stage logic supply occurs. This can be the result of a motor short to ground, a short in the motor windings, a motor cable short or the failure of a switching transistor.

H High DC Bus

This fault will occur whenever the voltage on the DC bus exceeds the High DC Bus threshold. The most likely cause of this fault would be an open external shunt, a high AC line condition, or an application that requires an external shunt (e.g., a large load with rapid deceleration) but none is installed.

	High DC Bus Threshold
Epsilon EP	415 Vdc

L Low DC Bus

This fault will occur whenever the voltage on the DC bus drops below the Low DC Bus threshold. The most likely cause of this fault is a reduction (or loss) of AC power. A 50 ms debounce time is used with this fault to avoid faults caused by intermittent power disruption. With an Epsilon EP drive, the low DC bus monitoring can be disabled. In an EP-B and EP-I this fault is disabled by clearing the check box on the Faults view, and for an EP-P the check box is located on the Advanced view

	Low DC Bus Threshold
Epsilon EP	60 Vdc

Special note for EP209 and EP216 drives, the Low DC Bus fault may not be disabled. The bus voltage must reach 140 Vdc (100 Vac input) before this fault will reset and it will reset automatically. It will occur again when the bus voltage drops by 50 V from the voltage on the bus when soft-start mode ends and the drive is ready to run.

E Encoder State

Certain encoder state transitions are invalid and will cause the drive to report an encoder state fault. This is usually the result of noisy encoder feedback caused by poor shielding. For some types of custom motors it may be necessary to disable this fault. With an Epsilon EP-P drive, the Encoder State monitoring can be disabled by clearing the check box on the Advanced view.

E Encoder Hardware

If any pair of complementary encoder lines (A, B, Z) are in the same state, an encoder line fault is generated. Also, can be generated if all three commutation channels (U, V, W) are 0 or 1, an illegal state. The most likely cause is a missing or bad encoder connection.

M Motor Overtemp

This fault is generated when the motor thermal switch is open due to motor over-temperature or incorrect wiring.

S RMS Shunt Power

This fault is generated when RMS shunt power dissipation is greater than the design rating of the internal shunt.

O Overspeed

This fault occurs in one of two circumstances:

1. When the actual motor speed exceeds the Overspeed Velocity Limit parameter or 150% of motor maximum operating speed. This parameter can be accessed with PowerTools Pro software.
2. If the combination of command pulse frequency and Pulse Ratio can generate a motor command speed in excess of the fixed limit of 13000 RPM, an Overspeed Fault will be activated. In Pulse mode operation and any Summation mode which uses Pulse mode, the input pulse command frequency is monitored and this calculation is made. For example, with a Pulse Ratio of 10 pulses per motor revolution, the first pulse received will cause an Overspeed fault even before there is any motor motion.

H Drive Over Temp

3. Indicates the drive internal temperature has reached an over temperature condition - not currently implemented in the EP202, EP204 or EP206 models. For Epsilon EP209 and EP216 drive models, this fault will occur if the soft-start circuit has failed or mis-wiring of the bus or shunt prevents proper start-up, resulting in overheating the soft-start resistor internal to the drive. With the EP216 model this over temperature condition could be the result of fan failure.

F Following Error

This fault is generated when the following error exceeds the following error limit. With PowerTools Pro you can change the Following Error Limit value, enable or disable it in the Position view.

Travel Limit +/-

This fault is caused when either the + or - Travel Limit input function is active.

Sync Fault

This fault occurs when the user selected trajectory update rate is set too short based on the processor requirements. The three possible trajectory update rates are 800µs, 1200 µs, or 1600 µs. Try changing the trajectory update rate to the largest value (1600 µs) and run the application again. If the problem persists after setting to 1600 µs, contact Control Techniques technical support.

Run Time Faults

A 4 will be displayed on the diagnostic display when the Epsilon EP-P drive experiences one of the following faults.

Trajectory Fault #1

This fault occurs when the drive commands motion that cannot be achieved due to excessive following error, accel, decel, velocity settings, or unusable user units. Check the user units, velocities, accels and decels for correct values.

Trajectory Fault #2

This fault occurs when using the "Using Capture.#" option in a user program. If the capture has never been triggered, or the capture data has gone "stale", the drive will not be able to process motion properly.

Program Fault

This fault indicates a problem was encountered in a user program. For example: an illegal math operation resulting in a divide by zero or overflow of 32-bit data. This error can also occur if trying to access a drive parameter that is non-existent or not available to the user.

Invalid Configuration Fault #2

The user program in flash memory will not run. Download the user program again using PowerTools Pro. A common cause of this fault could be an interrupted configuration download, such as a cable being disconnected during the download.

No Program

This fault will be displayed on initial power-up indicating that no configuration has been downloaded to the drive. To clear the fault, download a valid configuration to the drive.

All "On"

This is a normal condition during power up of the drive. It will last for less than 1 second. If this display persists, call Control Techniques for service advice.

7.2 Drive Faults

The Active Drive Faults dialog box is automatically displayed whenever a fault occurs. There are two options in this dialog box: Reset Faults and Ignore Faults.

Resetting Faults

Some drive faults are automatically reset when the fault condition is cleared. Others require drive power to be cycled or the drive to be “rebooted” to be cleared. If you wish to continue working in the PowerTools Pro software without resetting the fault, click the Ignore Fault button.

To reset faults that can be reset with the Reset Faults button, simply click the Reset Faults button in the Drive Faults Detected dialog box or push the Reset button on the front of the drive where the fault occurred.

Viewing Active Drive Faults

To view all active drive faults, select the View Faults command from the Device menu. The dialog box displayed is the same as Active Drive Faults dialog box described above.

Rebooting the Drive

To reboot the drive, cycle power or select Reboot Drive from the Device menu. This command reboots the drive attached to the active Configuration Window.

7.3 Error Messages

PowerTools Pro will pop-up an error message box to alert the user to any errors it encounters. These message boxes will describe the error and offer a possible solution.

The terms below appear in a list of common problems you might encounter when working with PowerTools Pro software along with the error message displayed, the most likely cause and solution.

Assign means to set a value using an equation. For example, x = 2, you are assigning the value of 2 to x.

A **Boolean value** is a value that represents two states such as On or Off. In the Epsilon EP-P drive there are three variable types that have Boolean values. They are Boolean Variables, Input Event Variables and Output Event Variables. They all have a Boolean Value and can be used in equations to assign their Boolean value to another variable or in a conditional test.

In an equation, conditional tests such as (vel > 3.1) become Boolean values.

In Epsilon EP-P drive, unquoted text names are used to represent Boolean constants. Several different names are available.

They all represent the two Boolean states and therefore are interchangeable.

An **Expression** is a collection of mathematical operands (variables, constants and numbers) and operators(+, -, *, <, >, etc.) that form a value. The right hand side of an equation (to the right of the =) is an expression.

The **Parser** is an internal component of PowerTools Pro software that reads your program text file and generates executable code used by the Epsilon EP-P drive firmware. The parser detects errors that are reported to you as Red Dot Error Messages.

Program errors are displayed in the program view in Red Dot Error Messages. They are indicated with red dots. To get further information on the cause of the error, use the program toolbar RedDot Help button. This is an on/off setting that enables error message displays and application help messages.

7.3.1 Non-Programming Error Messages

These messages occur while you are working in a view other than the Program view. The Program view has error messages specific to it, and they are described in a Programming Error Messages section. The popup messages are listed below.

Can Not Add Index, until current is valid.,
Can Not Add Jog, until current is valid.,
Can Not Add Program, until current is valid.

The current view must be valid before you can create a new instance of Index, Jog or Home.

Error: The maximum limit of Instances is reached.

The number of Index instances, Jog Instances and Home Instances is limited. If you attempt to add an instance and the number of existing instances is at the maximum, you will get this message.

7.3.2 Programming Error Messages

These Red Dot Error messages occur while you are working in the Program view.

When creating a program, the parser is executed when you left mouse click, when you arrow off the current Line, when you enter the carriage return, when you paste or when you drop a drag source. The parser detects errors and marks the line with a “Red Dot”. To get further information on the cause of the error, use the program toolbar Red Dot Help button. This is an on/off setting that enables error message displays and application help messages.

Problem/Message	Cause	Solution
Your Application is not valid to download ...	There are errors such as “Red Dot” errors in one or more programs that prevent the program from being downloaded to the drive.	The message will provide more information such as which program is invalid to help the user correct the problem.
Your Application has ...	There are errors such as “Red Dot” errors in one or more programs. The operation (i.e. file save) was completed, however other operations such as download would fail for this application.	The message will provide more information such as which program is invalid to help the user correct the problem.
A Epsilon EP-P drive number’s decimal Point resolution can not be greater than ten	The Epsilon EP-P drive does not use standard floating point. It uses Integer arithmetic to prevent round off errors. Decimals are used, but decimal point position is handled separately from the integer value. Zero puts the decimal point to the far right. Ten puts the decimal point at the far left.	The decimal point position must be between zero and ten.
A Epsilon EP-P drive number’s mantissa must be between -2147483647 and 2147483647	The mantissa must be between -2147483648 and 214748364.	The decimal point position must be between zero and ten.
A numeric variable can only be assigned a numeric value	The Variable is a numeric. It only accepts types consisting of numeric values.	
A string variable can only be assigned a quoted text string	The Variable is string. It only accepts types consisting of text strings.	
Can only compare(>,<,etc) numeric results	This message occurs in conditional Expressions (i.e. If then). Variables are type identified, so equation and assignments (x = 9) can be verified. In an expression only numerical values can be compared for greater than and less than conditions.	
Could not find the variable	See message.	

Problem/Message	Cause	Solution
Could not find the variable defined by program text	See message.	
Destination Event variables can only be assigned an Event or Boolean	The Variable is an Input event. It only accepts types consisting of Boolean, and events.	
Program Instance does not Exist	You attempted the “Call Program.#”, but the program does not exist.	
Single value expressions can only be Boolean constants, Events or Boolean variables	This message occurs in conditional Expressions (i.e. If then). Variables are type identified, so equation and assignments .(x = 9) can be verified.	In an expression you can use single variables without a comparison, but , then they must be a Boolean constants, Events or Boolean variable.
String does not represent a predefined name...	The string needs to match one of the defined strings on record in the Epsilon EP-P drive Registry data base.	
String is not a selection	The string matches one of the defined strings, but that string is not a selection.	
Syntax error encountered	Parser Error Message. The Parser can not understand your text sequence.	
Text Strings are limited to 12 characters...	To change a Name you assign a quoted text string to that name. In Epsilon EP-P drive, text strings are fixed at 12 characters. If you use fewer than 12 characters, blanks are automatically inserted. An error occurs if you attempt to use more than 12 characters.	
The Boolean variables can only be assigned an Event or Boolean value	The Variable only accepts types consisting of Boolean, and events.	
The destination variable does not accept Data	The Variable’s internal data type attribute was not found.	
The destination variable does not accept negative numbers	You attempted to assign a negative number to an unassigned variable .	
The destination variable is Read Only	This message occurs when trying to assign a value to a read only variable.	
The destination variable only accepts a numeric value	See message.	
The destination variable only accepts a Boolean or Event value	See message.	
The destination variable only accepts quoted "text"	See message.	
The destination variable only accepts selection text	See message.	
The destination variable's resolution is less than the resolution of the number	You attempted to assign a number with a greater resolution of decimal points than the variable will accept (i.e. index.0.vel = 2.34567).	The User Units setup will allow you to define the desired decimal point resolution.
The number is outside the range of the destination variable	You attempted to assign a number that is outside the variable’s range.	To determine the range comment out this instruction and use the red dot help on the variable.
The mix of variable or expressions types can not be added or subtracted	This message occurs in equations. Variables are type identified, so equation and assignments (x = 9) can be verified. In an equation only numerical values can be multiplied or divided. Booleans, Selections, text and events can not be added.	

Problem/Message	Cause	Solution
The mix of variable or expressions types can not be compared(=)	This message occurs in conditional Expressions (i.e. If then). Variables are type identified, so equation and assignments (x = 9) can be verified. In an expression numerical, Boolean and event values can be compared for equality conditions. Selections and text can not be used.	
The mix of variable or expression types can not be multiplied or divided	This message occurs in equations. Variables are type identified, so equation and assignments (x = 9) can be verified. In an equation only numerical values can be multiplied or divided. Booleans, Selections, text and events can not be compared.	
The Source Event-<variable name> can only be assigned <max number> times	The limits to Source Event assignments are the number of destinations assigned to a single Output event is limited. Generally this is three. For Selections it is one. The Waitfor Instruction temporarily assigns its Output Event Operands. This is subject to the assignment limitations.	To use an Output Event in a Waitfor instruction, there must be at least one free assignment.
The Selection variable can only be assigned a Selection value	The destination variable only accepts selection values. Selection values are fixed unquoted text. The selection text must exactly match the available selections of the Destination variable.	
This instance does not exist	This variable is referencing an instance that has not been created in your application. For example “Index.9.vel” the instance 9 of index has not been defined.	
This is not a fully qualified variable	To use a variable it must be fully defined. Some variables are global variables and only the name is defined. Other Variables require a name.name convention. Other variables require an instance number (index.1.vel). All the components identifying the variable must be available to qualify the variable.	
This variable cannot be assigned a selection value	The variable that you are attempting to define with a selection does not accept selection values.	
This variable type cannot be assigned a value	The Variable is of a type that does not accept any assignments. A Source Event variable is an example. You can not assign a value to an output event.	
The selection is not valid for this variable	The variable that you are attempting to define with a selection does accept selection values. However the selection you are attempting to use is not accepted by this variable.	
Trying to assign a selection variable with bad selection data	When checking to see if the selection goes with the destination variable, the source is not a defined selection for the destination variable.	

7.4 Online Status Indicators

7.4.1 Watch Window

PowerTools Pro contains a diagnostic utility called the Watch Window. The Watch Window can be used while PowerTools Pro is running and the PC is online with the device. The Watch Window allows the user to monitor the status of all the desired system parameters in one location.

An example of the Watch Window is found in Figure 148 below.

To setup the Watch Window, select Tools\Watch Window from the PowerTools Pro menu. If not online with the device, the Watch Window will be unavailable on the menu. Upon selecting Watch Window, the following window will appear.

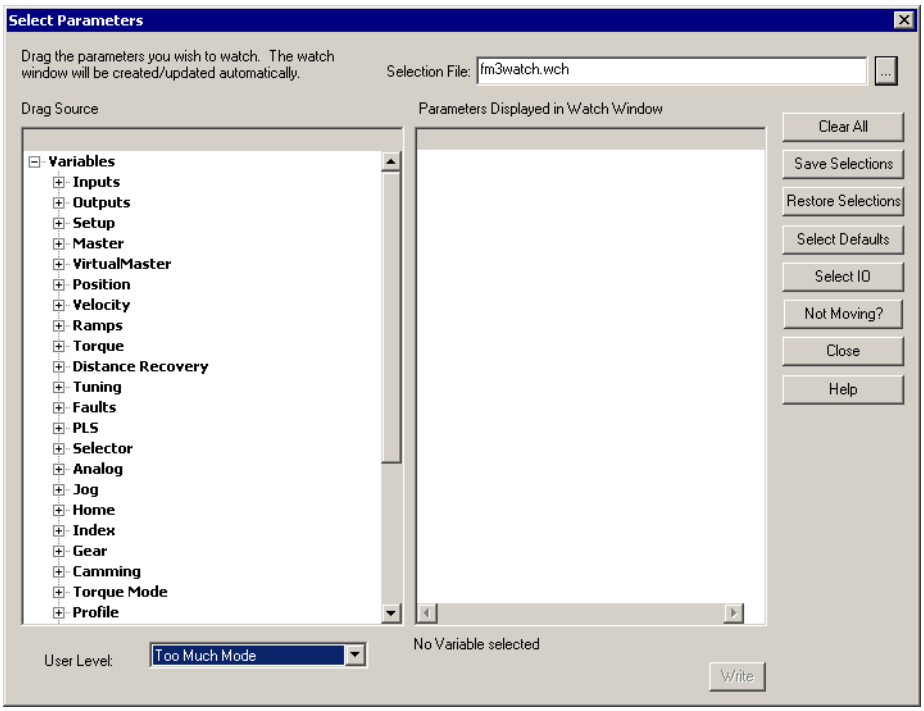


Figure 147: Watch Window, Select Parameters Window

The Select Parameters window as seen in Figure 147 allows the user to specify which parameters are to be seen in the Watch Window. To select a parameter for the Watch Window, simple double-click on the parameter in the Drag Source window or drag and drop the parameter from the Drag Source window on the left over to the Parameters Displayed in Watch Window on the right and it will be added to the Watch Window.

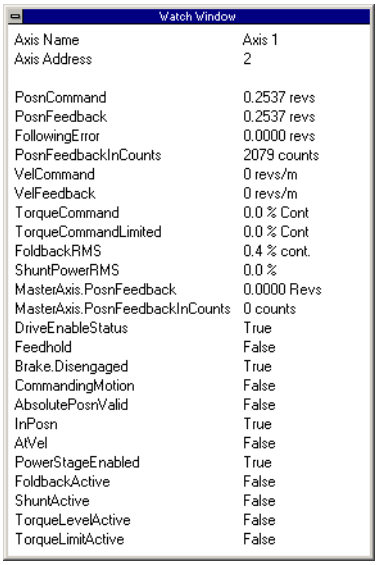


Figure 148: The Watch Window

Once a parameter is added to the Watch Window, it's 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 Parameters window:

Clear All

By clicking on the Clear All button, all of the parameters in the Parameter Displayed in Watch Window pane will be erased and

the Watch Window closes.

Save Selections

By clicking on ***Save Selections***, the user can save the specific parameters that have been added to the Watch Window. 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 settings can be saved in the default file named “fm3watch.wch” or the user can create their own file. Setup the parameter that will be displayed in the Watch Window, enter the file name in the Selection File text box, then click the ***Save Selections*** button.

Restore Selections

By clicking on the ***Restore Selections*** button, the Watch Window will be filled with the list of parameters that were last saved using the ***Save Selections*** button.

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 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.
Fault.Faulted	False	Motion stops on a fault.
DriveOK	True	The drive status must be ok to enable motion.
PowerStageEnabled	True	Must have power to move the motor.
Stop	False	Motion is stopped if stop is active.
PowerSupplyReady	True	PowerSupplyReady will become inactive if the 24 volt logic power is taken away or if the bus voltage drops below the "low bus fault" threshold.
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.
FoldbackActive	False	When Foldback is active it indicates a current overload and the Foldback circuit is limiting the current. Foldback in it’s self indicates motor selection of wiring problems that effect motor motion.
SoftwareTravelLimitMinusActive	False	Motion stops when a travel limit is active.
SoftwareTravelLimitPlusActive	False	Motion stops when a travel limit is active.
TorqueLimitActive	False	Torque limits may effect motor motion as it reduces motor generated torque.
Profile.0.MotionStop	False	Each profile has it’s own Stop. To get motion out of this profile this parameter must be false.
Profile.0.FeedHold	False	Each profile has it’s own FeedHold. To get motion out of this profile this parameter must be false.
Profile.1.MotionStop	False	Each profile has it’s own Stop. To get motion out of this profile this parameter must be false.
Profile.1.FeedHold	False	Each profile has it’s own FeedHold. To get motion out of this profile this parameter must be false.

Close

The ***Close*** button will close the Select 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 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.

Write Parameter

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.

7.4.2 Global Where Am I Button

In the Program View, when online and executing a program or sequence of programs, the user can display current program status. Pressing the ***Where Am I*** button on the PowerTools Pro toolbar creates a blue triangle that appears on the line of the program currently being executed.

The Global Where Am I can be used for diagnostics. When the user needs to know where in a complicated program the drive is or when the user wishes to follow the logical flow of the program.

7.4.3 Motion Status

While the drive is online, the name of the program currently running or the motion type currently running will appear in the status bar at the bottom left corner of the PowerTools Pro window.

7.5 Diagnostic Analog Output Test Points

The drive has two 10 bit Analog Outputs which may be used for diagnostics, monitoring or control purposes. These outputs are referred to as Channel 1 and Channel 2. With the Epsilon EP-P drive the Analog Outputs are accessed from the Analog/ Sync Output connector (J5).

The Analog Output source is assigned to the outputs on the Analog Outputs view.

8 Specifications

8.1 Dimensions and Clearances

The following table applies to A and B as shown in Figure 149 below.

Drive Model	Dimension “A” (shown in inches/mm)	Dimension “B” (shown in inches/mm)
EP202-B,-I,-IDN	2.11 [53.59]	0.45 [11.4]
EP204-B,-I,-IDN	2.11 [53.59]	0.45 [11.4]
EP206-B,-I,-IDN	2.82 [71.63]	0.45 [11.4]
EP202-P	2.69 [68.3]	1.03 [26.16]
EP204-P	2.69 [68.3]	1.03 [26.16]
EP206-P	3.40 [86.9]	1.03 [26.16]

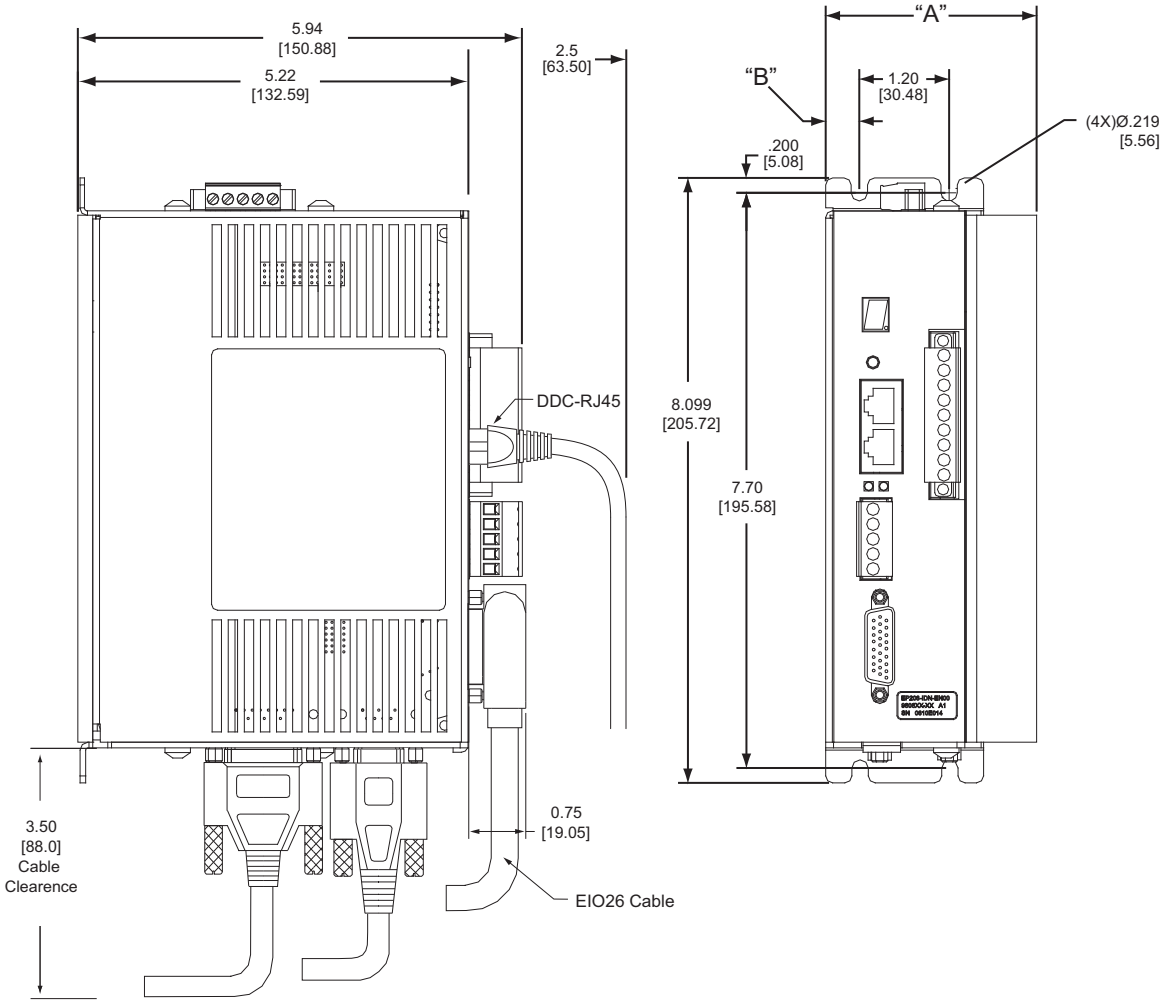


Figure 149: Epsilon EP202 through 206 Drive Dimensions and Clearances

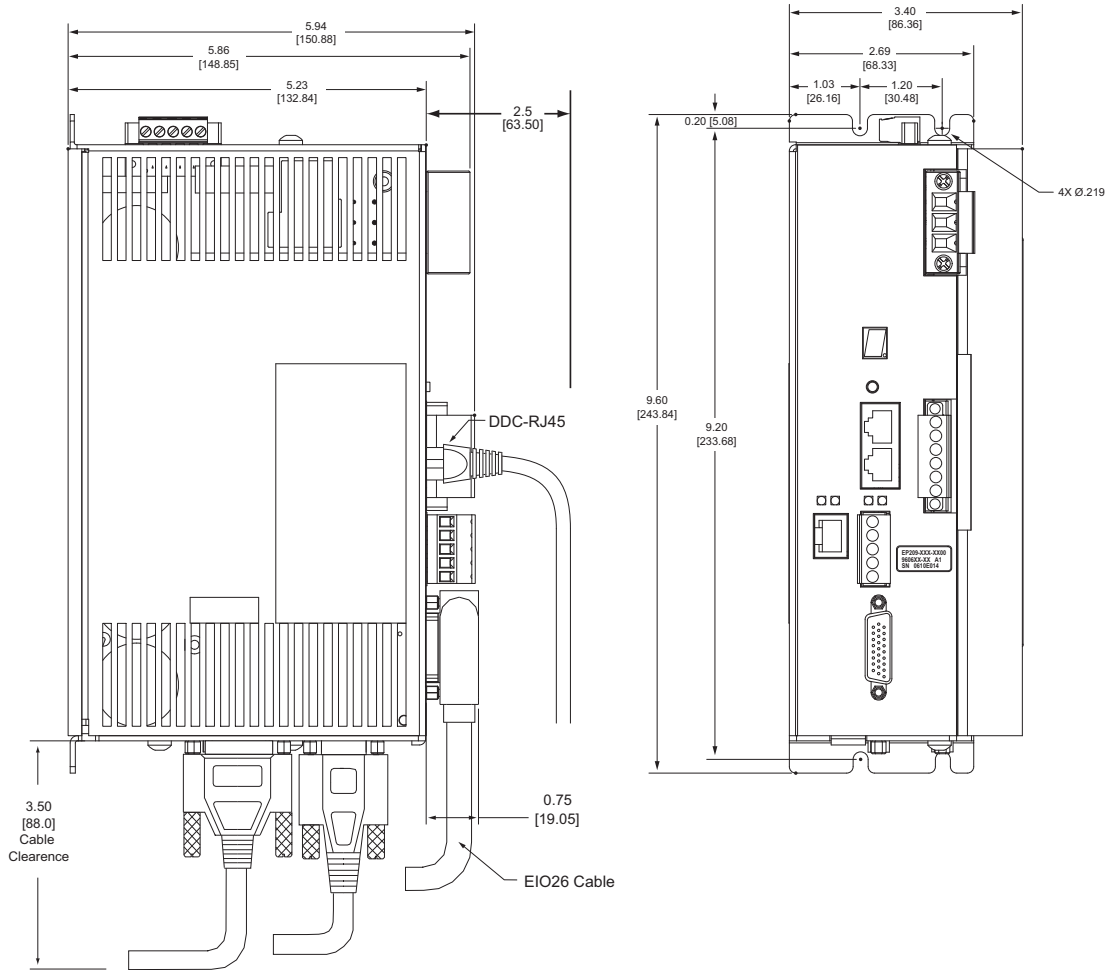


Figure 150: Dimensional Drawing for all Epsilon EP209 Drive

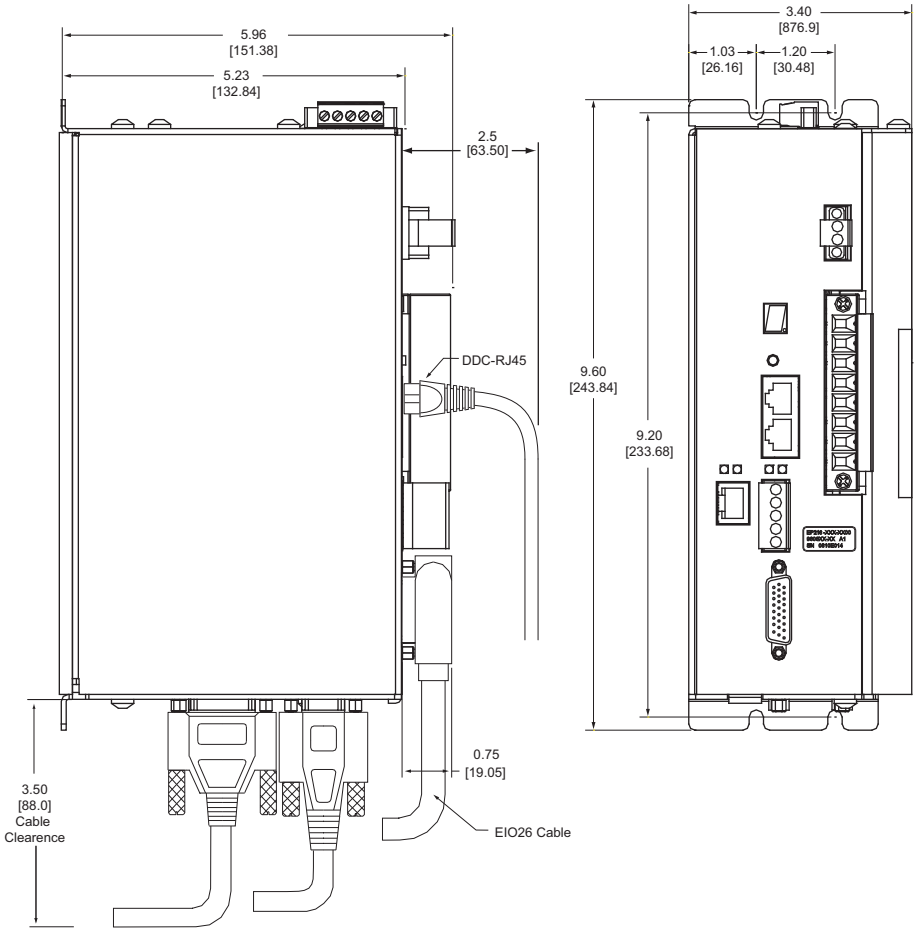
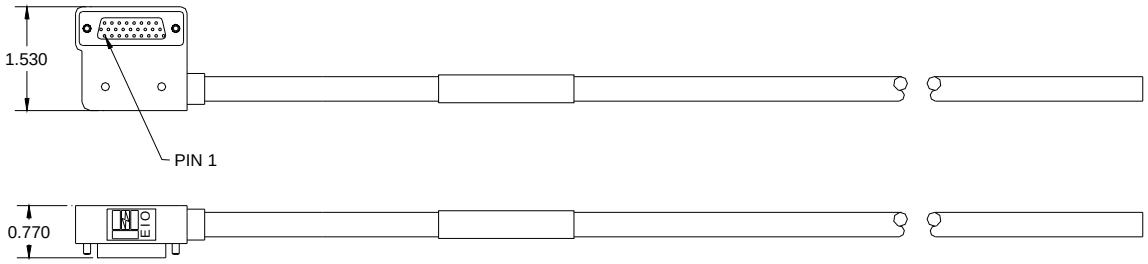
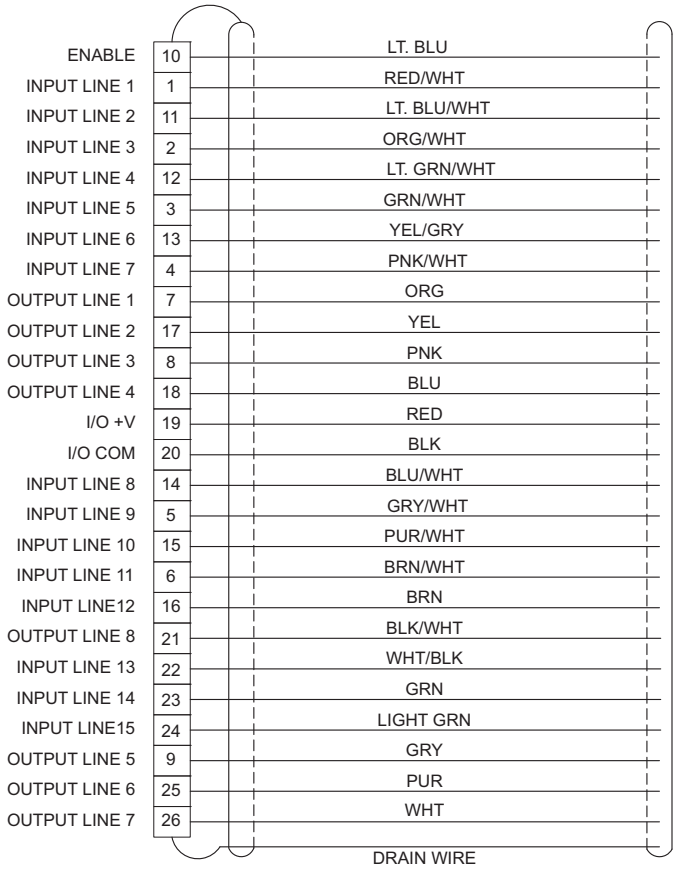


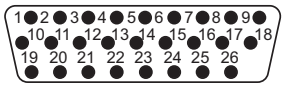
Figure 151: Dimensional Drawings for all Epsilon EP216 Drives

8.2 Cable Diagrams

8.2.1 EIO26



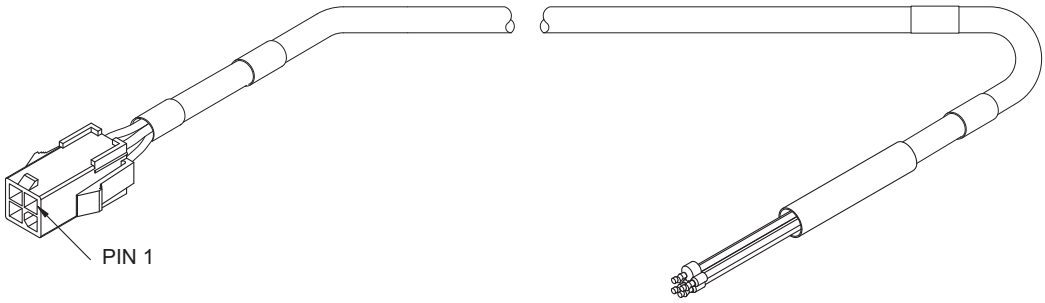


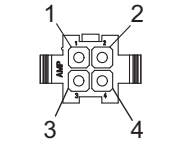


SOLDER SIDE
Socket

8.3 XV Motor Cables

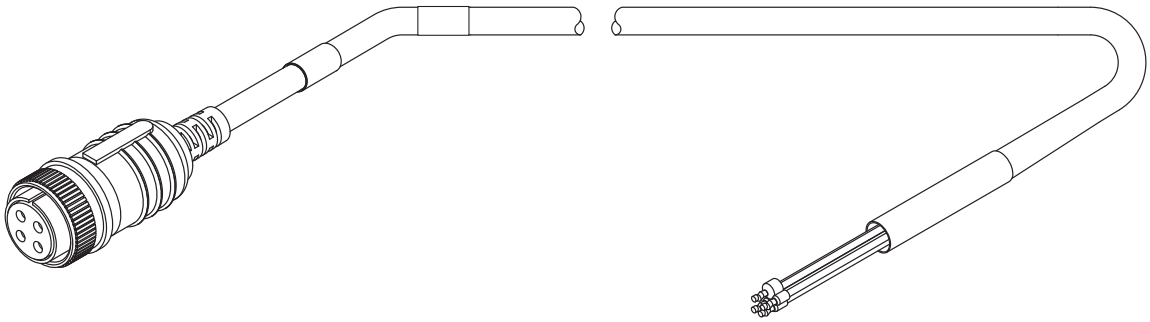
8.3.1 XTMDS

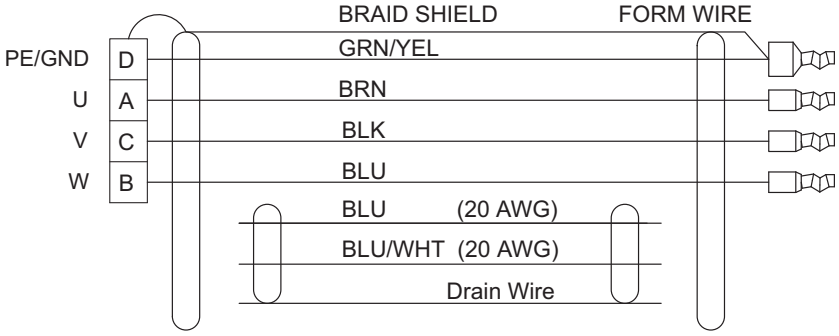


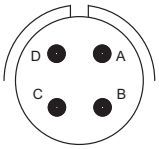


REAR VIEW OF CONNECTOR

8.3.2 XCMDS

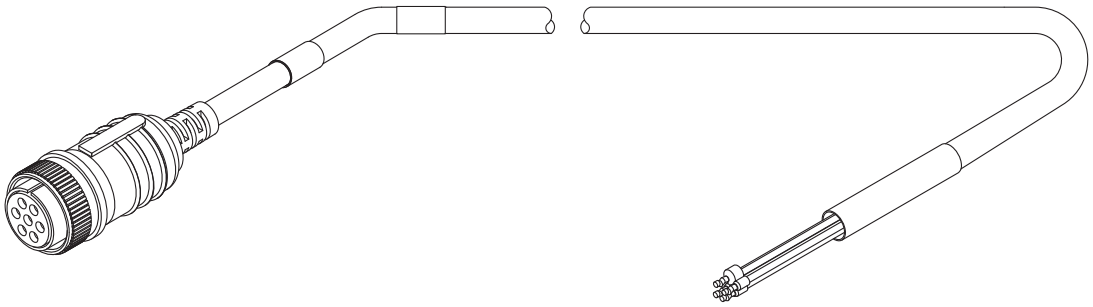


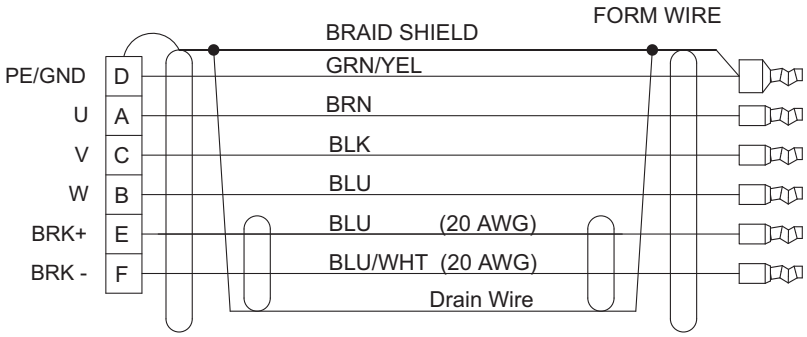


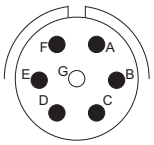


SOLDER SIDE
Socket

8.3.3 XCMDBS

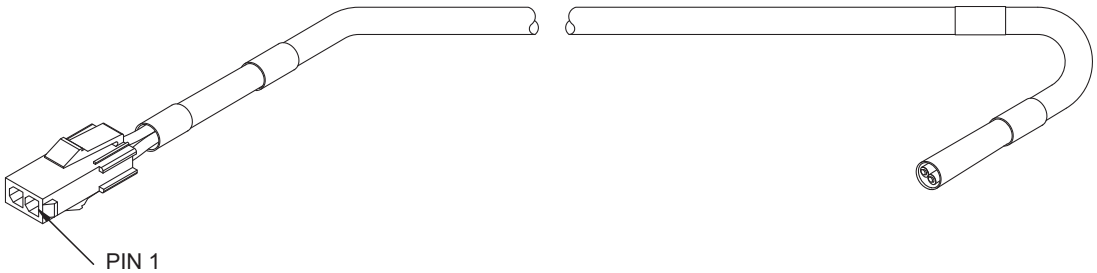


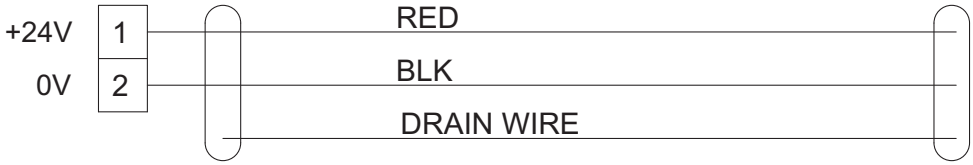


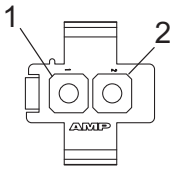


SOLDER SIDE
Socket

8.3.4 XTBMS

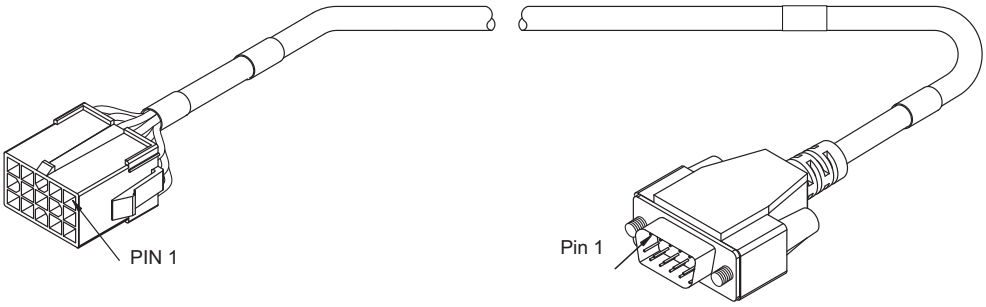


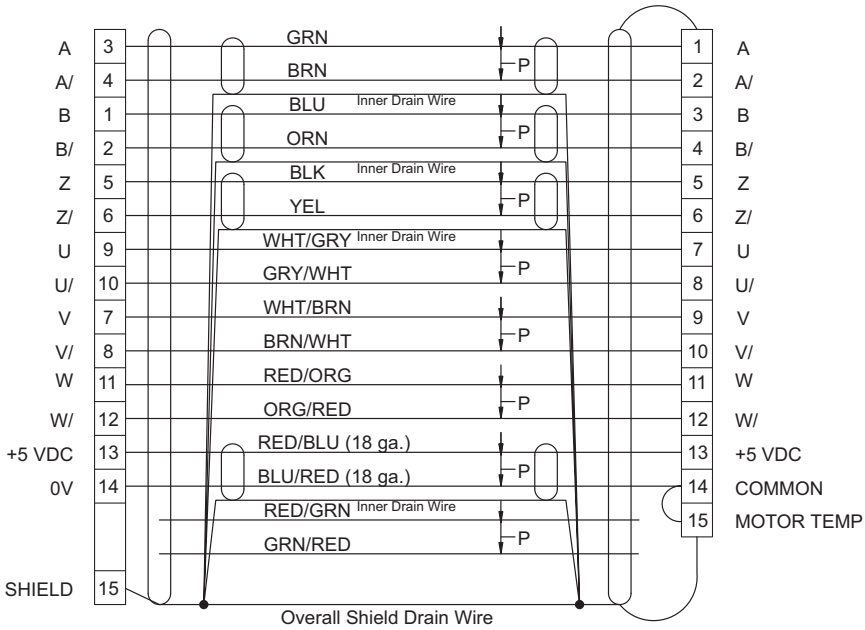


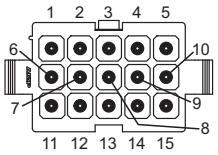


REAR VIEW OF
CONNECTOR

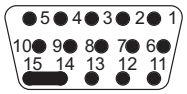
8.3.5 XEFTS/XUFTS





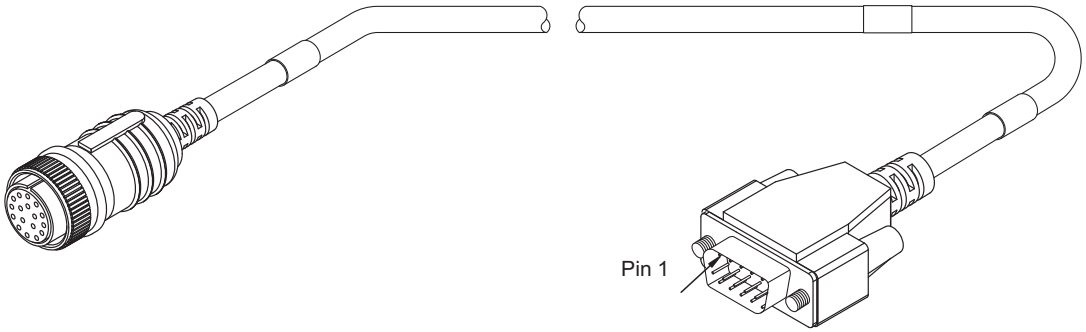


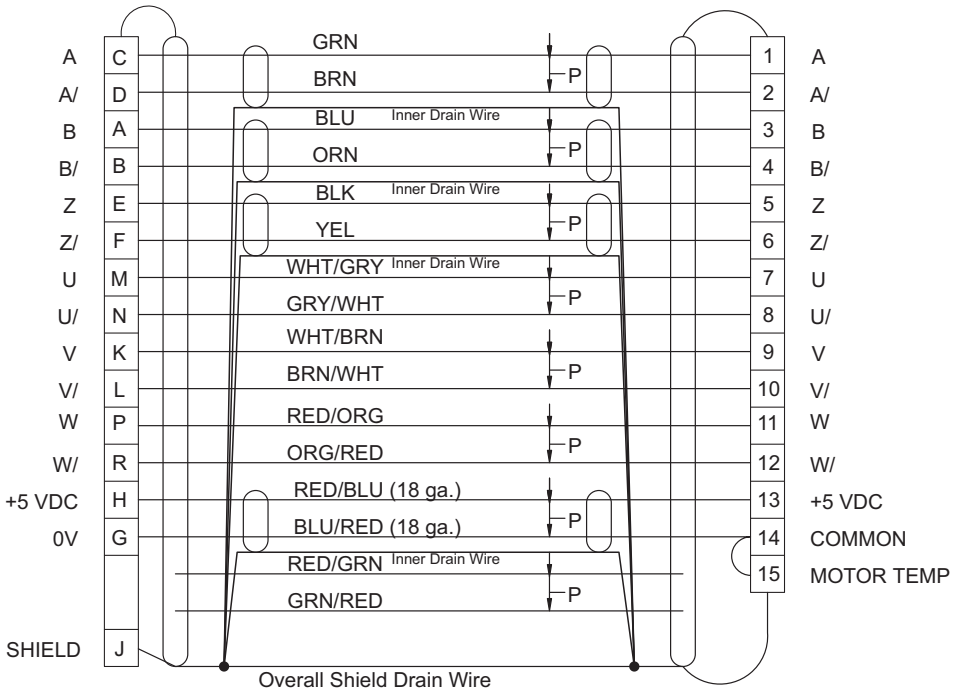
REAR VIEW OF CONNECTOR

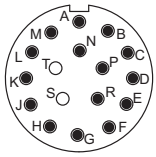


SOLDER SIDE
Pin

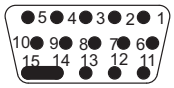
8.3.6 XEFCS/XUFSC







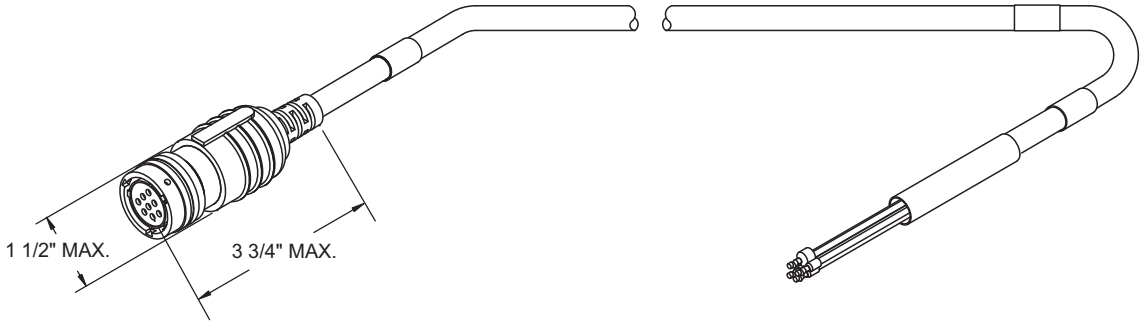
SOLDER SIDE
Socket

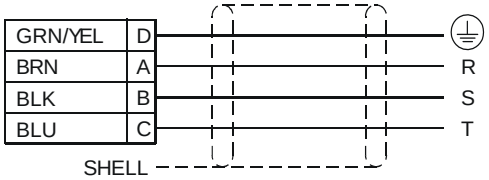


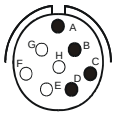
SOLDER SIDE
Pin

8.4 NT and MG Motors Cables

8.4.1 CMDS

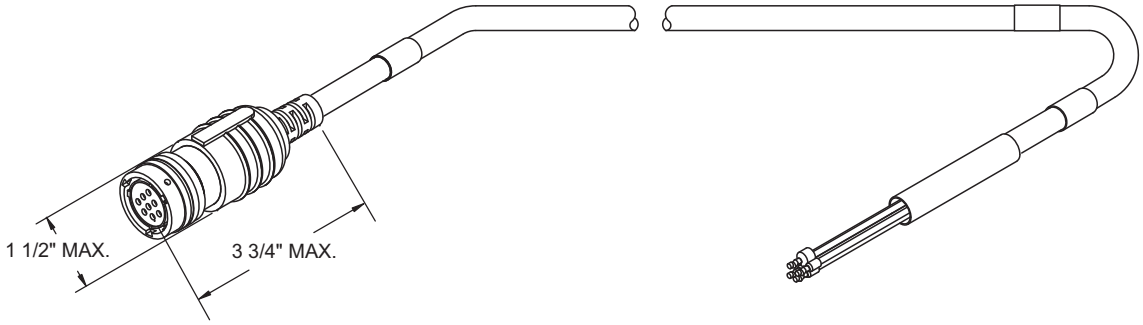


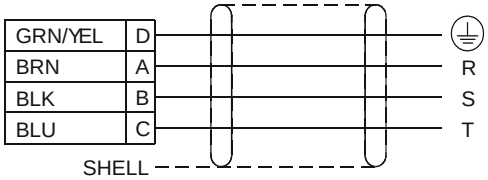


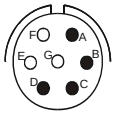


SOLDER SIDE

8.4.2 CMMS

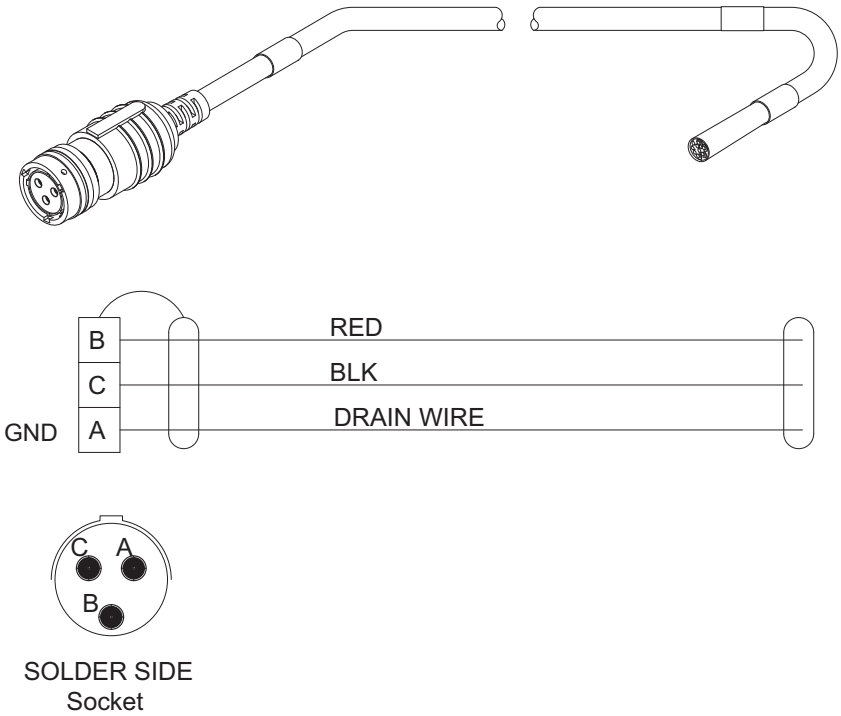




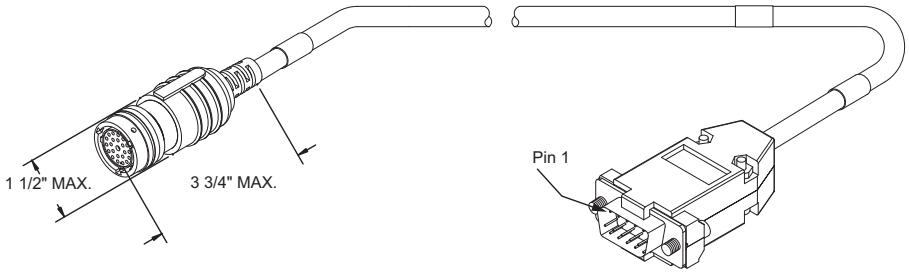


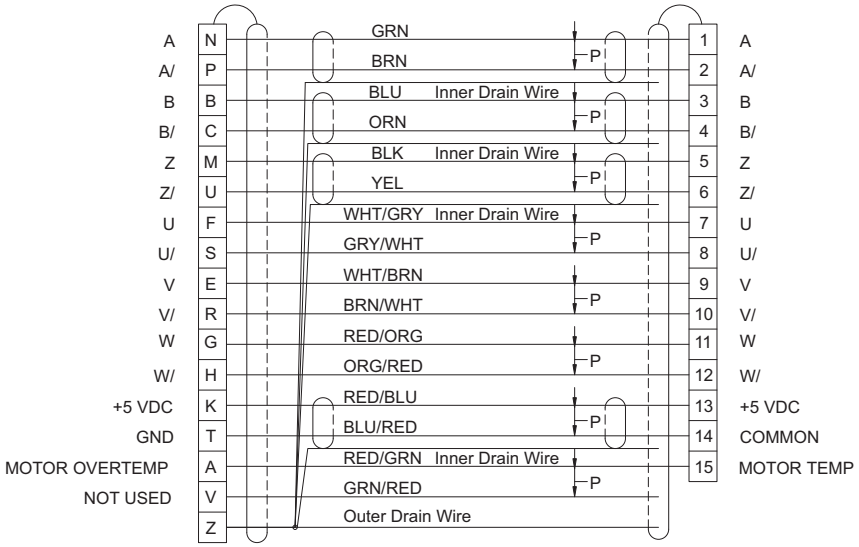
SOLDERSIDE

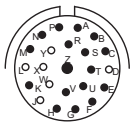
8.4.3 CBMS



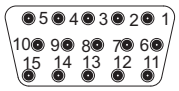
8.4.4 EFCS / UFCS







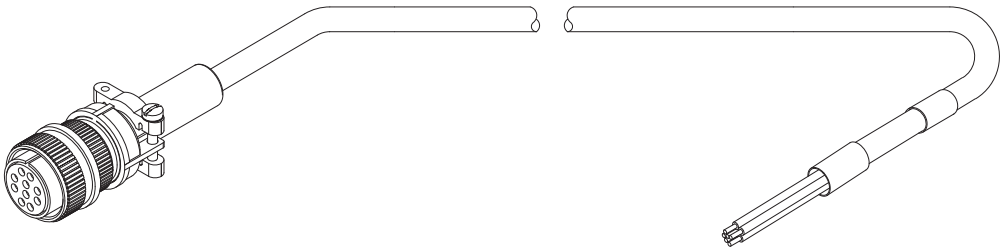
SOLDER SIDE
Socket
To Motor MG/NT/MH

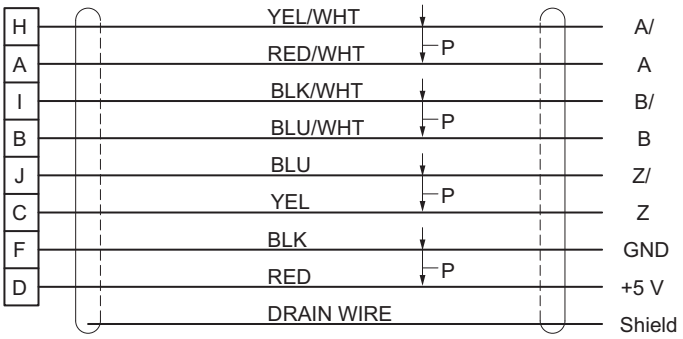


SOLDER SIDE
Pin
To DRIVE

8.5 SYNC Cables

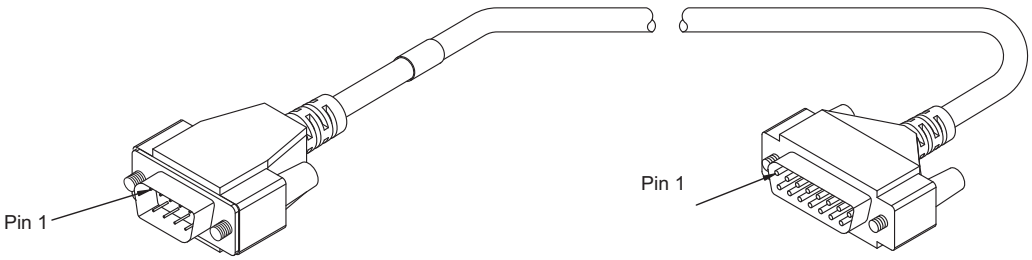
8.5.1 ENCO Cable

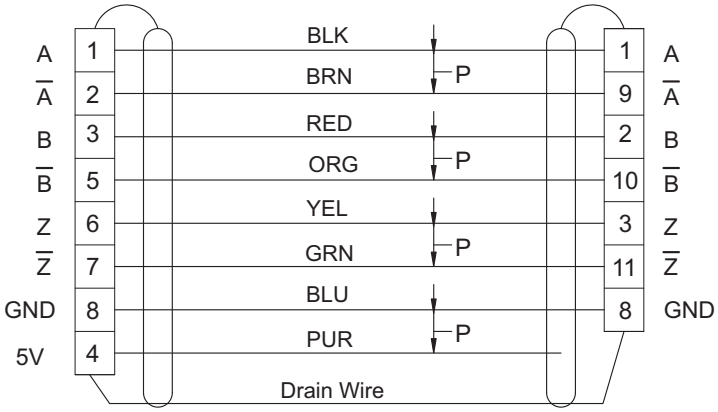


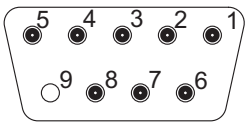


SOLDER SIDE
Socket

8.5.2 SNCDD-915

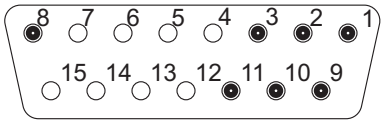






SOLDER SIDE

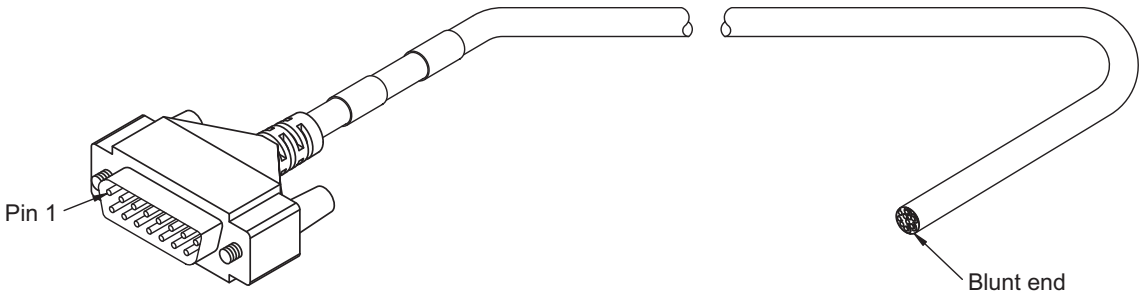
Pin

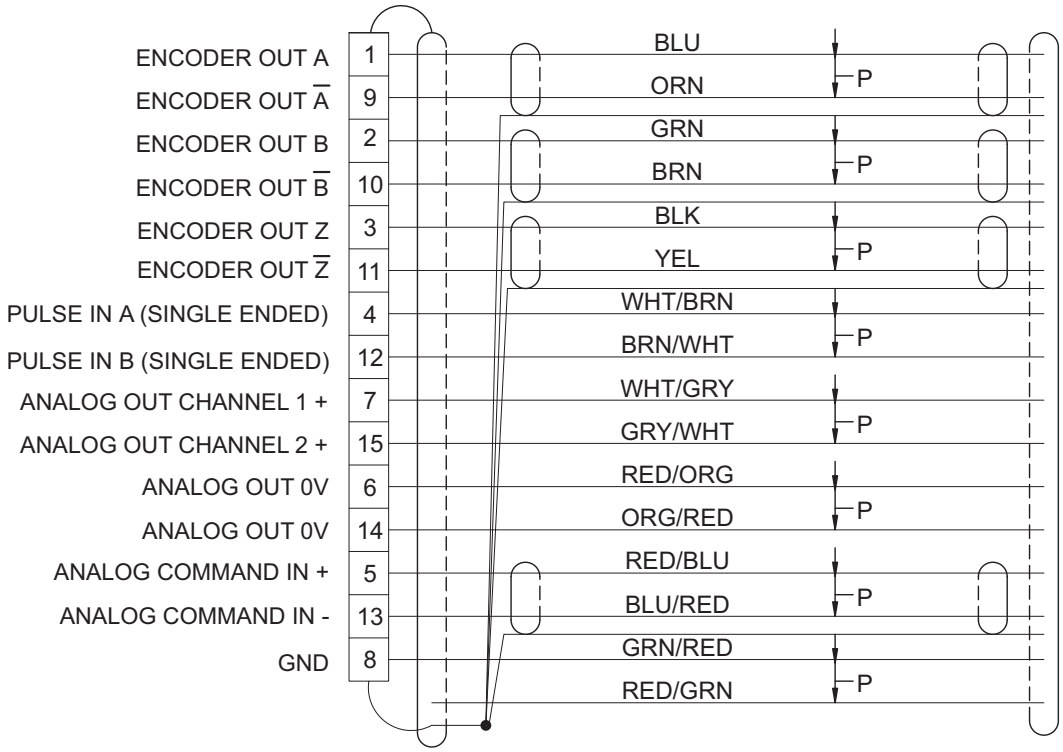


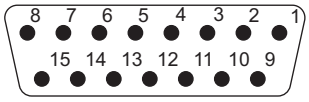
SOLDER SIDE

Pin

8.5.3 SNCFLOA

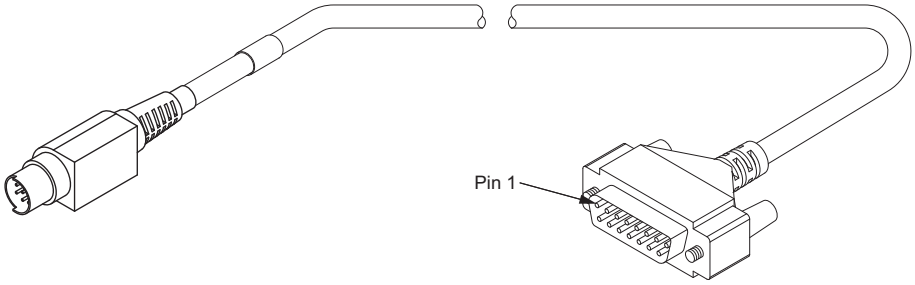


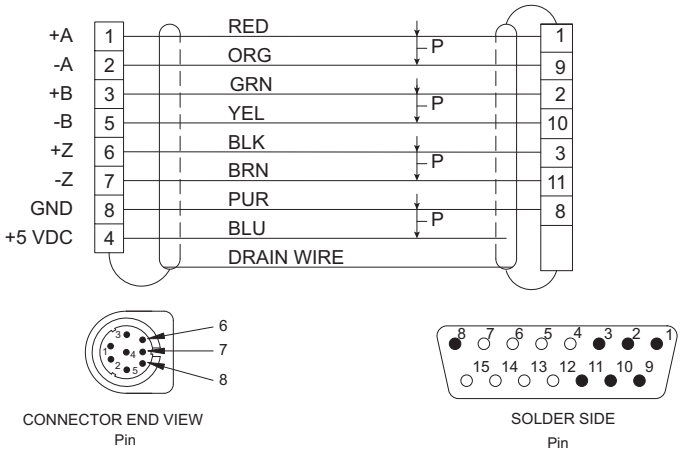




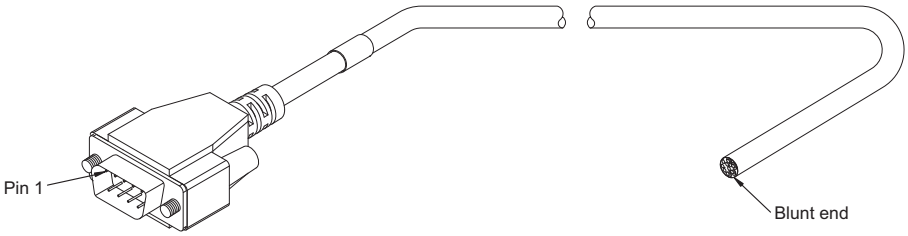
SOLDER SIDE
Pin

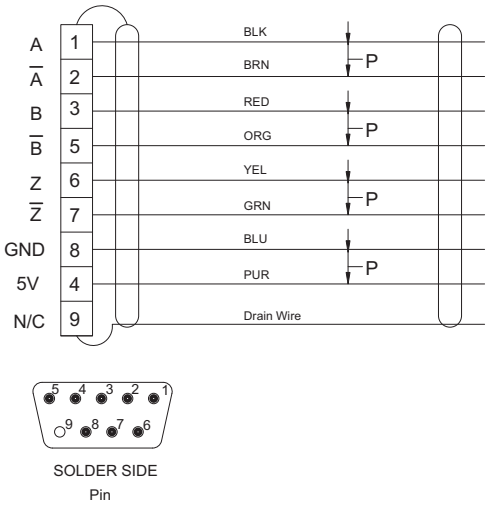
8.5.4 SNCMD-815



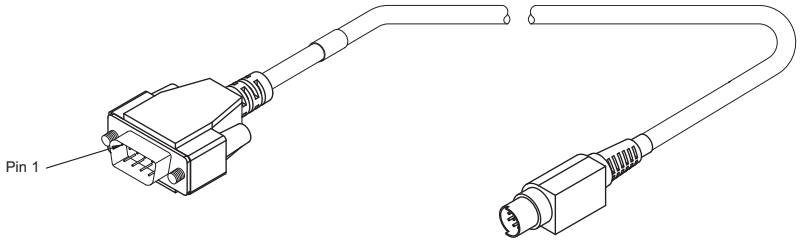


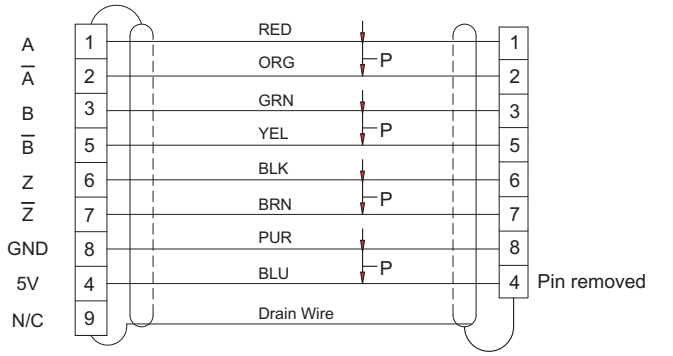
8.5.5 SNCFLI

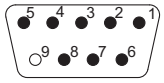




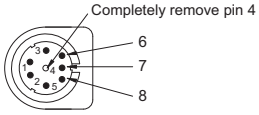
8.5.6 SNCMD-89





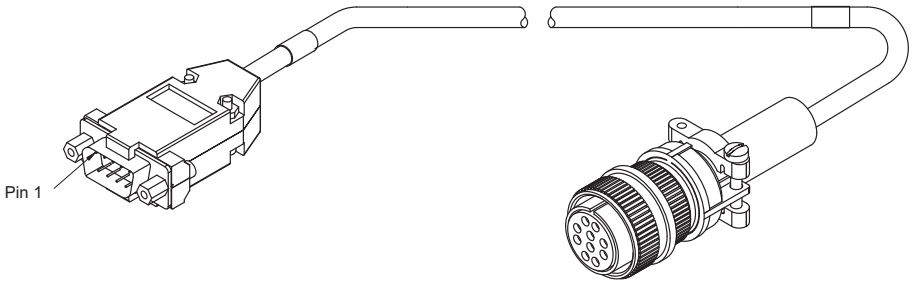


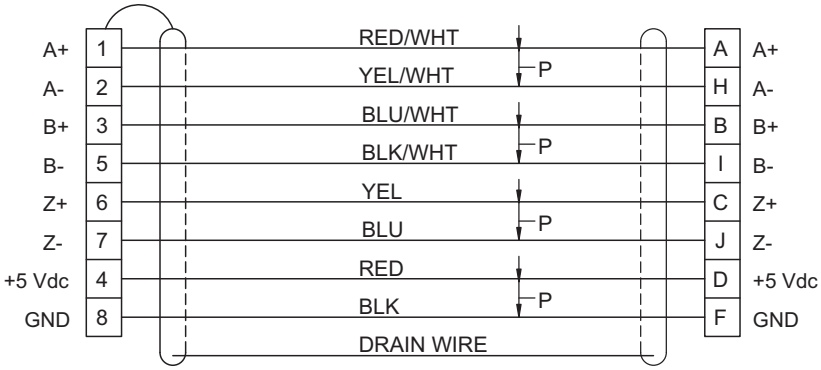
SOLDER SIDE
Pin



CONNECTOR END VIEW
Pin

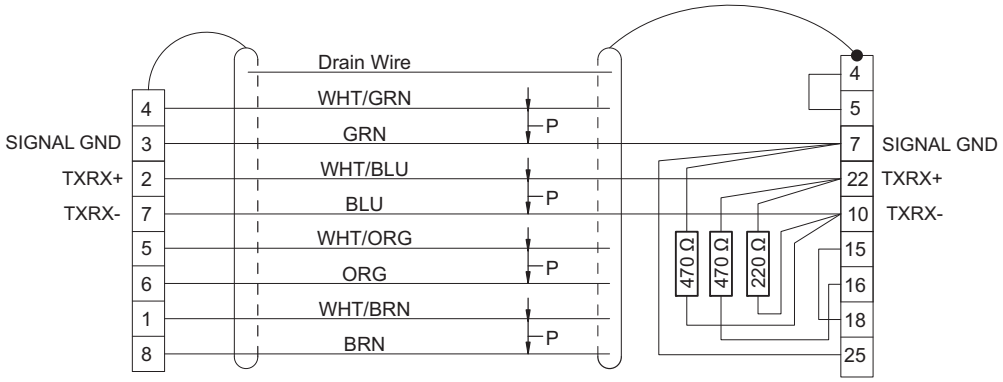
8.5.7 SNCE

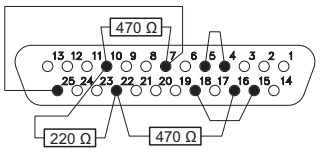




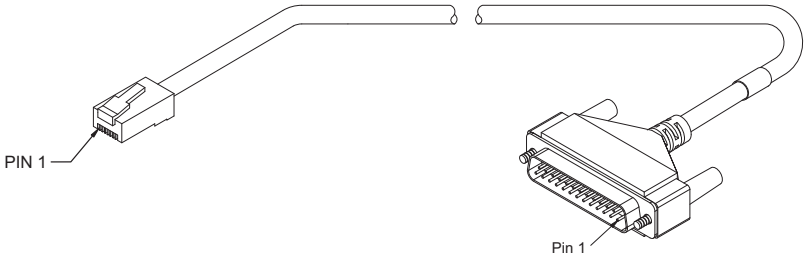
8.6 Communications Cables

8.6.1 ESA-SP-485

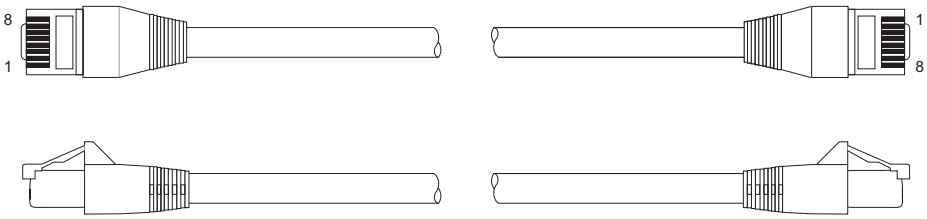




SOLDER SIDE
Pin



8.6.2 ETH-PATCH



9 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 Epsilon EP-P 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 Epsilon EP-P drive, 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 Epsilon EP-P drive. 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 Epsilon EP-P drive, the input function is connected to the action through click and drag operations in PowerTools 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 the computer to a Epsilon EP-P 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 Epsilon EP-P drive, this file loads the firmware into the drive. Flash files can field upgrade the firmware.

Global Where Am I

PowerTools 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 Pro, the input function is connected to the action through click and drag operations in PowerTools 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 the drive.

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.

Most Significant Bit

The bit in a binary number that is the most important or that has the most weight.

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 Pro V#

PowerTools Pro V# is a Windows® based software to interface with the Epsilon EP-P drive.

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 the configuration file.

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.

10 Index

Symbols

A

B

C

D

E

F

G

H

I

J

L

M

N

O

P

R

S

T

U

W

Since 1979, the “Motion Made Easy” products, designed and manufactured in Minnesota U.S.A., are renowned in the motion control industry for their ease of use, reliability and high performance.

For more information about Control Techniques “Motion Made Easy” products and services, call (800) 893-2321 or contact our website at www.emersonct.com.

Control Techniques Americas LLC
Division of EMERSON Co.
12005 Technology Drive
Eden Prairie, Minnesota 55344
U.S.A.

Customer Service

Phone: (952) 995-8000 or (800) 893-2321

Fax: (952) 995-8129
Technical Support

Phone: (952) 995-8033 or (800) 893-2321

Fax (952) 995-8020

graphics/CT3.5.tif @ 300 dpi i
graphics/eia_BlackWhiteOnWhite.tif @ 2400 dpi i
graphics/cover_globe4.eps i
graphics/CT3.5.tif @ 300 dpi i
graphics/Warning_AQ.wmf v
graphics/Caution_AQ.wmf v
graphics/EpsilonEP-202_4EthrOptionISO.eps 1
graphics/EP209FeaturesView2D.eps 2
graphics/EP216PPB_ISO.eps 2
Screens/EPP_Hierarchy.jpg @ 300 dpi 3
Screens/JogView.JPG @ 150 dpi 4
graphics/Home_01.eps 4
graphics/Sensor_01.wmf 5
graphics/Sensor_03.wmf 5
graphics/Sensor_02.wmf 6
graphics/FM_08.wmf 6
graphics/fm_12b.eps 6
graphics/Offset_05.wmf 7
graphics/offset_01a.eps 7
graphics/offset_02a.eps 8
graphics/offset_03a.eps 8
graphics/Home7.eps 9
graphics/FM_09.wmf 9
graphics/Fm_10.eps 10
graphics/Home_09.eps 10
graphics/Index_06.wmf 10
graphics/FM_03.wmf 11
graphics/Index_absolute.gif @ 150 dpi 11
graphics/Index_03.eps 12
graphics/Index_08.eps 12
graphics/Index_04.eps 12
graphics/Index_09.eps 13
graphics/Upload_button.gif @ 75 dpi 14
Screens/UploadDrive.JPG @ 150 dpi 15
graphics/Download_button.gif @ 75 dpi 15
Screens/DownloadDevice_New.JPG @ 150 dpi 16
Screens/ChangePath_New.JPG @ 150 dpi 16
graphics/Upload NVM.bmp @ 300 dpi 17
graphics/NVM Parameters on Download.bmp @ 300 dpi 17
graphics/Update to RAM.bmp @ 75 dpi 17
graphics/Caution_AQ.wmf 17
graphics/updateRAM.wmf 18
Screens/Preference_Comms.GIF @ 150 dpi 18
Screens/Preference_PopUps.GIF @ 150 dpi 19
Screens/Preference_MotorDDF.GIF @ 150 dpi 20
Screens/SecureDownload.GIF @ 150 dpi 21
fm4 screens/Secure Download Screen - 2.bmp @ 150 dpi 21
fm4 screens/Secure File Upload Screen.bmp @ 150 dpi 21
Screens/StatusOnlineTab.GIF @ 150 dpi 23