Bayside Motion Group
27 Seaview Blvd.
Port Washington, NY, 11050
Phone: 516 484 5353
Fax: 516 484 5496
Email is: inside@baysidemotion.com

Tec Tools

Tec Tools Software Manual

For the Bayside 3400 series servo amplifiers

The **Tec Tools** software provides a simple user interface for setting up, tuning and programming the Bayside Digital and Intelligent Servo drives. The simple point and click programming utilizing directory tabs, pop-up windows and instant help features makes any Windows® user feel right at home. The system software includes Events, Commands, Parameters, Macros and user defined variables that allow the drive to be configured to your specific application.

Manual Revision Page

Rev 1.0

- **Tec Tools** original Release

Tec Tool Revision Page

Version 4.0 Build

- Original release

PREFACE

Congratulations!

You've have just purchased the finest amplifier in its class. The 34xx Servo Drive is designed to provide reliable long-term and economical operation in demanding field environments. The Tec Tools software provides a simple user interface for setting up, tuning and programming both the digital and intelligent servo drives.

Tec Tools is a Windows-based graphical user interface providing the user with the tools to quickly and easily setup and the digital or intelligent servo drives. Minimum PC requirements include a Pentium processor, minimum of 8 MB free space, and Window 95 / NT 3.0 or later releases. The system software includes Events, Commands, Parameters, Macros and User Defined variables that allow the servo to be configured to your specific application.

The flexibility of programming allows the user to quickly and easily set-up and configure the digital servo controller. The "Wizards" allow for quick motor and I/O setup along with easy system tuning. The on-board Oscilloscope provides real-time graphical representation of analog, position, velocity, acceleration and/or drive/user variables.

Each BDA-34xx drive includes an integrated heatsink and/or fan kit along with a self-sourcing 115 VAC – 230 VAC single or 3 phase power supply. The BDD-34xx drive requires 18–74VDC. The Intelligent servo servo drives are fully programmable for a single stand-alone package. Their compact size and integrated design simplifies the installation process and reduces downtime should a need for a replacement arise. Since the drives utilize "Flash" memory, they can be preprogrammed and stored indefinitely. The drive firmware is also stored in "Flash" memory making it easily field upgrade through the serial port to take advantage of future enhancements.

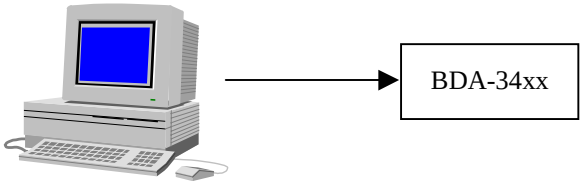
The Digital and Intelligent drives can be set-up in multiple drive modes including Step/Dir., Step Up/Down, Quadrature, Master/Slave, Analog/Digital Velocity or Current (servo only). The Intelligent drive can also store motion programs for stand-alone operation. Available drive modes vary among the Digital and the Intelligent drives as shown below:

Drive Modes	Digital Servo Drive	Intelligent Servo Drive
Analog Current	X	X
Digital Current		X
Analog Velocity	X	X
Digital Velocity		X
Position Mode		X
Gearing (Fixed Ratio)	X	X
Electronic, Variable Gearing		X
Step and Direction Mode	X	X
Step Up/Down	X	X
Quad Encoder	X	X
Spline Tables		X
CAM Tables		X
Macros		X
DeviceNet (Separate Manual)		X
Ethernet (Separate Manual)		X

Fundamentally, the Digital Servo drive is a multi-tasking computer that is dedicated to motion control. It has it's own operating system, data storage (Intelligent Drive), data manipulation capabilities and interface for data communication. The localized inputs/outputs allow for hard wired connections to ensure the drive is in sync with its environment.

There are several ways for the drives to interface with their environment.

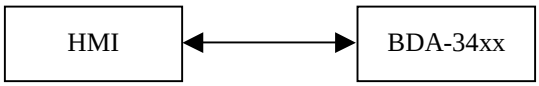
- **Standalone using Tec Tools Software**



Tec Tools

This manual is dedicated to **Tec Tools** software, which is used as a development tool for configuring and/or programming the drive. After the drive is configured or programmed, it can be disconnected from the PC for standalone operation.

- **Handheld Terminal Interface**



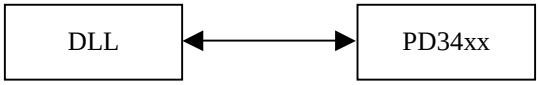
An ASCII terminal interface is used in conjunction with the Intelligent Servo drive in order to change variables, enable/disable or reset the drive, load programs from program library, Start/Stop program execution, check fault history and status, etc. (See section on ASCII Drive Interface)

	NOTE
	The HMI (Human Machine Interface) terminals are used as a data interface and cannot be used to program a drive.

- **ASCII Protocol Interface**

Allows the user through the RS-232/485 to send ASCII commands directly to the drive in order to change variable, load, run programs, and execute move commands. This interface is useful in applications that cannot utilize the DLL drivers such as connected to a PLC. This method does not allow for checksums. (Reference the section on ASCII Protocol.

- **Customer Software Interface**



This method allows the customers software to interface directly to one or multiple servo drives. The intelligent Servo drive can store individual programs with the machine interface changing variables or it can be used to send down complete command strings. The Intelligent drive receives the command string, which are a combination of hexadecimal and ASCII format, and includes a node number and checksum to ensure error free transmission.

The DLL drivers are compatible with:

- Visual Basic rev 4.0 – 6.0
- Labview
- C++
- Microsoft Compatible software

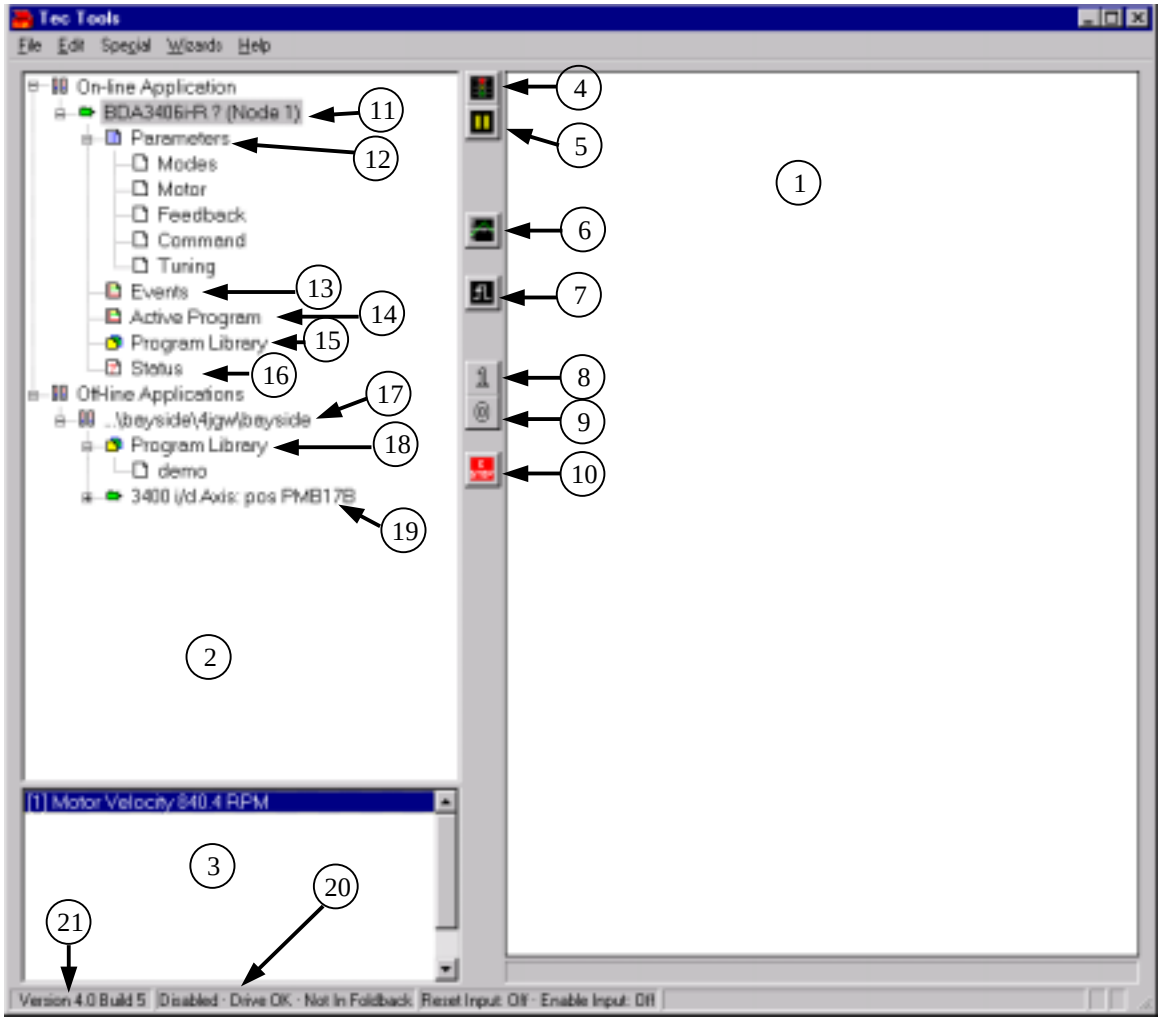
Note:

TABLE OF CONTENTS

1. Tec Tools

The following section discusses the Tec Tools programming screen, providing an overview of the software. Please note that by selecting any item in Tec Tools and clicking the <Right-Mouse-Button> displays that item's sub-menu or help screen.

1.1 Overview



(1) Display Window

This screen displays the information of the highlighted line in the Application Window.

To create new program on the drive or edit an existing program, highlight **Active Program** or <Right-Mouse-Click> on **Program Library** on the drive. Then double click the view window, to open the command window, which lists the available motion commands. The command window allows the user to choose the desired command(s) and enter the required data. Once you click on <Next> or <Previous> the command will appear in the programming window and the cursor will move to the next programming line. To delete a command highlight the line and press the <Delete> key on the keyboard. To insert a line, press the <Insert> key on the keyboard and a blank line will appear above the program line, which is highlighted.

(2) Application Window

The Application Window shows all connected servo drives as well as any open offline applications. To display the application information in the ***Display Window***, just highlight any line.

(3) Monitor Window

The Monitor Window is used to display drive information including feedback position, commanded current, etc. as well as user variables and/or mathematical expressions. The information is updated every 500 msec and is available only when a connected amplifier is present.

(4) Run/Stop Execution Button

This button executes the "**Active Program**". Pressing the button when the stoplight is red will begin program execution. Pressing the stoplight when it is green will stop program execution. The program will automatically stop execution on a software or drive fault.

(5) Pause Execution Button

This command is used in conjunction with the ***"Run/Stop Execution Button***" and will single step through the program. Each command will be executed as the button is pressed. Available only when a connected amplifier is present.

(6) Oscilloscope Button

This button opens the Oscilloscope window. The scope can be used to plot drive parameters verses time. (Example: Feedback Position vs. Time) Available only when a connected amplifier is present.

(7) Toggle Input Button

This button is used to software toggle a single input on and off. To configure the button for a specific input <Right-Mouse-Click> and enter the input. <Left-Mouse-Click> on the button to toggle the input on and off. This allows the user to run a program and toggle the input state from the same screen.

(8) Software Enable On Button

When pressed, this button sends the software enable command through the RS-232 port to the amplifier enabling the drive. Note the enable source for the controller must be set for Software Enabled for the command to work. This button only appears when a connected amplifier is present. See Parameter "Enable Source" (EM).

(9) Software Enable Off Button

When pressed, this button sends the software disable command through the RS-232 port to the amplifier. This will disable the drive if the enable source for the controller is set for software Enabled. Available only when a connected amplifier is present. See Parameter "Enable Source" (EM).

(10) E-Stop Button

Pressing this button sends a global E-Stop to all connected amplifiers via the serial port, hard decelerating the motor to a stop, disabling the drive and displaying an <ES> fault on the amplifier LED. This E-Stop fault will be recorded in the amplifier fault history. The drive must be reset to clear the fault before the active program can be executed. Available only when a connected amplifier is present.

	WARNING
	*Do not use as system E-Stop, this is intended as a tool to assist in programming and debugging an amplifier and does not meet OSHA specifications.*

(11) Connected Axes (Node)

The connected application that will only appear if the controller is powered and connected to the PC through the RS-232 or RS-485 port. Multiple axes will appear if more than one amplifier is connected. The Node number of each axis will appear next to the amplifier.

<Right Mouse-Click> on the Connected Axis:

Firmware Information – Displays the drive model number and firmware revision of the drive.

Set Drive Name - Sets the drive name for the specific axis. The drive name will appear next to the axis.

Calibrate Analog Inputs – Clicking on this command will calibrate all analog inputs to zero.

Axis Memory -> File – Used for application assistance only.

Delete All User Variables - Deletes all user variables in the drive.

(12) Parameters

The parameter field contains the set-up information for the application. When highlighted the parameters are displayed on the display screen. To change a parameter, just double click on it in the ***Display Window***. Parameters are set under the ***Axis Setup Wizard*** and under most conditions should not need to be individually set.

Modes

Drive Mode (DM) – Sets up the drive to operate different types of motors. (Brush DC, Brushless DC, Variable Frequency, Brushless 6-step (trap), Brushless Ignore Halls) Not all drives can run all drive modes. Check manual for specific drive and motor combinations.

Enable Source (EM) – Defines when and how a drive can be enabled. It can be defined based upon an optically isolated enable input, the software enabled parameter (SWE) or a combination of the two.

Command Mode (CM) – The drive can be operated in Position, Velocity, Step & Direction, Open Loop, Step Up/Down, Quad Encoder, or Current Command modes.

Command Source (DCM) – Sets the digital command source to analog or digital. Applies only to drives in current or velocity modes.

Stop Pgm on Faults (SWF) - Determines if the program execution should stop on a software fault. If the SWF is set for active, the positioning program should be written using absolute moves to prevent the axis from having to re-home in order to recover

Check Motor OT Input (COT) – This parameter determines whether or not the drive should check the motor over temperature input. This parameter should only be enabled if a motor over temperature switch is wired to the amplifier.

Input Output Configuration Word (IOCW) – This parameter displays the value of the I/O configuration setup when using the I/O Setup Wizard. To interpret the value of open the I/O Wizard or see system variable “IOCW”.

Motor

Commutation Offset (COFF) – This parameter displaces the motor commutation angle used to commutate the motor.

Motor Inertia (INER) – Displays the motor inertia in (Kg-m2). It is a required parameter used by the "Set-up Wizard" for calculating the optimal tuning parameters for the connected motor. (The inertia is the actual rotor inertia before any gear reduction)

Motor Torque Constant (KT) – Defines the torque constant of the motor in (Nm/Amp). (The KT value winding KT before any gear reduction)

Motor Pole Count (MPOL) – Defines the number of motor poles for the connected motor and is required for generate the correct motor commutation. (2 Pole for Linear motors)

Feedback

Feedback Encoder (EPPR) – Defines the number of pulses per revolution of the feedback encoder and is required when the primary motor feedback (FEED) is set for encoder.

Example: 1024 line differential encoder has 4096 pulse/rev (EPPR=4096)

Feedback Filter (FBF) – Controls the digital filter used on the motor feedback. This low pass filter sets the update rate per second of the velocity loop sample.

Motor Feedback (FEED) – This parameter sets the type of motor feedback. (Resolver, Encoder or Encoder with Hall Encoded Z-Channel)

Position Error Tolerance (PET) – This parameter defines the width of the position window when using the "Wait for In Position" command. It is scaled in motor revolutions**.** A zero value will cause the program to indefinitely "latch" on a Wait for In Position command.

Feedback Resolver (RPOL) – This parameter defines the number of resolver poles of the feedback resolver. Only required when the primary motor feedback (FEED) is set for resolver.

Feedback Invert Bit (FINV) – This parameter inverts the primary encoder signal from the motor and is used to change the direction of the encoder without rewiring. Only required when the primary motor feedback (FEED) is set for Encoder.

Hall Invert Bit (HINV) – This parameter inverts the Hall signal from the motor and is used to change the direction of the hall output without rewiring. Only required when the primary motor feedback (FEED) is set for Encoder with Halls.

Command

Acceleration Rate (ACC) - Sets the acceleration rate (RPM/sec) for Velocity mode only. (In Position Mode the "Set Acceleration" command is used.)

Digital Current Reference (DCC) – This parameter controls the current commanded to the motor when the drive is in Current Mode and the Command Source (DCM) is set to "Digital". (Amps)

Digital Velocity Reference (DCV) - This parameter controls the commanded velocity of the motor when the drive is in Velocity Mode and the Command Source (DCM) is set to "Digital".

Deceleration Rate (DEC) - Sets the deceleration rate (RPM/sec) for Velocity mode only.

Velocity @ 10V Analog Cmd (FSV) – This command sets the amplitude (+/-) of the commanded velocity when the drive is set for analog velocity mode.

Maximum Current (IMAX) – This parameter controls the "Peak" current the drive will supply the motor. (RMS Amps)

Foldback RMS Current Limit (IRMS) – This parameter sets the "Continuous" RMS motor current. To protect the motor and drive the amplifier calculates the I^2t over time and if the average current over time exceeds the RMS current the drive will foldback the current output to the RMS current limit. (Amps RMS)

Overspeed Fault Setpoint (OSPD) – This parameter sets the point at which the over speed fault will be generated. It should be set high enough so that an "OS" fault is not generated during normal operation. For safety reasons, OSPD can be set low when initiating a drive for the first time. (RPM)

Step Pulse Count (SPPR) - Used in the Step/Dir, Step Up/Down, Quad modes as well as for Electronic Gearing. This parameter sets the desired number of step pulses required to command one motor revolution. The ratio of SPPR to feedback resolution sets an implied translation ratio. (Pulses/Rev) Drive must be "Disabled" before value can be changed.

Jog Speed (JOGS) – Sets the jog in the + (CCW) and – (CW) direction when the Jog Inputs are utilized. See I/O Configuration Wizard to configure inputs as jog inputs and to determine the correct drive configuration.

34xx Series	Jog Speeds	2.23 RPM ≤ JOGS ≤ 146,000 RPM

Tuning

Velocity Forward Gain (KF) – This parameter controls the dynamic response of the velocity loop, especially during step changes in the commanded velocity.

Velocity Integral Gain (KI) – This parameter controls the stiffness of the velocity loop and partially it's dynamic response. Increasing the value increases stiffness. Too high of a value could cause instability or oscillation.

Velocity Proportional Gain (KP) - This parameter controls the dynamic response of the velocity loop. It is present in both the velocity and position command modes.

Position Proportional Gain (PPG) – This parameter controls the gain of the position loop. Higher values reduce following errors. Too high a value can cause instability.

CL D-Axis Gain (IKPD) – This parameter is the proportional current gain. This parameter is optimized in the "Axis Setup Wizard for all standard Bayside Motion motors and should not be adjusted. (IKPD and IKPQ should always be equal.)

CL Q-Axis Gain (IKPQ) – This parameter is the proportional current gain. This parameter is optimized in the "Axis Setup Wizard for all standard Bayside Motion motors and should not be adjusted. (IKPD and IKPQ should always be equal.)

CL Integral D-Axis Gain (IKID) – This parameter is the integral current gain. This parameter is optimized in the "Axis Setup Wizard for all standard Bayside Motion motors and should not be adjusted. (Generally should be 10% of IKPD.)

CL Integral Q-Axis Gain (IKIQ) – This parameter is the integral current gain. This parameter is optimized in the "Axis Setup Wizard for all standard Bayside Motion motors and should not be adjusted. (Generally should be 10% of IKPQ.)

Bus Undervoltage Trip Level (BUSU) – This parameter controls minimum Bus Voltage required to trip the "Power Supply Under Voltage" fault (PS) in the (d i) drives only.

	NOTE
	*The default value is 80 VDC for the AC series and 18 VDC for the DC series.* The bus voltage is 1.4 times the incoming AC voltage for the BDA-34xx series.

(13) Events

Events are specific to the intelligent servo drive and are used to monitor conditions in the background while a program is executing. Highlighting "Events" will display the active events being performed as a task by the controller. To add, remove or modify an event double click on the event in the display screen. An event window will appear which allows you to make modifications to the events. By <Right Mouse Clicking> on "Events" the user can Enable or Disable all events. When the drive is powered down or reset, all events are re-enabled.

(14) Active Program of Connected Amplifier

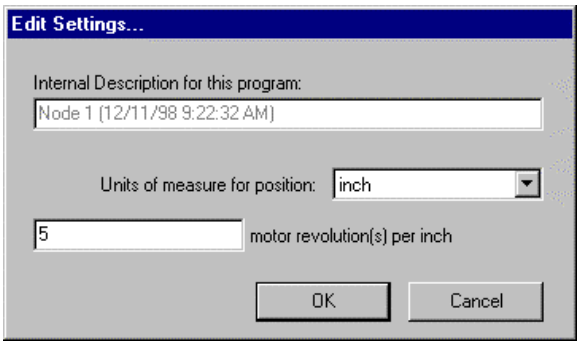
When this line is highlighted, the active program in the amplifier will appear in the display window. To create or modify an active program, double click in the display screen, which will open up the command window. The program must be stopped and it is recommended that the drive be disabled before modifying the active program.

Right Mouse-Click on Active Program:

- **Copy Program To**… - Allows the program to be copied to an off line application library.
- **Store Program in Axis 'Library** – Copies program into the amplifiers Program Library.
- **Clear Program** – "Erases the Active Program" from the drive.
- **Edit Settings** – Sets the "Transition Ration (TR)" for the program. The Transition Ratio allows the user to work in units other than system-revs. The ratio will be saved with the individual program.

  ◄Select units of measurement for position.

  ◄Enter motor revolution(s) per "unit of measurement."



	NOTE
	*The Translation Ratio (TR) is* ***not*** *used when calculating position in an "**Event**". The position units for events are always in System/revs.* ***Example:*** *If the units in the main program are 5 revs/inch. To stop if feedback position greater than 10 inches, X=50 revs.*

(15) Program Library of Connected Amplifier

This field contains all programs stored in the controller. To display any of the programs in the library, just highlight the program name. To copy a program to the "***Active Program***" <Right Mouse Click> on the program and select "***Copy Program To***" then "***Active Program***."

(16) Status of Connected Amplifier

When selected, the Status of the amplifier will be displayed on the display window.

I/O - Shows the status of the drive Inputs and Outputs. Double clicking on an Input or Output allows the user to force it on or off in software.

Fault - The fault window lists the last 16 drive faults. This information is useful in debugging hardware or software. Double clicking on the fault will display additional information.

General - This window displays the status of the drive. Clicking on "***RESET***" will reset the drive and clear re-settable faults.

Variables - This window displays the user variables and the value of the highlighted variable. User variables are non-system variables, created by the user and can be added, deleted or updated from this window.

(17) Off Line Application

This is an open application, which is saved on disk or hard drive of the PC. The application will contain a "***Program Library***" and may contain the amplifier "***Parameters***" and/or "***Events***". To open an existing application, click on "***File***" on the menu bar then click on "***Open Existing Application***"

(18) Off Line Application Program Library

This program library contains program(s), which were created or saved to disk. A new motion program can be created and/or edited offline. To load the program from the Program Library to the Active Program, just highlight the program and right mouse click ***"Copy To***" then click on "***Active Program***". All offline programs are stored in the **Tec Tools** directory under the application name.

(19) Offline Application Axis

The Application Axis contains the set-up "Parameters" and/or the "Events" which were saved from an existing application. To create an Application Axis right mouse click on the **(16) "Application"** and select the axis type (BDA or BDD).

(20) Status Bar

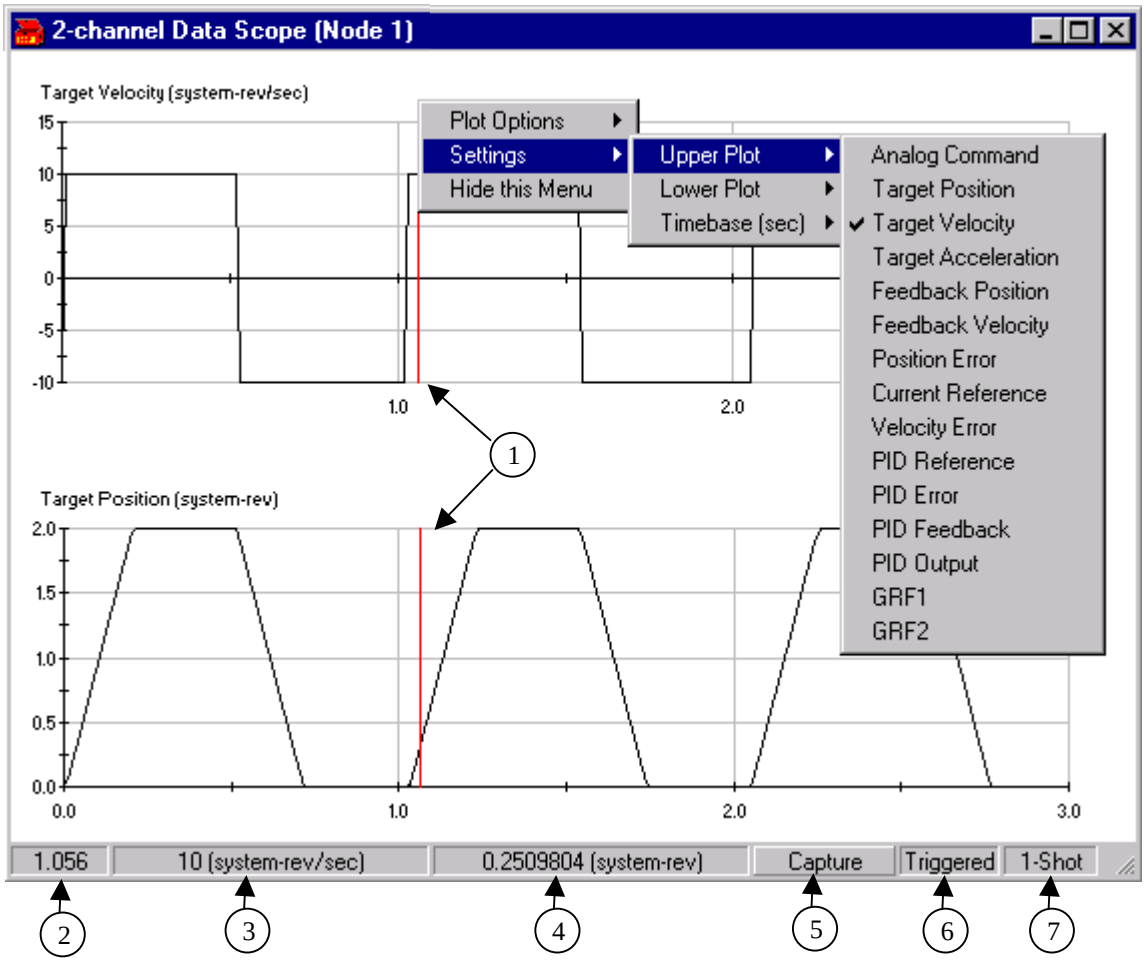
The status bar displays the current status of the highlighted **Connected** drive. Double click on the status bar will reset the drive and clear any resettable faults.

(21) Tec Tools Revision Level

The software revision level and build are displayed in the block.

1.2 Oscilloscope

The Oscilloscope is an online diagnostic tool for tuning an axis, verifying a move profile or even plotting a user defined variable.



GRF1 and GRF2 can be used to plot a drive variable or custom variable which do not appear on plot menu. This is done by using the "Link" command to continuously recalculate the variable every position loop cycle.

Example:

```
Clear Link
Link GRF1=X1*X2+Y
```

Setting Up the Oscilloscope

Left mouse click anywhere on the scope window to open the menu.

Plot Options

Color - The color menu is used to change the Foreground, Background, Data, Grid and marker colors. Note a white background will provide the best printable copies.

Printer - This menu allows the user to print the scope plot to either a printer or a specified file. When saving to a file, **Tec Tools** creates the file name using the Date/Time stamp from the PC.

Settings

Upper Plot - Lists the available drive parameters, which can be plotted on the upper plot. Double click on the desired variable.

Lower Plot - Lists the available drive parameters, which can be plotted on the upper plot. Double click on the desired variable.

Timebase (sec) - This menu selects the time base to be used for the plot. (0.1 to 25 sec)

Hide this Menu - Clicking on the "Hide this Menu" will close the setup menu.

Zoom Function

To zoom on any area of the upper and/or lower plot, hold down the right mouse button and drag the mouse across the desired area. When the mouse button is released, the scope will zoom into the boxed area. To return to normal zoom just double-click the right mouse button.

Scope Features

(1) Crosshair - The crosshair can be moved back and forth across the plot using the mouse. It is used to display the digital plot values on the status bar.

(2) Digital Time Display - This box on the scope status bar displays the digital time value based on the crosshair. By placing the crosshairs at the end of one move then sliding it to the beginning of the next move, an accurate time between moves can be measured.

(3) Upper Plot Display - This box on the scope status bar displays digital value of the upper plot based on the crosshair.

(4) Lower Plot Display - This box on the scope status bar displays digital value of the lower plot based on the crosshair.

(5) Capture Button - Pressing this button sends a software command to the drive to begin capturing data. **Tec Tools** will begin uploading the plot data from the drive or it can be used in combination with the "Trigger Capture" command

(6) Plot Type - Clicking on this box toggles between "Free Run" and "Trigger".

Free Run - The drive will begin collecting data immediately when the Capture Button is pressed.

Trigger - The drive will begin capturing the plot data once the Capture Button is pressed and the "Trigger" command is executed from within the program.

(7) 1-Shot - Clicking on this button toggles between plotting "1-Shot" (One time only) or "Repeat" (Plots continuously).

2. Axis Setup

This section provides general information on setting up the Servo drive/motor. It includes motor selection using the "Axis Set-up Wizard", setting up the drive mode and configuring the I/O.

2.1 Mode Set-up

There are unique parameters that must be configured for each command mode. (Velocity, Current, Position, Step & Dir, Step Up/Down or Quad Encoder) Refer to the appropriate section below:

2.1.1 Velocity Mode

The Velocity mode allows the user to control the velocity via an external analog or digital signal. When the drive is in ***Velocity Mode*** it no longer tracks position.

Required Settings for "**Analog**" Command Source:

- Command Mode (CM): **Velocity**
- Command Source (DCM): **Analog**
- Acceleration Rate (ACC): **(0.1 < ACC < 100,000 RPM/sec)**
- Deceleration Rate (DEC): **(0.1 < ACC < 100,000 RPM/sec)**
- Velocity @ 10V Analog Cmd (FSV): **Maximum Velocity at 10V**

	NOTE
	*The analog signal still needs to be calibrated by right mouse clicking on the drive and selecting "Calibrate Analog Signal".* (**See Calibrating Analog Signal**)

Required Settings for "**Digital**" Command Source:

- Command Mode (CM): **Velocity**
- Command Source (DCM): **Digital**
- Acceleration Rate (ACC): **(0.1 < ACC < 100,000 RPM/sec)**
- Deceleration Rate (DEC): **(0.1 < ACC < 100,000 RPM/sec)**
- Digital Velocity Reference (DCV): **Desired Velocity**

A move command cannot be executed though an Active Program while the drive is in Velocity Mode. This will result in a Software Fault.

2.1.2 Position Mode

The position mode is used in conjunction with an "Active Program". The program executes absolute or relative move commands for motor positioning. This mode is also required for "Electronic Gearing

Required Settings:

- Command Mode (CM): **Position**
- Position Error (PET): **(0.0001– 100,000 revs)**

A program must be loaded in to the ***Active Program*** of the drive and a start program command must be executed either through software or I/O.

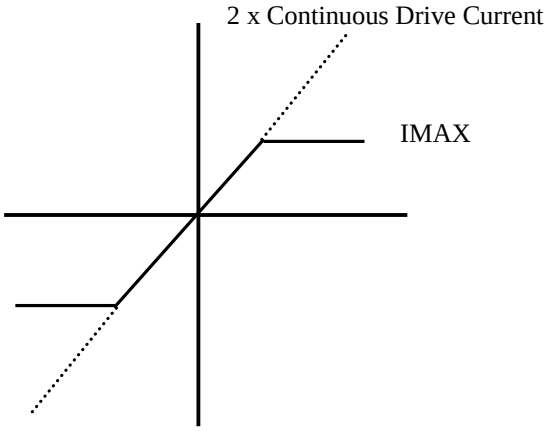
2.1.3 Current Mode

This mode allows the user to control the Current via an external analog or digital signal. When the drive is in ***Current Mode*** it no longer tracks position. The amplifier will send the specified current to the motor windings. The speed of the motor is a function of the current reference and the torque in the system. For Analog command source the amplifier sets maximum current at 10V to 2 times the amplifier's continuous current rating but the drive limits the current to IMAX.

Required Settings for "**Analog**" Command Source:

- Command Mode (CM): **Current**
- Command Source (DCM): **Analog**

The command current is limited by the parameter **IMAX** and **IRMS** to prevent damage to the drive.



Required Settings for "**Digital**" Command Source:

- Command Mode (CM): **Current**
- Command Source (DCM): **Digital**
- Digital Current Reference (DCC): **(0 < DCC < IMAX)**

A move command can not be executed though an Active Program while the drive is in Current Mode. This will result in a Software Fault.

2.1.4 Step & Dir Mode

This command mode allows the drive to follow a digital step and direction signal. The step and direction inputs are wired to the high-speed differential inputs.

Required Settings:

- Command Mode (CM): **Step & Dir**
- Command Source (DCM): **Digital**
- Step Pulse Count (SPPR): **(1 < SPPR < 100,000 pulses/rev)**

A move command cannot be executed through an Active Program while the drive is in Step & Dir Mode. This will result in a Software Fault.

2.1.5 Step Up/Down Mode

This command mode allows the drive to follow a digital step plus and step minus signal. Both step +/- inputs are wired to the high-speed differential inputs.

Required Settings:

- Command Mode (CM): **Step Up/Down**
- Command Source (DCM): **Digital**
- Step Pulse Count (SPPR): **(1 < SPPR < 100,000 pulses/rev)**

A move command cannot be executed though an Active Program while the drive is in Step Up/Down Mode. This will result in a Software Fault.

2.1.6 Quad Encoder Mode

This command mode is used to follow a quadrantial encoder signal from a master drive. A ratio between master and slave axis can be achieved by setting the Step Pulse Count (SPPR). (d series only)

See Position Mode foe applications requiring electronic-gearing.

Required Settings:

- Command Mode (CM): **Quad Encoder**
- Command Source (DCM): **Digital**
- Step Pulse Count (SPPR): **(1 < SPPR < 100,000 pulses/rev)**

If move command is executed while the drive is in Quad Mode, it will result in a Software Fault. A 1000 line quadrennial encoder will supply 4000 pulses per motor revolution. This is important when setting the ***Step Pulse Count***.

2.1.7 Open Loop (Brush Servo)

This command mode is used to run DC Brush motors without hall or encoder feedback.

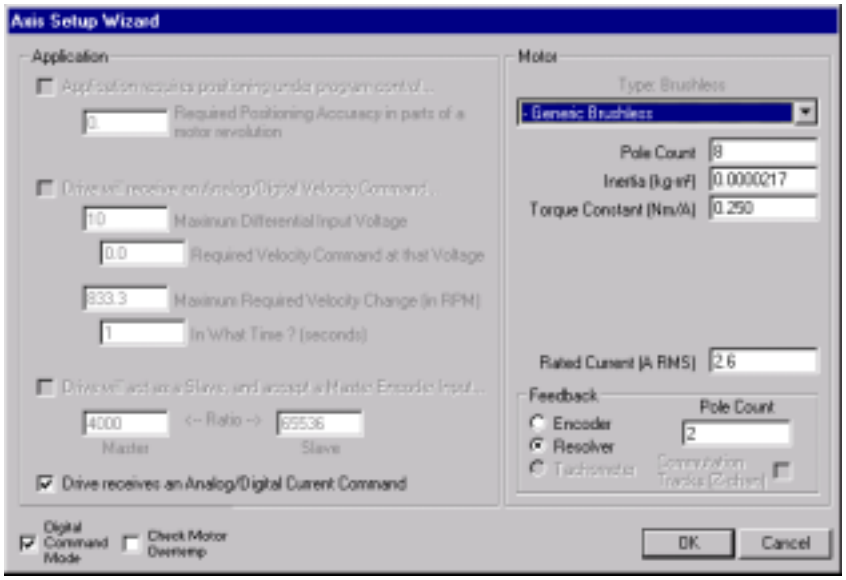
Required Settings:

- Command Mode (CM): **Brush DC**
- Command Source (DCM): **Analog**

2.2 Motor Set-up

To use the Axis Setup Wizard:

- Highlight the axis
- On the toolbar click "Wizard"
- Click "*Axis Setup…*". The axis setup window will appear.



From the Motor window, select the motor from the list of standard Bayside motors. If your motor does not appear on the list, select the generic motor type and enter the applicable motor information. Finally, select the application type (Positioning, Analog Velocity Command or Slave) and complete the required information.

	NOTE
	*When the drive is configured for the given motor it may still require tuning to optimize performance to the specific application.*

	WARNING
	*The "Axis Wizard" calculates several parameters including tuning parameters. Any changes to the Axis Wizard will result in the tuning parameters being recalculated and set to default values.*

2.3 I/O Configuration Wizard

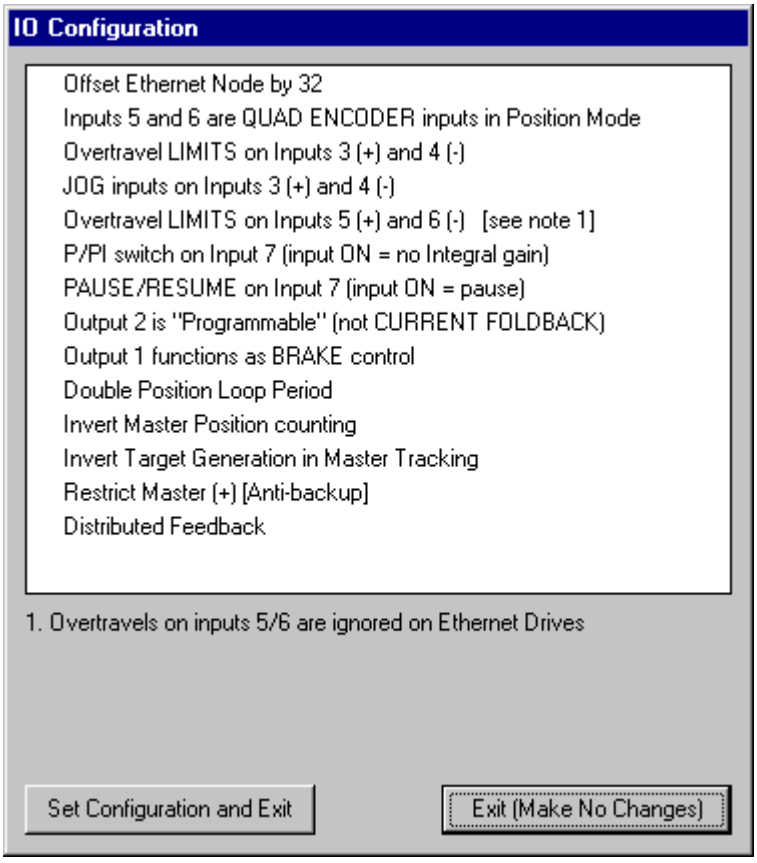
I/O Configuration Wizard

The I/O Configuration Wizard is used to setup specific inputs and/or outputs with specialized functions. The I/O Configuration can also be changed by using the system variable "***IOCW***". The Configuration Wizard can differ between drive types.

To use the I/O Configuration Wizard:

- Highlight the axis,
- Click "***Wizard***" on the toolbar.
- Click "I/O Configuration…". The tuning window will appear.
- Double click on any configuration to activate it.

2.3.1 Digital and Intelligent 34xx Servo Drives



IOCW - The "*IO Configuration Word*" contains the bit values of the I/O configuration set in the I/O Wizard (***Wizard / I/O Configuration…***). Each configuration checked in the I/O Wizard sets a bit. The IOCW variable is equal to the sum of all the bits.

34xx Series Digital and Intelligent Servo Drive:

1 = Inputs 5 and 6 are QUAD ENCODER inputs.
2 = Overtravel LIMITS on Inputs 3 (+) and 4 (-)
4 = JOG Inputs on Inputs 3 (+) and 4 (-)
8 = Overtravel LIMITS on Inputs 5 (+) and 6 (-)
16 = P/PI switch on Input 7 (input ON = no Integral gain)
32 = PAUSE/RESUME on Input 7 (input ON = pause)
64 = Output 2 is "Programmable" (not CURRENT FOLDBACK)
128 = Output 1 functions as BRAKE control
256 = Position Loop period is 4ms
512 = Invert Master Position counting.
1024 = Invert Target Generation in Master Tracking
2048 = Restrict Master (+) [Anti Backup] for Electronic Gearing
4096 = Distributed Feedback
8192 = Offset Ethernet Node by 32

Inputs 5 and 6 are QUAD ENCODER inputs.

When activated this I/O Configuration defines Inputs 5 and 6 as Quad Encoder input. The drive variable MENC can be cleared and set in any mode except for step/dir or step/step. Quad Encoder inputs are required for both CAM and Electronic Gearing applications

Mode	Current *Analog*	Current *Digital*	Velocity *Analog*	Velocity *Digital*	Position	Electronic Gearing	Step/Dir	Step/Step	Quad
Digital									
Intelligent	**X**	**X**	**X**	**X**	**X**	**X**			**X**

Overtravel LIMITS on Inputs 3 (+) and 4 (-)

When activated Input 3 is configured as the CCW limit and Input 4 as the CW limit. Regardless of the "Drive Mode", when a limit becomes active the amplifier will hard decelerate any motion in the direction of the active limit. Requires normally closed Limit Switches.

Mode	Current *Analog*	Current *Digital*	Velocity *Analog*	Velocity *Digital*	Position	Electronic Gearing	Step/Dir	Step/Step	Quad
Digital	X	X	X	X			X[2]	X[2]	X[2]
Intelligent	X	X	X	X	X[1]	X	X[2]	X[2]	X[2]

X1 – Stops all motion if either input is active

X2 – Target generator continues when limit is active, motor jumps to position when limit removed.

	WARNING
	*Limits must be active to and beyond the ends of mechanical travel in order to insure the motor will stop and remained stopped.*

JOG Inputs on Inputs 3 (+) and 4 (-)

Defines Input 3 as the CCW Jog input and Input 4 as the CW Jog input. The Jog speed is set under the ***Parameters/Command*** screen or by using the system variable "***JOGS***". When the Jog input becomes active the motor will rotate at the defined jog speed. When the input becomes inactive the will motor decelerate to a stop.

Mode	Current *Analog*	Current *Digital*	Velocity *Analog*	Velocity *Digital*	Position	Electronic Gearing	Step/Dir	Step/Step	Quad
Digital			X	X					
Intelligent			X	X	N	N			

N – The program cannot be running for the Jog Inputs to work.

	NOTE
	*Jog Inputs are only functional in **Position** or **Velocity** Mode.* ***Velocity Mode*** *- When a jog input becomes active the motor will accelerate or decelerate moving at the jog speed in the direction defined by the input.* ***Position Mode*** *- The jog inputs are ignored if the program is running. A "**Start/Stop Pgm on Rise/Fall of Input**" event can be used to suspend the program in order to use the jog input while the motor is moving.*

Overtravel LIMITS on Inputs 5 (+) and 6 (-)

When activated Input 5 is configured as the CCW limit and Input 6 as the CW limit. Regardless of the "Drive Mode", when a limit becomes active the amplifier will hard decelerate any motion in the direction of the active limit. Requires normally closed Limit Switches to de-activate limits.

Mode	Current *Analog*	Current *Digital*	Velocity *Analog*	Velocity *Digital*	Position	Electronic Gearing	Step/Dir	Step/Step	Quad
Digital	X	X	X	X			X[2]	X[2]	X[2]
Intelligent	X	X	X	X	X[1]	X	X[2]	X[2]	X[2]

X1 – Stops all motion if either input is active

X2 – Target generator continues when limit is active, motor jumps to position when limit removed.

	WARNING
	*Limits must be active to and beyond the ends of mechanical travel to insure the motor will stop and remained stopped.*

P/PI switch on Input 7 (input ON = no Integral gain)

When configured, Input 7 removes the Integral Gain K_I from the Velocity PI loop. This can be utilized if the integral gain is done in a main controller or to soften the reaction of the motor.

Mode	Current *Analog*	Current *Digital*	Velocity *Analog*	Velocity *Digital*	Position	Electronic Gearing	Step/Dir	Step/Step	Quad
Digital			X	X			X	X	X
Intelligent			**X**	**X**	**X**	**X**	**X**	**X**	**X**

PAUSE/RESUME on Input 7 (input ON = pause)

When configured, Input 7 becomes a "***Pause***" and "***Resume***" input. When activated, the program will pause execution, the motor if moving will decelerate to a stop in Position Modes. The motion will not stop if you are in Current, Velocity or one of the Step modes (only program execution stops). When Input 7 is deactivated, the program will resume execution and the motor will complete its motion using the current acceleration rate. This command has no effect on the "Wait" command or Gearing Macro.

Mode	Current *Analog*	Current *Digital*	Velocity *Analog*	Velocity *Digital*	Position	Electronic Gearing	Step/Dir	Step/Step	Quad
Digital									
Intelligent					X	X			

Output 2 is "Programmable" (not CURRENT FOLDBACK)

This configuration converts Output2 to a general-purpose output. On the 34xx series drives Output 2 defaults to the "Current Foldback" output. (When the drive goes into Foldback Output 2 becomes active.)

Mode	Current *Analog*	Current *Digital*	Velocity *Analog*	Velocity *Digital*	Position	Electronic Gearing	Step/Dir	Step/Step	Quad
Digital									
Intelligent	X	X	X	X	X	X	X	X	X

Output 1 Functions as BRAKE control

When activated this configuration uses output 1 as a Brake control. When the axis is moving the Output is active, holding the brake open. When the motor stops the output is inactive applying the "Power Off" brake. Output 1 becomes active 200 msec after the unit is enabled. The output becomes in-active immediately when the unit is disabled.

Mode	Current *Analog*	Current *Digital*	Velocity *Analog*	Velocity *Digital*	Position	Electronic Gearing	Step/Dir	Step/Step	Quad
Digital	X	X	X	X			X	X	X
Intelligent	X	X	X	X	X	X	X	X	X

Position Loop Period of 4ms

When activated this configuration changes the I-drive position update loop from 2ms to 4ms. This is required when using macros and gearing commands. When used in modes other than gearing it will speed-up overall processing and communication time.

Mode	Current *Analog*	Current *Digital*	Velocity *Analog*	Velocity *Digital*	Position	Electronic Gearing	Step/Dir	Step/Step	Quad
Digital									
Intelligent	X	X	X	X	X	X	X	X	X

Invert Master Position counting.

Activating this configuration will invert the direction of the Master position encoder input. If the master encoder input indicates the motor is moving CW, the drive will interpret the signal as CCW essentially reversing the direction of the slave axis.

Mode	Current *Analog*	Current *Digital*	Velocity *Analog*	Velocity *Digital*	Position	Electronic Gearing	Step/Dir	Step/Step	Quad
Digital							X	X	X
Intelligent					X	X	X	X	X

Invert Target Generation in Master Tracking

Activating this function inverts the direction of the slave axis in respect to the Master Encoder. If the Master Encoder is moving CCW the drive will set the slave target direction as CW reversing the direction between slave and master.

Mode	Current *Analog*	Current *Digital*	Velocity *Analog*	Velocity *Digital*	Position	Electronic Gearing	Step/Dir	Step/Step	Quad
Digital							X	X	X
Intelligent					X	X	X	X	X

	NOTE
	*This command saves the time and expense of rewiring the slave motor and encoder to change direction of rotation.*

Restrict Master (+) [Anti-backup]

When activated this configuration prevents the motor from backing-up when using the electronic gearing commands. The drive will keep track of the master encoder but will not reverse direction. Once the master encoder begins moving in the positive direction the drive will resume moving once the master position reaches its original value before reversing.

Mode	Current *Analog*	Current *Digital*	Velocity *Analog*	Velocity *Digital*	Position	Electronic Gearing	Step/Dir	Step/Step	Quad
Digital									
Intelligent						X			

	NOTE
	*This command prevents the motor from backing-up when doing electronic gearing. This is used in following applications where the slave is not allowed to reverse direction.*

Distributed Feedback

This configuration uses the primary motor feedback for velocity and current loops but use the secondary encoder signal to close the position loop. The secondary encoder is mounted elsewhere in the system and can be used to detect product stretch or slippage.

Mode	Current *Analog*	Current *Digital*	Velocity *Analog*	Velocity *Digital*	Position	Electronic Gearing	Step/Dir	Step/Step	Quad
Digital									
Intelligent					X				

	NOTE
	*The secondary encoder is wired into the HS1 and HS2 and the drive must be in position mode.*

34xxd Digital Drive I/O Configuration

I/O Configuation	Current Analog	Current Digital	Velocity Analog	Velocity Digital	Position	Electronic Gearing	Step/Dir	Step Up/Down	Quad Encoder
Input 5 & 6 are Quad Encoder inputs									
Over Travel Limits on Inputs 3 (+) & 4 (-)	X	X	X	X			X[2]	X[2]	X[2]
Jog Inputs 3 (+) & 4 (-)			X	X					
Over Travel Limits on Inputs 5 (+) & 6 (-)	X	X	X	X			X[2]	X[2]	X[2]
P/PI Switch on Input 7			X	X					
Pause/Resume on Input 7									
Output 3 is "Programmable"									
Output 1 functions as Brake control	X	X	X	X			X	X	X
Position Loop Period of 4 ms									
Invert Master Position Counting									X
Invert Target Generation in Master Tracking									X
Restrict Master (+) (Anti-backup)									
Distributed Feedback									

X1 -Stops all motion if either input is active.

X2 -Target Generator continues when limit is active, motor jumps to position when limit removed.

34xxi Intelligent Drive I/O Configuration

I/O Configuation	Current Analog	Current Digital	Velocity Analog	Velocity Digital	Position	Electronic Gearing	Step/Dir	Step Up/Down	Quad Encoder
Input 5 & 6 are Quad Encoder inputs	X	X	X	X	X	X			X
Over Travel Limits on Inputs 3 (+) & 4 (-)	X	X	X	X	X[1]		X[2]	X[2]	X[2]
Jog Inputs 3 (+) & 4 (-)			X	X	N	N			
Over Travel Limits on Inputs 5 (+) & 6 (-)	X	X	X	X	X[1]	X	X[2]	X[2]	X[2]
P/PI Switch on Input 7			X	X	X	X			
Pause/Resume on Input 7					X	X			
Output 3 is "Programmable"	X	X	X	X	X	X	X	X	X
Output 1 functions as Brake control	X	X	X	X	X	X	X	X	X
Position Loop Period of 4 ms	X	X	X	X	X	X	X	X	X
Invert Master Position Counting					X	X			X
Invert Target Generation in Master Tracking					X	X			X
Restrict Master (+) (Anti-backup)						X			
Distributed Feedback					X				

X1 -Stops all motion if either input is active.

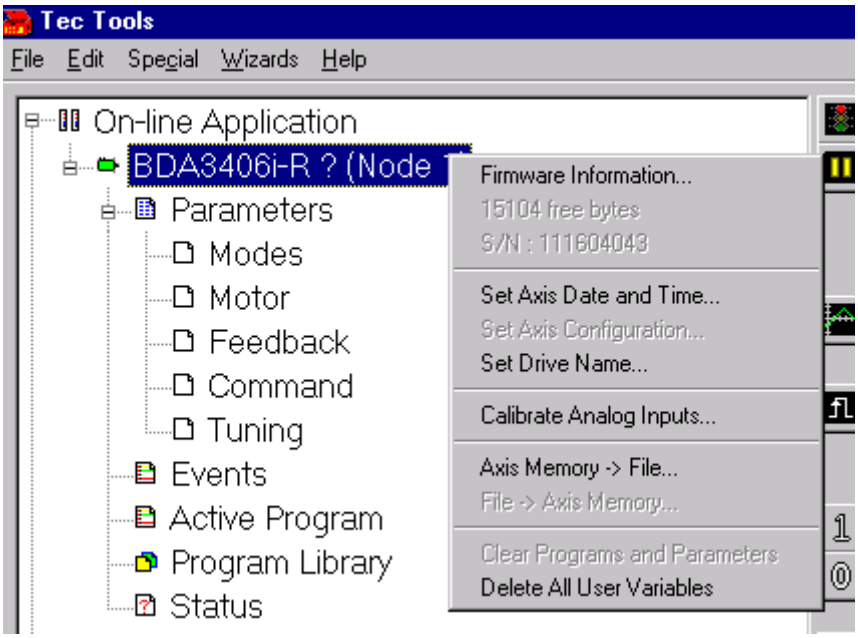
X2 -Target Generator continues when limit is active, motor jumps to position when limit removed.

N - The program can not be running for the Jog inputs to work.

2.4 Calibrating Analog Inputs

To calibrate the analog inputs, highlight the connected axis in the "*Monitor Window*" and adjust all analog signal(s) to the desired zero point.

- Right mouse click on the connected axis
- Click on "Calibrate Analog Inputs…".



The drive will create variables CAL1 and CAL2 as analog offsets and use the supplied analog inputs as the zero reference.

	NOTE
	All analog inputs are calibrated at the same time. If your drive has more than one analog input they must all be set to the desired zero reference before performing the calibration command.

3. Events

This section discusses "Events" which are set within the drive and run as a background task. Events can be used to start/stop a programs, setting/clearing outputs or as software limits. Any event can be enabled/disabled from within the active program.

Events are used to monitor conditions in the background while a program is executing. The user is allowed to define 10 separate events, which can interrupt execution of the program. A red checkmark next to the event signifies the event is enabled. If the event does not have the red checkmark, double clicking on the event and check the enable box to enable it. To maximize speed of execution, variables or equations are not allowed within an event.

	NOTE
	*The check box allows the tech to momentarily disable events for system diagnostics. For safety reasons, the events will automatically be re-enabled when the amplifier is power cycled. The Red-check will not appear in front of events that cannot be disabled.*

3.1.1 <empty>

Signifies that an event does not exist or can be used to erase an existing event.

3.1.2 Run/Stop on Rise/Fall of Input

The program execution begins on the rise of an input (active) and stops when the input falls (inactive). The input number must be defined.

	NOTE
	*The drive must be "Enabled" using the hardware ENABLE input or with parameter SWE within the active program.*

3.1.3 If Feedback Position > X, Stop

This event acts as a software CCW limit. If the feedback position exceeds a specified position, the motor will stop. The user must enter both the maximum feedback position and error number. The error number will appear on the General Status screen as the reason for End of Program. (Revs)

	NOTE
	*The Translation Ratio (TR) is not used when calculating position in an "**Event**". The position units for events are in system/revs.* ***Example:*** If the units in the main program are 5 revs/inch. To stop if feedback position greater than 10 inches, X=50 *revs.*

3.1.4 If Feedback Position < X, Stop

This event acts as a software CW limit. If the feedback position fall below a specified minimum position, the motor will stop The user must enter both the maximum feedback position and error number. The error number will appear on the General Status screen as the reason for End of Program. (Revs)

	NOTE
	*The Translation Ratio (TR) is not used when calculating position in an "**Event**". The position units for events are in system/revs.* ***Example:*** *If the units in the main program are 20 revs/inch. To stop if feedback position less than 1 inches, X=20* revs..

3.1.5 If Fdbk Pos > X, Set/Clr Output

When this event is specified, the drive will set an output when the feedback position exceeds Position X and will clear the output when the position is less than Position X. Both a maximum position and output must be specified. (Revs)

	NOTE
	*The Translation Ratio (TR) is not used when calculating position in an "**Event**". The position units for events are in system/revs.* ***Example:*** *If the units in the main program are 6 cm/rev. To Clear an Output if feedback position greater than 30 cm, X=5 revs.*

3.1.6 If Fdbk Pos < X, Set/Clr Output

When this event is specified, the drive will set an output when the feedback position is less than Position X and will clear the output when the position exceeds Position X. Both a minimum position and output must be specified. (Revs)

	NOTE
	*The Translation Ratio (TR) is not used when calculating position in an "**Event**". The position units for events are in system/revs.* ***Example:** If the units in the main program are 10 cm/rev. To Clear an Output if feedback position less than 5 cm, X=0.5 revs*

3.1.7 Run/Stop on Rise/Fall of Enable

The active program will begin executing when the drive is enabled and stop execution when the drive is disabled. The program always begins execution on the top line.

3.1.8 Quick Stop' if Input is Off

This event can be utilized as a motion limit input. If the specified input becomes inactive when the axis is moving, the motor will HARD stop and the drive will become disabled. Following a 'Quick Stop' the drive must be reset. When using this event, the user must specify the input.

3.1.9 Run from Step on Rise of Input

This event allows the program to jump to a label when the input becomes active but does not interrupt a currently running program. Both an Input and Label must be specified as well as with the drive being enable.

3.1.10 If Following Error > X, Fault

If the Following Error exceeds the specified value the drive will go into a fault condition and abort the program. The user must specify the maximum following error. (Revs)

3.1.11 Output Controlled by Speed

The specified output becomes active when the motor feedback velocity exceeds a predetermined amount otherwise the output is deactivated. The user must enter the speed where the output becomes active. (RPM)

3.1.12 Run/Stop based on Level of Input

On the rising edge of the input the program begins execution. On the falling edge of the input the program suspends execution until the input becomes active, at which time the program will restarts execution at the top.

	WARNING
	*The program can run regardless if the drive is enabled. If a move command is issued while the drive is disabled it can result in the motor taking off when enabled.*

3.1.13 Start Program if Input On

When the selected input is active or becomes active the program begins execution. The level of the Input triggers the event. If the input goes off, the drive does **not** stop execution. User must specify "Start" input.

3.1.14 Start Program on Rise of Input

When the selected input rises, the program begins execution. The event is triggered by the leading edge of the Input. If the input goes off, the drive does **not** stop execution.

	NOTE
	*Unlike the "**Start Program if Input On**" the Input must first be inactive then rise before program execution begins .*

3.1.15 Stop Program on Rise of Input

When the program is executing and the selected input rise (active) the program stops execution. If the input falls, (active) the drive does not begin execution. .

3.1.16 Start Program Running at Power-up

When the drive is powered up the active program will begin execution. The drive must be enabled before any motor movement will occur.

	WARNING
	*The program can run regardless if the drive is enabled. If a move command is issued while the drive is disabled it can result in the motor taking off when enabled.*

3.1.17 Set Output when In Position

This event sets a specified output when the target position (TPOS) and feedback position (FPOS) are within the defined position error tolerance (PET). The output will be cleared when this condition is not true.

4. System Variables

This section provides a list of variables that are available within the drive. Some variables are "Read Only" while other can be set. (See Appendix 1 for variable ranges)

System Variables are predefined drive variables, which represent specific drive registers like, feedback position, motor velocity, analog input, etc. Some variables are read only and can be displayed in the monitor window of **Tec Tools** while other variables can be redefined. All variables can be used in a math expression.

4.1 Analog Inputs

ADC1 - Analog 1 input in volts DC. (–10 to +10 VDC)

ADC2 - Analog 2 input in volts DC. (–10 to +10 VDC) (**I-Drive**)

CAL1 - System Offset Calibration for bias ADC1 (inverse of bias in volts DC)

CAL2 - System Offset Calibration for bias ADC2 (inverse of bias in volts DC) (**I-Drive**)

4.2 Position, Velocity & Torque

ACC - Acceleration Rate (RPM/sec) Analog Velocity Mode Only

CPOS - CAM position, (CAM Rotation 0-1 revs) **Read Only**

DCV - Digital Command Velocity (RPM) Digital Velocity Mode only

DCC - Digital Current Command (Amps RMS) Digital Current Mode only

DEC - Deceleration Rate (RPM/sec) Analog Velocity Mode Only

FPOS - Feedback Position (Revolutions)

JOGS - Sets the jog speed (RPM).

34xx Servo Series Jog Speeds 2.23 RPM ≤ JOGS ≤ 146,000 RPM

Note: Jog Inputs are defined in the I/O Configuration Wizard.

PERR - Position Error (TPOS –FPOS) **Read Only**

PET - Position Error Tolerance (Revs) Used in conjunction with the "Wait for In Position" command

PFLD - Percent current available before going into foldback. This variable displays the percent of average IRMS current available before the average current over time (I^2t) exceeds IRMS current. (0% indicates in Foldback, 50% indicates the average drive current is ½ of IRMS current)

POS - The position in feedback counts. (Rolls over at 131,072 counts) **Read Only**

	NOTE
	*This variable may not be used where a <variable> is required by a PLS, Link or Macro step as it is stored as an integer and not a floating-point decimal.*

POSS - Commanded Position in pulses for Step Up/Down, Step/Dir. or Quad (Steps) **Read Only**

	NOTE
	*This variable may not be used where a <variable> is required by a PLS, Link or Macro step as it is stored as an integer and not a floating-point decimal.*

MARK - Marked Position, (Revolutions) Is the position captured when using a "***Move At***"command. **Read Only**

TPOS - Target Position (Revolutions)

UU - User Units (Represents the units the Translation Ratio (TR) is set for)

0 = cm
1 = degrees
2 = inches
3 = meters
4 = mm
5 = system revs
6 = table revs

VEL - Feedback Velocity (RPM) **Read Only**

4.3 Configuration

BUSU - Bus Under Voltage Trip Level

CDLY - Commutation delay Factory set tuning parameter, motor dependent.

CM - Command Mode:

0 = Velocity
1 = Current
2 = Position
3 = Step & Dir.
4 = Step Up/Down
5 = Quad
6 = Open Loop

COFF - Commutation Offset (Degrees between -180° to 180°)

COT - Check Motor Over Temperature Input (0=Inactive, 1=Active)

CTMP - Drive Temperature in Degrees Celsius

DCM - Command Source (0=Analog, 1=Digital)

DM - **Servo Drive Modes**:
0 = Variable Frequency
1 = Brushless
2 = Reserved
3 = Brush DC
4 = Brushless 6-step (trap)
5 = Brushless Ignore Halls

EM - Enable Source:
0 = Opto Input
1 = SWE and-ed with Opto Input
2 = SWE or-ed with Opto Input
3 = Parameter SWE

EPPR - Feedback Encoder (pulses/rev) (**Encoder** models only)

FBF - Feedback Filter (Hz)

FEED - Motor Feedback Device Type:

0 = Encoder
1 = Resolver
2 = Tachometer/Encoder
3 = Encoder w/ Z- Commutation

FINV - Invert Feedback – Software inverts the feedback direction without rewiring the feedback device.

0 = Default
1 = Invert Direction

FLT - Displays the sum of the Fault Codes of a current fault condition.

Fault Codes:

1 - Over Speed
2 - E/Quick Stop
4 - End Program Fault
8 - Bad Variable/Label
16 - Bad Expression
32 - Feedback Loss
64 - Bad I/O Number
128 - Bad Event I/O Number
256 - Invalid Argument
512 - Spline Error
1024 - Following Error
2048 - Illegal Operation
4096 - Short Circuit Fault
8192 - Bus Fault
16384 - Amplifier Over Temperature
32768 - Motor Over Temperature
65536 - Over-current
131072 - Firmware Fault
262144 - Commutation Fault
524288 - Real-time Loop
1048576- HED Fault Occurred
2097152- Power Supply Undervoltage
4194304- Watchdog Timeout

FSV - Maximum Velocity at 10 VDC used for Analog Velocity Mode. (0 – 18,000 RPM)

GRF1 - Graph Variable 1 is used to graph a custom variable on the Data Scope feature of the drive. The variable can be set up as a link if the variable is continuously changing

Example: Clear Link
Link GRF1=VDC*ONE+ZERO

The Data Scope will plot GRF1, which is the DC buss voltage of the drive.

GRF2 - Graph Variable 2 is used to graph a custom variable on the Data Scope feature of the drive. The variable can be set up as a link if the variable is continuously changing

HED - Displays current status of the HED (Hall Effect Device) Inputs. (Value 1 to 6) **Read Only**

	NOTE
	*This variable may not be used where a <variable> is required by a PLS, Link or Macro step as it is stored as an integer and not a floating-point decimal.*

HINV - Invert Hall Encoded Device – Software inverts the direction the Halls count without rewiring the Halls.

0 = Default

1 = Invert Direction

HSIF - Sets the High Speed Input Filter for the step/dir and/or the secondary encoder inputs. The High Speed Filter filters out electrical noise on the high speed inputs.

IFBK - Displays the current feedback reference. (Actual current to motor in the Digital and Intelligent drive)

IKID - Velocity Loop D-axis Gain, motor dependent.

IKIQ - Velocity Loop D-axis Gain, motor dependent.

IKPD - Velocity Loop D-axis Gain, motor dependent.

IKPQ - Current Loop Q-axis Gain, motor dependent.

IMAX - Maximum Allowed Current (Amps)

INER - Motor Inertia (kg-m^2)

IRA - Current Reference (Amps)

IRMS - Foldback RMS Current Limit (Amps)

KF - Velocity Feedforward Gain

KI - Velocity Integral Gain (Nm/RPM/sec)

KP - Velocity Proportional Gain (Nm/RPM)

KT - Motor Torque Constant (Nm/Amp)

MPOL - Motor Pole Count (Number of motor poles)

OSPD - Overspeed Fault Setpoint (RPM)

PPG - Position Proportional Gain (1000/min)

RERR - This variable contains the value of the "Run Time Fault" when a fault condition occurs due to an "End Program" command or macro fault.

RGNI - Regen Integration Increment **Read Only** (Factory set tuning parameter, motor dependent.)

RGNL - Limit on Integrated Regen **Read Only** (Factory set tuning parameter, motor dependent_

RPOL - Number of resolver poles. (**Resolver** models only)

SPPR - Step Pulse Count per one motor revolution. Used for Step/Step, Step/Direction, Quad or Electronic Gearing Modes (pulses/rev). It is also used in the "Distributed Feedback" mode on the Intelligent drive.

STEP - Is the program step number the drive is executing.

	NOTE
	*This variable may not be used where a <variable> is required by a PLS, Link or Macro step as it is stored as an integer and not a floating-point decimal.*

SWE - Software Enabled; 0=Disabled, 1=Enabled

SWF - Stop Program on Fault (0=Inactive, 1=Active)

THET - The angular position of the motor rotor based on the Z-Channel pulse. Resets to zero once per rotation after receiving Z-channel pulse. (**Read Only**)

	NOTE
	*This variable may not be used where a <variable> is required by a PLS, Link or Macro step as it is stored as an integer and not a floating-point decimal.*

TBAS - This parameter controls the "time base scaling" for all time based commands (except Wait Delay) and has a range from ZERO ≤ TBAS ≤ ONE, which is unit-less. TBAS is used to change time constant for the velocity and acceleration rates. This parameter is useful in conjunction with the LINK command and an analog input. (Creates an effective analog-based "feed-rate override")

Example 1: (Also See Example 7 in appendix)

TBAS=0.5
Wait 3000 msec
Set Acceleration Rate to 2000 system-rev/sec^2
Index of 10 system-revs At 8.2 system-rev/SEC

In the above example the program will dwell 3 seconds, then move 10 system-rev at 4.1system-rev/sec with an acceleration rate of 500 system-rev/sec^2.

Example 2:

Clear Links
Link TBAS=ADC1*ONE+ZERO
Set Acceleration Rate to 2000 system-rev/sec^2
Index of 10 system-revs At 8.2 system-rev/sec

In the above example the "time base" is varied between 0 to 1 based on the Analog 1 input.

	NOTE
	***TBAS** does not effect the "Wait" command or CAM moves* *Velocity = (Velocity/TBAS) and Acceleration = (Acceleration/TBAS^2)* *TBAS above one defaults to 1 and below zero defaults to 0* *See "**Links**" and "**Clear Links**" commands*

TR - Translation Ratio, Ratio of distance to motor revolutions. (0.01 – 180,000)

Example: TR=1.2 (1 inch = 1.2 motor revolutions)

VDC - Bus Voltage present in the drive (VDC) (1.4 x AC voltage)

VFF - VF (Velocity Following) Mode Frequency (Hz)

VFI - VF (Velocity Following) Mode Current Command (Amps)

ZCHN - Represents the Z-Channel pulse. (ZCHN=0 when the drive is first powered up and ZCHN≠0 once the motor passes the Z-Channel the first time) **Read Only**

	NOTE
	*This variable may not be used where a <variable> is required by a PLS, Link or Macro step as it is stored as an integer and not a floating-point decimal.*

ZERO - System Variable = 0 (This will zero the variable to the 14th bit.)

Example: V1=ZERO (V1=0.0000000….)

ONE - System Variable = 1 (This sets One to the 14th bit.)

Example: V1=One (V1=1.000000000….)

4.4 Inputs/Outputs

IN - Bit representation of **standard** inputs, which are active. (Read Only)

Inputs are read in binary from right > left on I/O Screen.

Example 1: If IN&49=16 Then Go To HOME
If IN&49=33 Then Go START

Inputs	7	6	5	4	3	2	1
Value	64	32	16	8	4	2	1

If Inputs 1, 5 and 6 are being tested IN&49

	Active Inputs
IN&49=0	None
IN&49=1	Input 1
IN&49=16	Input 5
IN&49=32	Input 6
IN&49=17	Input 1 and 5
IN&49=33	Input 1 and 6
IN&49=48	Input 5 and 6

This allows the user to check the status of all INPUTS and system variables within a motion program.

Example 2: (Also See Example 6)

This variable can be used to call a Macro based on Input State

Macro Start IN&15 Repeat 0

Will start macros 1, 2, 3, … 15 based upon Inputs 1 to 4.

	NOTE
	*This variable may not be used where a <variable> is required by a PLS, Link or Macro step as it is stored as an integer and not a floating-point decimal.*

INX - Bit representation of **extended** inputs, which are active. Represent extended input 1-16 (Read Only)

Inputs are read in binary from right > left on I/O Screen.

Example 1: If INX&644=256 Then Go To HOME
If INX&644=132 Then Go START

Input Number	Etc.	26	25	24	23	22	21	20	19	18	17
Extended I/O #	Etc.	10	9	8	7	6	5	4	3	2	1
INX Value	Etc.	512	256	128	64	32	16	8	4	2	1

If Inputs 1, 5 and 6 are being tested IN&49

Active Inputs

INX&644=0	Extended Input 19, 24 & 26 all Off
INX&644=4	Extended Input 19
INX&644=128	Extended Input 24
INX&644=132	Extended Input 19 & 24
INX&644=512	Extended Input 26
INX&644=516	Extended Input 19 & 26
INX&644=640	Extended Input 24 & 26
INX&644=644	Extended Input 19, 24 & 26 all On

	NOTE
	*The Extended I/O when used in a program command are Inputs (17-32) and Outputs (9-16). Example: If **Input 19 Then GoTo Top** represents Extended Input 19 (INX=4).*

	NOTE
	*This variable may not be used where a <variable> is required by a PLS, Link or Macro step as it is stored as an integer and not a floating-point decimal.*

IOCW - The "*IO Configuration Word*" contains the bit values of the I/O configuration set in the I/O Wizard (***Wizard / I/O Configuration…***). Each configuration checked in the I/O Wizard sets a bit. The IOCW variable is equal to the sum of all the bits.

BDx-34xx Series Servo Digital and Intelligent Drive:

1 = Inputs 5 and 6 are QUAD ENCODER inputs.
2 = Overtravel LIMITS on Inputs 3 (+) and 4 (-)
4 = JOG Inputs on Inputs 3 (+) and 4 (-)
8 = Overtravel LIMITS on Inputs 5 (+) and 6 (-)
16 = P/PI switch on Input 7 (input ON = no Integral gain)
32 = PAUSE/RESUME on Input 7 (input ON = pause)
64 = Output 3 is "Programmable" (not CURRENT FOLDBACK)
128 = Output 1 functions as BRAKE control
256 = Position Loop period is 4ms
512 = Invert Master Position counting.
1024 = Invert Target Generation in Master Tracking
2048 = Restrict Master (+) [Anti Backup] for Electronic Gearing

IN0= - Force Standard Input bits off based on binary input values.

Example: IN0=2 (Forces Input 2 off)
IN0=10 (Force Inputs 2 and 4 off)
IN0=0 (Turns off all Forced Inputs)

INX0= - Force Extended Input bits off based on binary input values.

Example: INX0=8192 (Forces Extended Input 29 off)
INX0=10 (Force Extended Inputs 18 & 20 off)
INX0=0 (Turns off all Extended Inputs Forced on)

	NOTE
	*The Extended I/O when used in a program command are Inputs (17-32) and Outputs (9-16). Example: If **Input 19 Then GoTo Top** represents Extended Input 19 (INX=4).*

IN1= - Forces Standard Inputs bits on based on binary input values.

Example: IN1=2 (Forces Standard Input 2 on)
IN1=10 (Force Standard Inputs 2 and 4 on)
IN1=0 (Turns off all forced on inputs)

	NOTE
	*When using an Extended I/O board the inputs are numbered from 17 – 32. The extended outputs are numbered 9-16.*

INX1= - Force Input bits on based on binary input values.

Example: INX1=2 (Forces Extended Input 2 on)
INX1=10 (Force Extended Inputs 18 & 20 on)
INX1=0 (Turns off all forced on inputs)

	NOTE
	*The Extended I/O when used in a program command are Inputs (17-32) and Outputs (9-16). Example: If **Input 19 Then GoTo Top** represents Extended Input 19 (INX=4).*

OFF(#) - Function call, represents Input bits which are inactive. **Read Only**

Example: If OFF(2) then END
(If Input 2 is inactive jump to label END)

	NOTE
	*The Extended I/O when used in a program command are Inputs (17-32) and Outputs (9-16). Example: **If OFF(23) Then Top** represents Extended Input 6 (INX=64).*

ON(#) - Function call, represents Input bits which are active. **Read Only**

Example: If ON(2) then TOP
(If Input 2 is active jump to label TOP)

Allows the user to check the status of individual inputs, while ignoring the status of the rest.

	NOTE
	*The Extended I/O when used in a program command are Inputs (17-32) and Outputs (9-16). Example:* ***If ON(30) Then Top*** *represents Extended Input 14 (INX=8192).*

OUT0 - Force Standard Output bit off. (Binary Representation)

Example: OUT0=4 (Forces Standard Output 3 OFF)

OUTX0 - Force Extended Output bit off. (Binary Representation)

Example: OUTX0=24 (Forces Extended Outputs 12 & 13 OFF)

	NOTE
	*The Extended I/O when used in a program command are Inputs (17-32) and Outputs (9-16). Example:* ***Set Output 12*** *represents Extended Output 12 (OUTX=8).*

OUT1 - Force Standard Output bit on. (Binary Representation)

Example: OUT1=6 (Forces Outputs 2 & 3 ON)

OUTX1 - Force Extended Output bit on. (Binary Representation)

Example: OUTX1=48 (Forces Extended Outputs 13 & 14 ON)

	NOTE
	*The Extended I/O when used in a program command are Inputs (17-32) and Outputs (9-16). Example:* ***Set Output 12*** *represents Extended Output 12 (OUTX=8).*

OUT - Bit representation of Standard outputs, which are active. (Read Only)

Standard Outputs are read in binary from right to left.

Example:

Output No.	4	3	2	1
Value	8	4	2	1

If Outputs 1 & 3 are active OUT=5

	NOTE
	*The Extended I/O when used in a program command are Inputs (17-32) and Outputs (9-16). Example:* ***Set Output 12*** *represents Extended Output 12 (OUTX=8).*

OUTX - Bit representation of Extended Outputs, which are active. (Read Only)

Outputs are read in binary from right to left.

Example:

Output Number	24	23	22	21	20	19	18	17
Extended Output No.	8	7	6	5	4	3	2	1
OUTX Value	64	48	32	16	8	4	2	1

If Extended Outputs 12 & 13 are active OUTX=24

	NOTE
	*The Extended I/O when used in a program command are Inputs (17-32) and Outputs (9-16). Example:* ***Set Output 12*** *represents Extended Output 4 (OUTX=8).*

OUTON - A function which returns True (-1) when the corresponding Output <n> is ON and returns a zero value when OFF.

4.5 Electronic Gearing

BDSP - This variable is set in conjunction with the "Wait for Rise/Fall of Both" command and is equal to the number of master counts between the inputs becoming rising/falling or the maximum displacement set in the command.

Example:

```
Macro 2
…Wait Input Rise 5
…Gear at 20000/30000 in 1000
…Wait for Fall of Both Inputs 6/7 Max 12000
…If BDSP>=12000 Then Chain 3
… Gear at Zero/30000 in 500
Macro 3
…Gear at Zero/30000 in 500
…Set Output1
…COUNT=COUNT*ONE+ONE
Macro End
Macro Start 2 Repat 0
End Program (0)
```

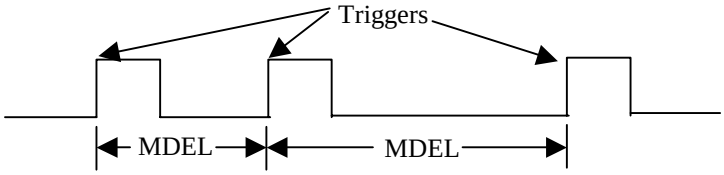
In the above program, Macro 3 waits for Input (5) then begins a Electronic Gearing at 2/3 ratio within 1000 master counts. The macro then waits for Inputs (6) and (7) to fall within the next 12000 master counts. If this condition is met, the macro continues. If both inputs do not fall within 12000 master counts the program jumps to Macro 3, sets output (2) adds one to the fault counter and continues.

MAST - Holds the present master encoder input value scaled in counts. This variable rolls over at "**MMC**" counts but is cleared at the point the controlling input transitions, when utilizing "position capture". The variable is also cleared when a begin "Master Tracking" command is issued. (**Read Only**)

	NOTE
	*This variable may not be used where a <variable> is required by a PLS, Link or Macro step as it is stored as an integer and not a floating-point decimal. (See '**PLSM**' the floating point version of MAST)*

MBR - The macro currently being run. It can be displaced in the monitor window as a diagnostic tool. (-1 = None)

MDEL - Holds the captured master displacement during High-Speed transitions. The variable is captured when using the "***Arm and Capture***" command and is the number of master counts "***MAST***" between triggers.



	NOTE
	*This variable may not be used where a <variable> is required by a PLS, Link or Macro step as it is stored as an integer and not a floating-point decimal.*

MENC - Master Encoder Counts received through the high speed quad encoder inputs. Unlike TMC, which only keeps track of the encoder input while electronically gearing, this variable keeps count of the secondary encoder input in any mode except step/dir or step/step. The MENC can be cleared at any time.

Example:

```
        Call GET_DIST
        Index Move V1 at 10 system-revs/sec
        ……
GET_DIST
        V1=MENC
        Clear MENC
        V1=V1/4000
                Return
```

In the above example, the customer is using encoder pulses from a PLC to set the move distance. The encoder pulse sent to the high speed inputs are counted in the background. Before the move command is issued the variable V1 is set equal to MENC then scaled by 4000.

MMC - Master Modulo Constant is the variable, which sets the "roll over" point for the MAST variable. It is initialized as 1,000,000 counts at start-up.

MPAI - This variable captures the number of "Master" pulses received during the last "**Gear At In**" command.

MPFI - This variable captures the number of "Master" pulses received during the last "**Gear For In**" command.

MSP - The virtual step pointer for the macro being run. It is used as a program debugging tool. (0 = first step of macro)

PLSM - Is the floating-point version of '***MAST***', although still scaled in counts it can be used wherever a <variable> is required by a PLS, Link or Macro step. '**PLSM**' rolls over at '**MMC**'

PLSS - Is the floating-point version of '***SLAV***', although still scaled in counts it can be used wherever a <variable> is required by a PLS, Link or Macro step. '**PLSS**' rolls over at '**SMC**'

PLSX - Is the floating-point version of '***SMOD***' (Slave Modulo), although still scaled in counts it can be used wherever a <variable> is required by a PLS, Link or Macro step.

SLAV - Holds the present target slave position value and is scaled in counts. The variable rollover coincides with the feedback resolution "**EPPR**". (**Read Only**)

	NOTE
	*This variable may not be used where a <variable> is required by a PLS, Link or Macro step as it is stored as an integer and not a floating-point decimal. (See '**PLSS**' the floating point version of SLAV)*

SMC - The Slave Modulo Constant is the constant used to create the system variable SMOD and is scaled in counts. Essentially, it is the number feedback counts per system revolution that is set by the customer. This variable is used with the ***"Phase Adjust…"*** commands.

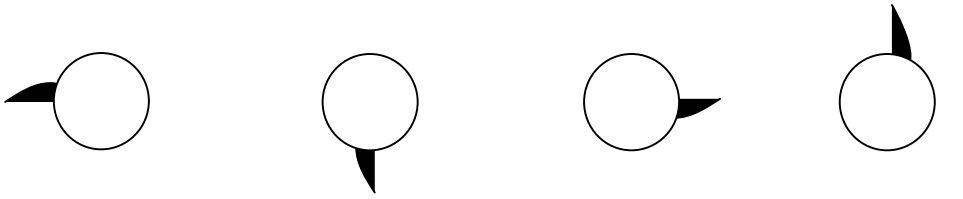
Example: A rotating knife application where the motor has a 1024 line encoder connected to cutting knife through a 2:1 pulley. Each rotation of the motor generates 4096 feedback pulses. Each rotation of the knife requires 2 motor revolutions or generates 8192 feedback pulses.

For the above example SMC would be set to 8192 counts per knife rotation. (SMC = 8192)

SMOD - The Slave Modulo holds the slave feedback position based on the "***SMC***" (Slave Modulo Constant). This value varies between 0 and SMC and indicates the position of rotation of the slave axis.

Example: A rotating knife application where the motor has a 1024 line encoder connected to cutting knife through a 2:1 pulley. Each rotation of the motor generates 4096 feedback pulses. Each rotation of the knife requires 2 motor revolutions or SMC = 8192 counts.

As the blade rotates SMOD varies between 0 < SMOD < 8192 (SMC)



SMOD = 0 (PLSX) | SMOD = 2048 (PLSX) | SMOD = 4096 (PLSX) | SMOD = 6144 (PLSX)

	NOTE
	*This variable may not be used where a <variable> is required by a PLS, Link or Macro step as it is stored as an integer and not a floating-point decimal. (See '**PLSX**' the floating point version of SMOD)*

SMRK - The Slave Mark variable is captured by the trigger when using the "***Arm and Capture …***" commands and equals "**SPH**Z" (Slave Phase Adjust) - SMOD (Slave Modulo). This variable is used in calculating phase adjustments.

Example: For this example assume SMC = 8192.
(0 < SMRK < 8191)

If SMOD = 60 counts and the slave phase adjust SPHZ = 357.
SMRK = 297

If SMOD = 1460 counts and the slave phase adjust SPHZ = 357.
SMRK = 7089

	NOTE
	*This variable may not be used where a <variable> is required by a PLS, Link or Macro step as it is stored as an integer and not a floating-point decimal.*

SPHZ - The Phase Slave Adjustment variable is used during high-speed capture to create the variable SMRK. ***SPHZ*** can be manually set to adjust the number of slave counts, which SMOD is subtracted from to create SMRK. System variable SMRK is used to calculate phase adjustment with the "***…Phase Adjust via Slave***" command.

Example: When starting a label application, if the label is not lined up properly, the operator can adjust the "**SPHZ**" to adjust the offset of the label.

SPAI - This variable captures the number of "Slave" pulses commanded during the last "**Gear At In**" command.

SPFI - This variable captures the number of "Master" pulses generated during the last "**Gear For In**" command.

TMC - A general-purpose counter that maintains the number of master pulses received since it was last cleared.

TMCP - Stores the previous value of TMC when the "Clear TMC on…" command is executed.

TSC - A general-purpose counter that maintains the number slave pulses generated since it was last cleared.

WLIM - Is the maximum wait limit, in slave counts, that the program will pause at a "Wait on Input" command before proceeding with program execution. If the input does not become active before TSC (Total Slave Counts) => WLIM, the input will be internally triggered and the program will continue. The TSC counter does not begin until the "Wait on Input" command is issued.

Example:

Macro 2
…When Input 1 Falls Clear TSC
…Wait Input 1 Fall
…Gear at 20000/30000 in 600
…WLIM=10000*1+ZERO
…Wait Input 2 Fall
…Compare If TSC>WLIM Then Chain 4
…Gear at Zero/30000 in 300

In the above example, the drive begins electronic gearing when Input 1 falls. A WLIM limit of 10000 master counts is set. The drive then waits for input 2 to fall. If the slave counter reaches 10000 counts before the input becomes active, the program will continue. A comparison is done to determine if the condition was met.

	NOTE
	***WLIM** is reset to zero at the start of each macro to prevent it from mistakenly being executed in another macro.*

5. Programming Commands

This section covers the programming commands available for the Intelligent drive. The commands are divided into (8) major categories including Macro, Utility, Gearing, Motion, Go To, I/O and Spline commands.

5.1 <empty>

This command leaves a blank line in the program.

5.2 <IML>

This command allows the programmer to enter the IML (Intelligent Motion Language) commands directly into a motion program. The IML commands are limited to those commands listed under the "IML" heading. The advantage of the IML command is that multiple commands can be put on a single command line and the commands take up less memory. This is important for programs, which approach maximum length or if the program library is close to its memory limit. (See IML document located on the **Tec Tools** disk.)

Example:

Set Acceleration Rate to 400 system-rev/sec^2
Index of -32.5 system-rev At 10 system-rev/sec
Wait for In Position

The above 3 line example is the same as the single line below.

A400:M-32.5,10:W

	NOTE
	Colons ":"are used to separate commands while <values> are separated by commas "," and the commands are case sensitive. *Example:* **A<value>:M<value>,<value>:W**

5.3 Macro Commands

Macros are a set of pre-computed, instantly accessible commands to produce complex motion. Since macros are pre-computed there is no program interpretation required providing execution times in a matter milliseconds.

- Within a macro, step-to-step transition is predicable and repeatable. Execution of a macro is performed during the position-loop update period and is guaranteed to complete within one position loop update. (2 msec)
- Macros are called by macro number, allowing the use of discrete inputs to select a macro within a program.
- Intelligent Drive can store a maximum of 6 (numbered 0-5) with each Macro being up to 16 steps long (steps 0-15).
- Macros are stored in RAM (Volatile) memory and need to be read each time the drive is reset or powered down.
- Macros can be embedded up to 16 levels deep.

With-in a motion program, Macros are defined using the “Macro” and “Macro End” steps. Macros are numbered from zero up to five. Macros can be defined and/or redefined in any order.

Example1:

```
Macro 1
…
…
Macro End
Macro 2
…
…
Macro End
Macro 0
…
…
Macro End
```

Example 2:

```
Macro 0
…
…
Macro 1
…
…
Macro 4
…
…
Macro 2
…
…
Macro End
```

Example 3:

```
Macro 5
…
…
Macro End
…
…
Macro End
(Program Steps)
Macro 5
…
…
 Macro End
```

5.3.1 …Absolute Move

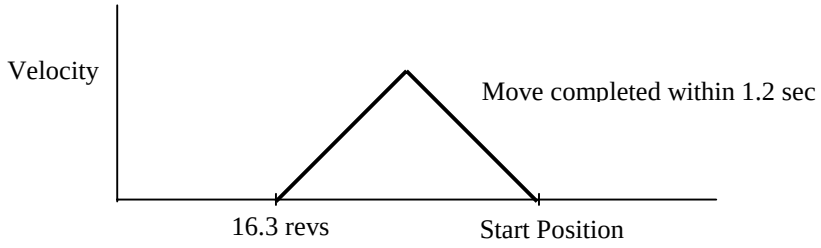
Execution Time: 1 MLP (Position Loop Updates)

This macro step executes a move command to an absolute position within a fixed time interval. The position is defined in revolutions while the move time is entered in milliseconds. Both the position and time can be defined as a constant or variable. The move profile is triangular to utilize the least amount of motor torque.

Example:

```
Macro 0
… Absolute Macro Move to 16.3 revs in 1200 msec
Macro End

Macro Start 0 Repeat 1
Wait macro Complete
```



In the above example, the Macro is read into RAM then called to execute. When Macro 0 is executed, the move command will begin execution within 1 MLP and repeat only once.

	NOTE
	*The move is always in system revs without the Translation Ratio (TR) being utilized. The "Wait for In Position" is implied for macro moves.*

5.3.2 …Add

Execution Time: 1 MLP (Position Loop Updates)

This macro step adds a constant to a specified variable.

Example:

Count = Zero
Macro 4
…
Add 1 to COUNT (Add –1 to COUNT to Decrement)
Macro End

In the above example, each time the macro is run the COUNT increases by 1.

5.3.3 …Arm Logic Input (0->1)

Execution Time: 1 MLP (Position Loop Updates)

This macro step arms the specified input to capture specific system variables at the "off" to "on" transition of the specified input. It is used in conjunction with the "Phase Adjust" and "Phase Delay" commands.

Captured Variables:

MDEL - The difference between the last captured master position and the new captured master position.

SMRK - Equals the value of "SPHZ – SMOD" (Slave Phase Adjust minus Slave Feedback Position Modulo)

Input Codes	***Defined Input*** *(I-Servo)*
A	Discrete Input 7
B	<reserved>
C	<reserved>
D	<reserved>
E	<reserved>

Example: Assume SMC=4096, SMOD=200 and SPHZ=328

Macro 3
…Arm and Capture (0->1) via Logic Input A (*Captures **SMRK**)*
…Wait HS Capture, Limit to 4096, Action P (*Waits for capture)*
…Phase Adjust via Slave 5000 (Executes "Gear for...in...")
Macro End

The above example macro captures the system variable **SMRK** (SPHZ-SMOD) when input 7 changes state from off to on. Once the position is captured the macro will Phase Adjust the Slave by 128 counts (SMRK=328 -200 in our example) within 5000 master counts.

5.3.4 …Arm Logic Input (1->0)

Execution Time: 1 MLP (Position Loop Updates)

This macro step arms the specified input to capture specific system variables at the "on" to "off" transition of the input. It is used in conjunction with the "Phase Adjust" and "Phase Delay" commands.

Captured Variables:

MDEL - The difference between the last captured master position and the new captured master position.

SMRK - Equals the value of "SPHZ – SMOD" (Slave Phase Adjust minus Slave Feedback Position Modulo)

Input Codes	***Defined Input***
A	Discrete Input 7
B	<reserved>
C	<reserved>
D	<reserved>
E	<reserved>

Example: Assume SMC=4096 and SPHZ=200

Macro 3
…Arm and Capture (0->1) via Logic Input A (*Captures **SMRK**)*
…Wait HS Capture, Limit to 4096, Action P (*Waits for capture)*
…Phase Adjust via Slave 5000 (Executes "Gear for...in...")
Macro End

The above example macro captures the system variable **SMRK** (SPHZ-SMOD) when input 7 changes state from off to on. Once the position is captured the macro will move "SMRK" counts within 5000 master counts.

5.3.5 …Call Macro

Execution Time: 1 MLP (Position Loop Updates)

This macro step calls the specified macro from within a macro. When the called macro is complete, it will return to the line following the "Call Macro" step in the original macro. The macro number may be specified as a constant or variable. The macro can also be called by "discrete inputs" using an input mask.

Example 1:

Macro 5
…Set Single Output 1
…Call Macro 4
…Clear Single Output1
Macro End
Macro Start 5 Repeat 8
Wait Macro Complete

The above example will start Macro 5, set Output1, execute Macro 4 then return to Macro 5 and clear Output 1. The process will repeat 8 times.

Example 2:

Macro 5
…Set Single Output 1
…Call Macro -3
…Clear Single Output1
Macro End
Macro Start 5 Repeat 3 times
Wait Macro Complete

The above example will start Macro 5, set Output 1 call and execute a macro based on inputs 1 and 2 then. After completing the called Macro the program control will return to Macro 5 and clear Output 1. The process will repeat 3 times.

Possible Macros	Active Inputs
Macro 0	Input 1 & 2 Off
Macro 1	Input 1
Macro 2	Inputs 2
Macro 3	Input 3 & 4

Input No.	7	6	5	4	3	2	1
Value	64	32	16	8	4	2	1
On/Off	N/A	N/A	0	1	1	N/A	N/A

(-3) masks the inputs (Same as IN&3 masks) Inputs 1 and 2
Values (1 + 2 = 3)

Based on the Input states of the table Example 2 would call Macro 3
(Inputs 1 & 2 Active = 1 + 2 = 3)

	NOTE
	*The call stack limit is 16 levels deep.*

5.3.6 …Chain Macro

Execution Time: 1 MLP (Position Loop Updates)

This macro step substitutes the execution of the active macro with a new macro. When this macro step is executed, macro execution shifts to the first step of the chained macro. The macro can also be chained by "discrete inputs" using an input mask.

- When the "***Chain Macro***" command is executed the "Call Macro" stack is cleared. The control will <u>not</u> return to the original macro but returns to the line below the "Macro Start" command in the main program.

- The repeat count that was specified in the "Macro Start" command is ignored, and the macro execution will cease when the next "Macro End" is reached.

Example 1:

Macro 5
… Absolute Macro Move to 8.4 revs in 900 msec
…Set Single Output 1
…Chain Macro 4
Macro End
Macro Start 5 Repeat 8
Wait Macro Complete

The above example will start Macro 5, make an absolute move to 8.4 revs, set Output1 then execute Macro 4 before returning to the line below the Macro Start command in the main program. The process will not repeat.

Example 2:

Macro 5
… Absolute Macro Move to 16.3 revs in 1200 msec
…Set Single Output 1
…Chain Macro -3
Macro End
Macro Start 5 Repeat 3
Wait Macro Complete

The above example will start Macro 5, make an absolute move to 8.4 revs, set Output 1 chain and execute the Macro 3 based on inputs 1 and 2 then. After completing the chained Macro the program control will return to the line below "Macro Start" in the main program. The process will not repeat.

Input No.	7	6	5	4	3	2	1
Value	64	32	16	8	4	2	1
On/Off	N/A	N/A	1	0	1	N/A	N/A

(-3) masks the inputs (Same as IN&3 masks) Inputs 1 and 2
Values (4 + 8 + 16 = 28)

Based on the table above Example 2 would call Macro 3
(Inputs 1 & 2 Active = 1 + 2 = 3)

5.3.7 …Clear Single Output

Execution Time: 1 MLP (Position Loop Updates)

This macro step clears a single output and must be entered as a constant. The counterpart to this command is the "Set Single Output"

	NOTE
	*The Extended I/O when used in a program command are Inputs (17-32) and Outputs (9-16). Example:* ***…Clear Single Output 10,*** *represents Extended Output 10 (OUTX=2).*

	NOTE
	*Due to software conflicts, the same output cannot be controlled in both the macro and main program or PLC.*

5.3.8 …Clear Variable

Execution Time: 1 MLP (Position Loop Updates)

This macro step clears a specified variable from it's current value to zero. (The variable can not be a Read Only variable)

5.3.9 …Compare If and Call

Execution Time: 1 MLP (Position Loop Updates)

This macro step makes a comparison of an argument if false, continue through the macro but if true will call the specified macro. Once the called macro is complete, program execution continues in the original macro. The argument can be a variable or constant.

Example:

Macro 3
…Wait Input 7 Fall Clear TMC
…Wait Input Rise 7
…Gear At 0/100000
…Gear At 50000/100000
…Wait Input Fall 6
…If TMC>120000 Then Call 4
…Gear At 0/100000
Macro 4
…Set Single Output 2
Macro End
Connect Master Encoder
Begin Master Position Tracking
Call Macro 3
Wait Macro Complete

The above example waits for Input 7 then sets the gear ratio between the master slave to 5000/10000 (1/2). When input 6 becomes active a comparison is done and if TMC is greater than 120,000, Macro 4 is called and Output 2 is set. Program execution then returns to Macro 3 and the gear ratio will become 0 stopping the slave axis.

	NOTE
	*The call stack limit is 16 levels deep.*

5.3.10 …Compare If and Chain

Execution Time: 1 MLP (Position Loop Updates)

This macro step makes a comparison of an argument if false, continue through the macro but if true will chain the specified macro. Once the chained macro is complete, program execution does not return to the original macro. The argument can be a variable or constant.

Example:

```
Macro 3
…Wait Input 7 Fall Clear TMC
…Wait Input Rise 7
…Gear At 0/100000
…Gear At 50000/100000
…Wait Input Fall 6
…If TMC>120000 Then Chain 4
…Gear At 0/100000
Macro 4
…Set Single Output 2
…Gear At 0/100000
Macro End
Connect Master Encoder
Begin Master Position Tracking
Call Macro 3
Wait Macro Complete
End Program (0)
```

The above example waits for Input 7 then sets the gear ratio between the master slave to 5000/10000 (1/2). When input 6 becomes active a comparison is done and if TMC is greater than 120,000, Macro 4 is Chained, Output 2 set and gear ratio set to zero.

5.3.11 …Connect Master Encoder

Execution Time: 1 MLP (Position Loop Updates)

This macro step connects the master position encoder input to the drive effectively reversing the effect of the "Disconnect" macro step. This macro step is identical to the "Connect" command under Gearing.

Example: In the example below, an arm is pulling material fed between two rollers, which prevents the material from stretching. The secondary encoder on the arm has a 2/1 gear ratio to the rollers. As the arm reaches the end of travel, Input 6 becomes active which ends master position tracking. This prevents material from being back fed through the rollers as the arm retracts. **(Also See Example 8)**

Macro 3
…Wait Input Rise 5
…Connect
…Gear At 20000/10000 in 300
…Wait Input Rise 6
…Disconnect
Macro End
Begin Master Position Tracking
Macro Start 3 Repeat 0
Wait Macro Complete

The above macro will connects the master position encoder on the rise of Input 5 so that the drive will follow the secondary encoder on the arm. When Input 6 becomes active, the drive will disconnect the master position encoder, ignoring any pulse transitions in the secondary encoder.

	NOTE
	*By default the master position encoder input is **Connected** when the Master Position Tracking is initiated.*

5.3.12 …Disconnect Master Encoder

Execution Time: 1 MLP (Position Loop Updates)

This macro step is used to disconnect the master position encoder input by ignoring any pulse. This macro step is identical to the "Disconnect" command under Gearing.

Example: In the example below, an arm is pulling material fed between two rollers, which prevents the material from stretching. The secondary encoder on the arm has a 2/1 gear ratio to the rollers. At the end travel for the arm, Input 6 becomes active which ends master position tracking. This prevents material from being back fed through the rollers as the arm retracts. **(Also See Example 8)**

Macro 3
…Wait Input Rise 5
…Connect
…Gear At 20000/10000 in 300
…Wait Input Rise 6
…Disconnect
Macro End
Begin Master Position Tracking
Start Macro 3 Repeat 0
Wait Macro Complete

The above macro will connects the master position encoder on the rise of Input 5 so that the drive will follow the secondary encoder on the arm. When Input 6 becomes active, the drive will disconnect the master position encoder, ignoring any pulse transitions in the secondary encoder.

5.3.13 …Gear At

Execution Time: 1 MLP (Position Loop Updates)

This macro step is used to change the "Master Tracking" gear ratio. The numerator or denominator may be specified as a constant or a variable. The relationship for the ratio is:

Slave Count = (EPPR/SPPR)(Numerator x Master Count)/Denominator

Example:

Begin Master Position Tracking
Macro 3
…Connect
…Gear At 50000/100000
…
…
Macro End

The above example sets the gear ratio between the master slave to 5000/10000 (1/2). The slave will move 1 rev for every 2 revs of the master.

	NOTE
	*It is recommended that the Electronic Gear Ratio numerator and denominator are as large as possible to provide a smooth transition between gear ratios. (Numerator and Denominator < 1,000,000)* *"Gear At 2/3 in 1000" will transition in one master count.* *"Gear At 20000/30000 in 10000" will transition within 10000 master counts.*

5.3.14 …Gear At In

Execution Time: 1 MLP (Position Loop Updates)

This macro step is used to change the "Master Tracking" gearing ratio within a specified master displacement. The numerator, denominator or the master displacement may be specified as a constant or a variable.

Slave Count = (EPPR/SPPR)(Numerator x Master Count)/Denominator

Example: (Also See Example 8)

```
Begin Master Position Tracking
Macro 3
…Gear At 3000/10000
…Wait Input Rise 5
…Gear At 5000/10000 In 1000
Macro End
```

The above example sets the master/slave gear ratio to 3/10. After Input 5 becomes active the gear ratio will transition from 3000/10000 to 5000/10000 within 1000 master encoder pulses.

	NOTE
	*It is recommended that the Electronic Gear Ratio numerator and denominator are as large as possible to provide a smooth transition between gear ratios. (Numerator and Denominator < 1,000,000)* *"Gear At 2/3 in 1000" will transition in one master count.* *"Gear At 20000/30000 in 10000" will transition within 10000 master counts.*

5.3.15 …Gear For In

Execution Time: 1 MLP (Position Loop Updates)

This macro step adjusts the "Master Tracking" gearing ratio in such a way that the <*feedback displacement*> is taken up within the <*master displacement*>. The <*feedback displacement*> is executed on top of the electronic gear ratio. Both the <*feedback displacement*> and <*master displacement*> can be a constant or variable.

Example:

Macro 3
…Gear At 10000/5000 *Sets Gear Ratio at 2/1*
…Gear for -100 In 1000
Macro End

The above example sets the master/slave gear ratio to 2/1. The macro then makes a correction of –100 feedback counts over the next 1000 master pulses. When master displacement "**MAST**" changes by 1000 counts the feedback displacement "**SLAV**" will change by 1900 counts. (2 x Master Position –100)

	NOTE
	*The maximum feedback displacement is limited to ½ the master displacement.* ***Gear For <feedback displacement> in <master displacement>***

5.3.16 …Master Virtual Rate/Limit

Execution Time: 1 MLP (Position Loop Updates)

This macro step effectively simulates a secondary encoder input allowing the drive to operate in "Tracking Mode" without a secondary encoder connected. A displacement <*rate*> by which the master position will be "moved" every millisecond is set along with the maximum displacement the master will be allowed to move when the master encoder input is "disconnected". The system must be tracking the master encoder and the secondary encoder must be "disconnected" for this command to execute. Both the <*rate*> and <*limit*> must be constants.

Example:

Begin Master Position Tracking
Macro 2
…Disconnect Master Encoder
…Master Virtual Rate 36 Limit 1000000
Macro End

The example above sets the Master Virtual Rate to 36 pulses/msec (36000 pulses/sec) with a maximum displacement of 1,000,000 master encoder counts. Negative values are allowed to invert the direction of master counts.

	NOTE
	*The <rate> on the Intelligent Drive has a minimum resolution of 24 counts due to its 2 millisecond update period.* *If the <limit> is set to zero the Master Virtual Rate will run forever.*

5.3.17 …Offset Slave…

Execution Time: 1 MLP (Position Loop Updates)

This macro command is used to make corrections for rounding errors when the electronic gear ratio is not an exact ratio. (Example 1000/3000) The slave count will be adjusted by the Offset Slave Value – TSC.

Example:	**Assume TSC=100**	**Result**
	Offset Slave 90-TSC	Slave move 10 counts back (90-100)
	Offset Slave 100-TSC	Slave does not move (100-100)
	Offset Slave 110-TSC	Slave move 10 counts forward

```
Macro  1
…Wait Input Rise 7
… Gear At ZERO/COUNT in M1
…Offset Slave by 815 - TSC
…Clear TSC
… Gear At SCOUNT/COUNT in ACCEL
… Gear At ZERO/COUNT in DECEL
Macro End
```

The slave motor is expected to move 815 counts. Due to the gear ratio and the accel/decel distances, we have the chance for missing slave counts during each cycle of the program. Since it is essential for the slave motor to move 815 counts per cycle, we capture the value for TSC to know the actual move distance; we add this difference to the slave motion at the beginning of the next geared motion thus eliminating creep error.

	NOTE
	*The user program is responsible for clearing TSC as required to cause the "Offset Slave" to function properly.*

5.3.18 …Phase Adjust via Master

Execution Time: 2 MLP (Position Loop Updates)

This macro step is used in Electronic Gearing when the product coming down the line is staggered. It generates and executes the appropriate "Gear For… In …" step such that the position captured in 'MDEL' variable is subtracted from the forced <desired value> and used as the feedback displacement. Both the <master displacement> and <desired value> can be a variable or constant. The "Phase Adjust via Master" adjusts the master encoder counts over a set displacement based on the spacing of the triggers.

Example: The product coming down the line is staggered.

For this Example Assume
At Trigger 2 MDEL = **4160**
At Trigger 3 MDEL = **5820**
Begin Master Tracking
Macro 3
…Arm and Capture (0->1) via Logic Input A (*Captures **MDEL**)*
…Wait HS Capture, Limit to 4096, Action P (*Sets Max Wait)*
…Phase Adjust via Master in 1200 Force
Macro 1
…Connect Master Encoder
…Gear at 10000/20000 in 100
Macro End
Macro Start 1 Repeat 1
Wait Macro Complete
Macro Start 3 Repeat 0
Wait Macro Complete

Trigger 2 The drive will add 840 (5000-4160) additional master encoder counts over the next 1200 master counts.

Trigger 3 The drive will ignore 820 (5000-5820) master encoder counts over the next 1200 master counts.

In the above example, the macro captures the system variable **MDEL** when input 7 changes state from off to on. Once the position is captured, the macro will either add or ignore master position counts to adjust the slave axis to product coming down the line.

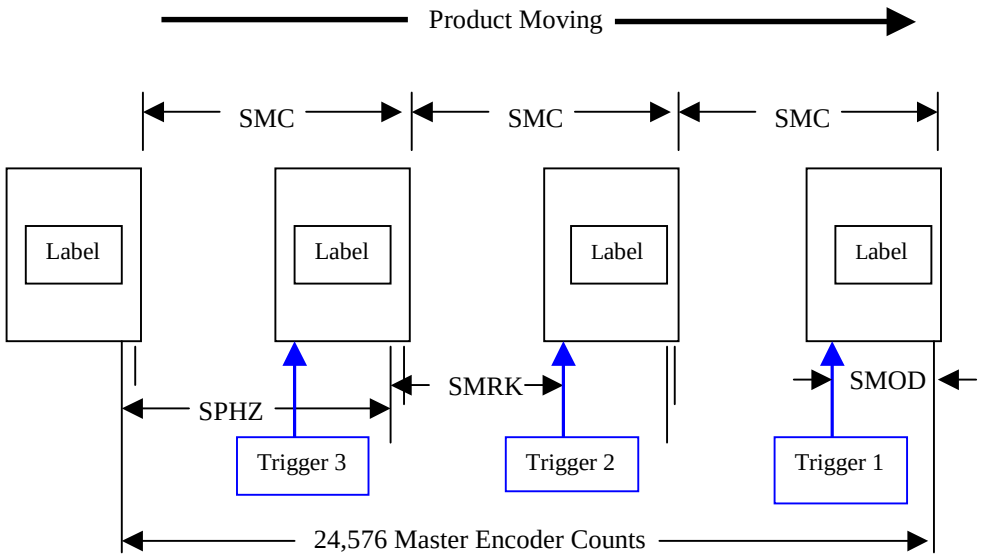
5.3.19 …Phase Adjust via Slave

Execution Time: 1 MLP (Position Loop Updates)

This macro step is used in Electronic Gearing to create an offset between the master and slave axes by adding slave counts. It generates and executes the appropriate "Gear For… in …" step such that the feedback (slave target) position offset captured into the '**SMRK**' variable is used as the feedback displacement in the "Gear For… in …" step. Essentially this command shifts the slave axis and adjusts to the evenly spaced product coming down the line. The <master displacement> can be a variable or a constant.

Example: The application is applying labels to bottles which are evenly spaced on a conveyor. The servo motor has a 1024 line encoder and there are 2 motor rotations between labels (SMC=8192). The "Phase adjust via Slave" command is used to center the label on the bottle via the 'SPHZ' variable. The adjustment will be made within 24576 master encoder counts (3 Bottles).

Begin Master Position Tracking
Macro 3
…Arm and Capture (0->1) via Logic Input A (*Captures **SMRK**)*
…Wait HS Capture, Limit to 10000, Action P (*Sets Max Wait)* …Phase Adjust via Slave 24576 (E*xecutes "Gear for...in..."*)
Macro 1
…Connect Master Encoder
…Gear at 10000/20000 in 100
Macro End
Macro Start 1 Repeat 1
Wait Macro Complete
Macro Start 3 Repeat 0
Wait Macro Complete



Simplified:

SMC – Number of counts for one cycle.

SMOD – Ranges from 0 to SMC and is the actual position in counts captured at the trigger.

SPHZ – The desired position in counts when trigger is received.

SMRK– The correction in counts. (SPHZ-SMOD) is made as a "Gear At In" function.

In the above example, the macro captures the system variable **SMRK** (SPHZ-SMOD) when input 7 is triggered. Once the position is captured the macro will Phase Adjust the Slave by '**SPHZ**' counts within 24576 master counts. The labels are centered on the bottles after the phase adjustment is complete.

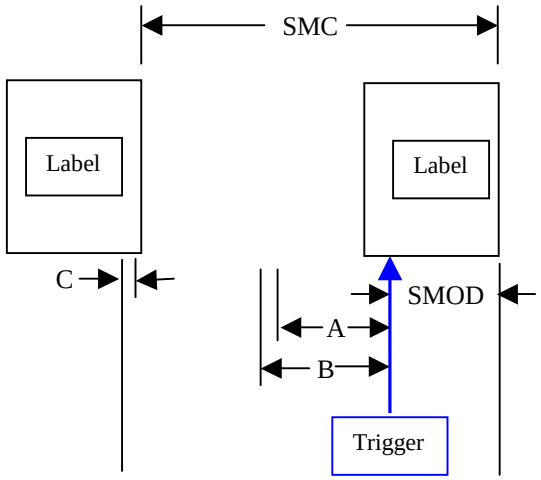
	NOTE
	Macro Execution "pauses" until the required adjustment is complete.

5.3.20 …Phase Delay via Master

Execution Time: 1 MLP (Position Loop Updates)

This macro step delays the calculated master displacement, essentially creating an offset between the master and slave by adjusting the master count. The calculated displacement is computed by taking the specified <displacement> and subtracting the amount the master has traveled between receiving the "Arm" input and the execution of the command.

Example: The application is applying labels to bottles, which are not evenly spaced on a conveyor. The servo motor has a 1024 line encoder and there are 2 motor rotations between labels (SMC=8192). The "Phase Delay via Master" command is used to center the label on the bottle by adding or subtracting master position counts.



Begin Mater Position Tracking
Macro 3
…Arm and Capture (0->1) via Logic Input A (*Captures **SMRK***)
…Wait HS Capture, Limit to 10000, Action P (*Sets Max Wait*)
…Phase Delay via Master: Target 800
Macro 1
…Connect Master Encoder
…Gear at 10000/20000 in 100
Macro End
Macro Start 1 Repeat 1
Wait Macro Complete
Macro Start 3 Repeat 0
Wait Macro Complete

"A" - Displacement, in master encoder counts, between the trigger in the "Arm and Capture" command and the time it took to execute the "Phase Delay via Master" command.

"B" - The Target <displacement> specified in the "Phase Delay via Master" command. (In above example Target Displacement = 800 master encoder counts)

"C" - The difference between Target Displacement and the distance the master encoder traveled between the trigger capture and execution of the "Phase Delay via Master" command. (C = B – A = Final Offset Displacement)

In the above example, the macro captures the master position when input 7 is triggered. Once the position is captured the macro will Phase Adjust the Master by specified <displacement> minus the amount the master traveled between the trigger input and execution of ***"Phase Delay via Master***". After the Master phase adjustment is complete, the labels are centered on the bottles.

5.3.21 …Pulse Delay

Execution Time: 1 MLP (Position Loop Updates)

This macro step delays for a specified number of master pulses to occur before continuing through the macro. Once the specified number of master pulses are received, the macro execution continues.

Example:

Macro 2
…Gear At 12000/60000 In 3000
…Pulse Delay 300
…Set Output 2
Macro End

In the above example, the macro sets an electronic gear ratio of 1:5, which comes up to speed within 3000 master counts. The macro will then wait 300 pulses before setting output 2.

5.3.22 …Quit Macro

Execution Time: 1 MLP (Position Loop Updates)

This macro step ends the execution of the active macro and clears the macro stack. Any chained macros will be cleared and program execution returns to the main program. The "…Quit Macro" command has no effect on motion. The motor will complete any move in progress when the quit command is issued.

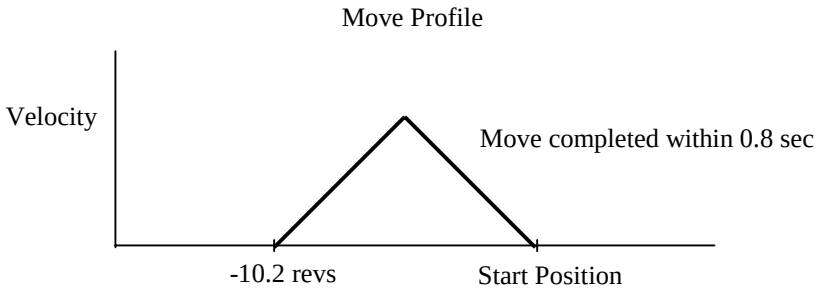
5.3.23 …Relative Move

Execution Time: 1 MLP (Position Loop Updates)

This macro step executes a relative within a specified time. The move profile is triangular to utilize the least amount of motor torque. The move command is divided into two fields. The first field specifies the<relative displacement> (in motor revolutions) while the second field <time limit> specifies maximum time allowed to complete the move in milliseconds. The displacement or time limit can be specified as a constant or variable.

Example:

Macro 6
…Relative Macro Move to -10.2 revs in 800 msec
Macro End
…etc
Macro Start 6 Repeat 1
Wait macro Complete



In the above example, the Macro is read into RAM then executed. The motor will begin to move 10.2 revs in the CW direction with the move being completed in 0.8 seconds.

	NOTE
	The Translation Ratio (TR) is not utilized. The "Wait for In Position" is implied for macro moves. *The position counter does not roll over. The maximum displacement limit is 2^{31} pulses of the encoder. When maximum displacement is reached, the motor will stop. (See "Zero Position" command)*

5.3.24 …Set

Execution Time: 1 MLP (Position Loop Updates)

This macro step sets a variable equal to a predefined math function within a macro. The math function **must** follow the following format:

X = ***A********B***+***C***

Where: ***X*** *is a <variable>*
A *is a <variable or constant>*
A *is a <variable or constant>*
C *is a <variable or constant>*

Example:

Macro 0
…Set POS1 to FPOS*0.5+1.5
Macro End

The example set custom variable POS1 = (Feedback Position x ½) + 1.5

5.3.25 …Set Single Output

Execution Time: 1 MLP (Position Loop Updates)

This macro step sets the specified output within a macro. The output number must be entered as a constant.

	NOTE
	*The Extended I/O when used in a program command are Inputs (17-32) and Outputs (9-16). Example:* ***…Set Single Output 12*** *represents Extended Input 12 (OUTX=8).*

	NOTE
	*Due to software conflicts, the same output cannot be controlled by both a macro and the main program. Therefore, one should utilize differing outputs within macros from those in the main program or a PLS.*

5.3.26 …Trigger Oscilloscope

Execution Time: 1 MLP (Position Loop Updates)

This macro command is used in conjunction with the on board Oscilloscope by triggering the acquisition of data which will be uploaded for display on the on-screen oscilloscope. The user can plot a number of variables vs. time, including feedback position, velocity, position error, etc. After opening scope make sure you set the begin plot on trigger capture.

Example: (Also See Examples 3, 4 & 8)

Macro 1
…Set Single Output 2
…Trigger Capture
…Relative Macro Move of 2 revs in 400 msec
Macro End

At the point the ***Trigger Capture*** command is executed the scope begins sampling the data to display on the screen.

	NOTE
	*Remove the **Trigger Capture** command from the program after debugging in order to speed up the program execution.*

5.3.27 …Wait

Execution Time: 1 MLP (Position Loop Updates)

This macro step is used to delay execution of the macro for the specified time. The time must be entered in milliseconds as a constant.

Example:

Macro 0
…Relative Macro Move to -10.2 revs in 800 msec
…Wait 500 milliseconds
Macro End

5.3.28 …Wait Fall Both

Execution Time: 1 MLP (Position Loop Updates)

This macro step is used in electronic gearing mode and waits for the fall (Edge Triggered) of two user specified inputs. The inputs do not have to be triggered in any specific order nor do they both have to be off at the same time. A maximum limit in master counts is set which, if reached will trip the event and continue in the macro.

Example:

```
Macro 2
…Wait Input Rise 5
…Gear at 20000/30000 in 1000
…Wait for Fall of Both Inputs 6/7 Max 12000
…If BDSP>=12000 Then Chain 3
…Gear at Zero/30000 in 500
Macro 3
…Gear at Zero/30000 in 500
…Set Output1
…COUNT=COUNT*ONE+ONE
Macro End
```

Macro 3 waits for Input (5) then begins Electronic Gearing at 2/3 ratio within 1000 master counts. The macro then waits for Inputs (6) and (7) to fall within the next 12000 master counts. If this condition is met, the macro continues. If both inputs do not fall within 12000 master counts the program jumps to Macro 3, sets output (2) adds one to the fault counter and goes back to the main program.

	NOTE
	*The Extended I/O when used in a program command are Inputs (17-32) and Outputs (9-16). Example:* ***…Wait for Fall of Both Inputs 6/17 Max 12000*** *represents Standard Input 6 (IN=32) and Extended Input 17 (INX=1).*

5.3.29 …Wait HS Capture

Execution Time: 1 MLP (Position Loop Updates)

This macro step is used in conjunction with the "***Arm and Capture***" command to set a maximum master displacement limit for the input to trigger. If the position capture happens within the maximum displacement the program continues. If the position capture does not occur within the maximum displacement limit the specified limit action <code> will be performed. The limit actions are specified as follows:

<code> Action

P	Process Phase Adjust as if 'trigger' occurred at limit
S	Skip any Phase Adjust until the next 'Arm' instruction
Q	Quit Macro
F	Fault Axis (Error Code 8004)

Example:

Macro 2
…Arm and Capture (0->1) via Logic Input B (*Captures **SMRK**)*
…Wait HS Capture, Limit to 10000, Action S (*Sets Max Wait)* …Phase Adjust via Master: Target 800
Macro End

In the above example the "*Arm and Capture*" command is executed. If Input 3 does not transition from off to on within 10,000 master encoder counts the macro will skip the "Phase Adjust" command until the next cycle. A "***Maximum Displacement Limit***" of zero will cause the macro to wait forever.

5.3.30 …Wait HS Logical Input Fall

Execution Time: 1 MLP (Position Loop Updates)

This macro step stops the execution of the macro until the specified High Speed Input makes an "on" to "off" transition

	NOTE
	*HSI 1 and HSI 2 are reserved for the secondary encoder when in Electronic Gearing, Quad or Distributed Feedback modes.*

5.3.31 …Wait HS Logical Input Rise

Execution Time: 1 MLP (Position Loop Updates)

This macro step stops the execution of the macro until the specified High Speed Input makes an "off" to "on" transition.

	NOTE
	*HSI 1 and HSI 2 are reserved for the secondary encoder when in Electronic Gearing, Quad or Distributed Feedback modes.*

5.3.32 …Wait Input Fall

Execution Time: 1 MLP (Position Loop Updates)

This macro step waits for the fall (Edge Triggered) of the specified input. Macro execution is stopped until this event occurs.

	NOTE
	*The Extended I/O when used in a program command are Inputs (17-32) and Outputs (9-16). Example:* ***…Wait Input Fall 23*** *represents Extended Input 23 (INX=64).*

5.3.33 …Wait Input Rise

Execution Time: 1 MLP (Position Loop Updates)

This macro step waits for the rise (Edge Triggered) of the specified input. Macro execution is stopped until this event occurs.

	NOTE
	*The Extended I/O when used in a program command are Inputs (17-32) and Outputs (9-16). Example:* ***…Wait Input Rise 23*** *represents Extended Input 23 (INX=64).*

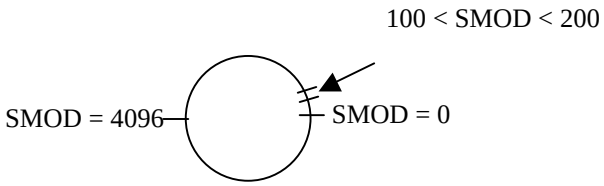
5.3.34 …Wait on SMOD Within

Execution Time: 1 MLP (Position Loop Updates)

This macro step waits until the system variable "SMOD" (Slave Modulo) lies between the specified positions. The SMOD limit is between 0 < SMOD < SMC (Slave Modulo Constant).

Example: A labeling application where there the motor has a 1024 line encoder connected to label head. Each rotation of the motor generates 4096 feedback pulses. Each application of a label requires 2 motor revolutions or SMC = 8192 counts. **(Also See Example 8)**

SMC=8192
Macro 1
…Wait on SMOD Within 100/200
…Set Single Output 1
…Arm and Capture (0->1) via Logic Input B



Too small of range can cause the drive to miss the window.

The above example macro waits for the slave modulo (SMOD) to lie between 100 to 200 counts before setting Output 1 and executing the "Arm and Capture" command.

	NOTE
	*If you want SMOD to lie outside the range reverse the range from high to low.* *Example: …Wait on SMOD Within 200/100*

5.3.35 …Wait Rise Both

Execution Time: 1 MLP (Position Loop Updates)

This macro step is used in electronic gearing mode and waits for the rise (Edge Triggered) of two user specified inputs. The inputs do not have to be triggered in any specific order nor do they both have to be on at the same time. A maximum limit in master counts is set which, if reached will trip the event and continue in the macro.

Example:

```
COUNT=ZERO
Macro 2
…Wait Input Rise 5
…Gear at 20000/30000 in 1000
…Wait for Rise of Both Inputs 6/7 Max 12000
…If CON=1 Then Chain 3
…Gear at Zero/30000 in 500
Macro 3
…Gear at Zero/30000 in 500
…Set Output1
…COUNT=COUNT*ONE+ONE
Macro End
Macro Start 2 Repeat 0
End Program (0)
```

In the above program, Macro 3 waits for Input (5) then begins a Electronic Gearing at 2/3 ratio within 1000 master counts. The macro then waits for Inputs (6) and (7) to rise within the next 12000 master counts. If this condition is met, the macro continues. If both inputs do not rise within 12000 master counts the program jumps to Macro 3, sets output (2) adds one to the fault counter and goes back to the main program.

	NOTE
	*The Extended I/O when used in a program command are Inputs (17-32) and Outputs (9-16). Example:* ***…Wait for Rise Both 2/20 Max 1200*** *represents Standard Input 2 (IN=2) and Extended Input 20 (INX=8).*

5.3.36 …When Input Fall Clear TMC

Execution Time: 1 MLP (Position Loop Updates)

This macro step takes 1 MLP to initialize and clear TMC (Total Master Count) after the specified input falls. After the macro step is initialized, the macro continues and the condition runs in the background.

Example:

```
NEG=-1
Macro 1
…Gear at 10000/10000 in 1000
Macro 2
…When Input 6 Falls Clear TMC
…When Input 7 Falls Clear TSC
…Wait for Fall of Both Inputs 6/7 Max 12000
…OFFSET=TSC*-NEG+TMC
…Gear for OFFSET in 4000
Macro End
Macro Start 1 Repeat 1
Macro Start 2 Repeat 0
```

In the above program, Macro 1 begins electronic gearing. Macro 2 waits for Inputs (6) & (7) to fall. When Input 6 falls TMC is reset and after Input 7 falls TSC is reset. The difference between TMC and TSC is calculated and a correction is made (OFFSET) within 4000 master counts.

	NOTE
	*The Extended I/O when used in a program command are Inputs (17-32) and Outputs (9-16). Example:* ***…Input 28 Falls Clear TMC,*** *represents Extended Input 28 (INX=4096).*

5.3.37 …When Input Fall Clear TSC

Execution Time: 1 MLP (Position Loop Updates)

This macro step takes 1 MLP to initialize and clear TSC (Total Slave Count) after the specified input falls. After the macro step is initialized, the macro continues and the condition runs in the background.

Example:

```
NEG=-1
Macro 1
…Gear at 10000/10000 in 1000
Macro 2
…When Input 6 Falls Clear TMC
…When Input 7 Falls Clear TSC
…Wait for Fall of Both Inputs 6/7 Max 12000
…OFFSET=TSC*-NEG+TMC
…Gear for OFFSET in 4000
Macro End
Macro Start 1 Repeat 1
Macro Start 2 Repeat 0
```

In the above program, Macro 1 begins electronic gearing. Macro 2 waits for Inputs (6) & (7) to fall. When Input 6 falls TMC is reset and after Input 7 falls TSC is reset. The difference between TMC and TSC is calculated and a correction is made (OFFSET) within 4000 master counts.

	NOTE
	*The Extended I/O when used in a program command are Inputs (17-32) and Outputs (9-16). Example:* ***…Input 28 Falls Clear TSC,*** *represents Extended Input 28 (INX=4096).*

5.3.38 …When Input Rise Clear TMC

Execution Time: 1 MLP (Position Loop Updates)

This macro step takes 1 MLP to initialize and clear TMC (Total Master Count) after the specified input rises. After the macro step is initialized, the macro continues and the condition runs in the background.

Example:

```
NEG=-1
Macro 1
Gear at 10000/10000 in 1000
Macro 2
…When Input 6 Rises Clear TMC
…When Input 7 Rises Clear TSC
…Wait for Rises of Both Inputs 6/7 Max 12000
…OFFSET=TSC*-NEG+TMC
…Gear for OFFSET in 4000
Macro End
Macro Start 1 Repeat 1
Macro Start 2 Repeat 0
```

In the above program, Macro 1 begins electronic gearing. Macro 2 waits for Inputs (6) & (7) to rise. When Input 6 rises TMC is reset and after Input 7 rises TSC is reset. The difference between TMC and TSC is calculated and a correction is made (OFFSET) within 4000 master counts.

	NOTE
	*The Extended I/O when used in a program command are Inputs (17-32) and Outputs (9-16). Example:* ***…Input 28 Rises Clear TMC,*** *represents Extended Input 28 (INX=4096).*

5.3.39 …When Input Rise Clear TSC

Execution Time: 1 MLP (Position Loop Updates)

This macro step takes 1 MLP to initialize and clear TSC (Total Slave Count) after the specified input rises. After the macro step is initialized, the macro continues and the condition runs in the background.

Example:

```
NEG=-1
Macro 1
…Gear at 10000/10000 in 1000
Macro 2
…When Input 6 Rises Clear TMC
…When Input 7 Rises Clear TSC
…Wait for Rises of Both Inputs 6/7 Max 12000
…OFFSET=TSC*-NEG+TMC
…Gear for OFFSET in 4000
Macro End
Macro Start 1 Repeat 1
Macro Start 2 Repeat 0
```

In the above program, Macro 1 begins electronic gearing. Macro 2 waits for Inputs (6) & (7) to rise. When Input 6 rises, TMC is reset and after Input 7 rises TSC is reset. The difference between TMC and TSC is calculated and a correction is made (OFFSET) within 4000 master counts.

	NOTE
	*The Extended I/O when used in a program command are Inputs (17-32) and Outputs (9-16). Example:* ***…Input 28 Rises Clear TSC,*** *represents Extended Input 28 (INX=4096).*

5.3.40 …Zero Position

Execution Time: 1 MLP (Position Loop Updates)

This macro step is used to zero the target and feedback positions. The following error is maintained. Zeroing the Position is useful in conjunction with the macro “Relative Move” command which, does not zero the feedback position after each move and has a maximum displacement limit.

5.3.41 Clear Macro Controlled Outputs

This command turns off all outputs controlled by a macro. Outputs controlled by the "PLS" (Programmable Limit Switch) or the main program are not effected.

5.3.42 Macro

This command begins the definition of a macro, which is defined by the <*macro number*> and followed by up to 16 macro steps. The macro definition ends with the "Macro End" command or the next "Macro" command. **(Also See Example 8)**

Example1:

```
Macro 1
…
…
Macro End
Macro 2
…
…
Macro End
Macro 0
…
…
Macro End
```

Example 2:

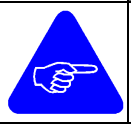
```
Macro 2
…
…
Macro 1
…
…
Macro 4
…
…
Macro End
…
…
```

Example 3:

```
Macro 5
…
…
Macro End
Program Commands
…
…
…
Macro 5
…
…
Macro End  Macro End
```

5.3.43 Macro Abort

This command is used to stop the execution of a specific macro. If any relative or absolute move is in progress, the motion will stop as quickly as possible without regard to programmed deceleration.

	NOTE
	*The "Call Macro" stack is **not** cleared with the program execution returning to the line below the "Call Macro" or "Macro Start" command.*

5.3.44 Macro End

This command is used in conjunction with the "***Macro***" command to end the macro definition. The "Macro End" command can follow each macro definition or be used to end several macros definitions.

5.3.45 Macro Start

This command starts the execution of a specific macro. The <macro number> can be specified as an expression, such as In&15, or simply a constant. The <repeat count> specifies how many times the macro should repeat execution.

Example: (Also See Example 8)

Macro 28
…Set Single Output 2 *Sets Output 2*
…Relative Macro Move to -10.2 revs in 800
…Clear Single Output 2
…Wait 500 milliseconds
Macro End
….
….
Macro Start 2 Repeat 12
Wait Macro Complete

In the above example, Macro 2 is defined then called to start. Output 2 is set, the motor will make a relative move of –10.2 revs, wait ½ sec then reset Output 2. This process will repeat 12 times.

	NOTE
	*A <repeat> value of zero signifies that the macro will repeat indefinitely until aborted.*

5.3.46 Wait Macro Complete

This command pauses execution of the main program until the current macro execution is complete. It generally follows a "***Macro Start***" command and used to prevent starting a move in the main program while the macro is executing a move.

5.4 Utility Commands

5.4.1 Clear Links

Breaks all links created by the "Link" and "Link Modulo" commands and ***must*** always be issued before creating a Link. **(Also See Example 7)**

5.4.2 Comment

Allows the programmer to enter comments throughout the program. The comment lines can be used to illustrate how the program works and are not executable.

	NOTE
	*Special or extended characters are not allowed: !, @, #, $, %, ^, &, *, (,), _, -, <, >,? ., ,, ', ", :,;, {, +, =, [, },], etc.*

5.4.3 Disable Event

This command allows the user to disable one or more events from within a program. This command when used in conjunction with the ***Enable Event*** command is useful when events may only be applicable to specific portion of a program.

	NOTE
	*Turning on/off events when required will speed up the execution of a program. Since events run in the background, the more events running the slower the processor becomes. (Look at the **Task/Second** in the monitor window).*

	NOTE
	*As a safety precaution, all events are re-enabled after the drive is reset or power cycled.*

5.4.4 Enable Event

This command allows the user to enable one or more events from within a program. This command when used in conjunction with the ***Disable Event*** command is useful when events may only be applicable to specific portion of a program.

	NOTE
	*Turning on/off events when required will speed up the execution of a program. Since events run in the background, the more events running the slower the processor becomes. (Look at the **Task/Second** in the monitor window).*

5.4.5 End Program

The ***End Program*** command is required as the last line of each program. It stops the motion program and provides for a non-zero end of a program to signal a fault condition. The user must enter an error code (default is zero).

Example:

```
        Index of 12 rev
        If (IN&2) Then Go To END
        ……..
END:
        End Program (99)
```

In the above example the motor will move 12 rev in the CCW direction. If Input is active after the move is complete, the program will terminate and have a Run Rime Fault of (–99).

	NOTE
	*Multiple **End Programs** may be used with specific Error Code Numbers to allow tracking of the conditions that resulted in program termination. Error Code "Zero" will not cause a "Run Time Fault" to appear, All NON-ZERO values will result in a negative "Run Time Fault" and drive disable.* .

5.4.6 Idle Main Program

This command is used to turn off secondary drive functions such as Splines, Oscilloscope and to idle the main program freeing up valuable processor time. It is used when the main program has nothing more to do since drive operation is controlled via macro's and inputs.

	NOTE
	*When running Macros, if the main program is not idled and an "**End Program**" command is executed, the Macro will abort.*

5.4.7 If…Then Clear Variable

This command sets the variable equal to 0 if a given condition is true. If the condition is false then the variable does not change. The user must enter both the expression for the condition and the name of the variable to be cleared.

Example:

```
If IN&4 Then Capture=0
```

In the above example if Input 3 is on when the ***If…Then Clear Variable*** command is executed the variable "Capture" is cleared.

5.4.8 If…Then End Program

This command ends the program if given condition is true. If the condition is false then the program continues on to the next line. The user must enter the expression or condition that will end the program with an error code of zero.

Example:

```
TOP:
    Index of 12 rev
    If TPOS>70 Then End
    Go To TOP
```

In the above example if Target Position is greater than 70 system-revs when the ***If…Then End Program*** command is executed the motor will stop and the program will end without generating a fault.

5.4.9 Link

This command creates a linked <*target variable*> which is recomputed once per position loop in the same order they are defined. (Intelligent Drive every 2 milliseconds). All specified parameters must be variables that have been previously defined. Once this link is created, the <*target variable*> will always equal the defined relationship, regardless of the state of the program. The linked <*target variable*> must follow the following format:

Link <*target variable*>=<*variable A*>*<*variable B*>+<*variable C*>

Example: (Also See Example 7)

```
V1=-1
V2=110
Clear Links
Link POS1=PLSX*V1+V2
Link POS2=POS1*V1+V2
```

The above example POS1 and then POS2 is calculated once every MLP.

	NOTE
	*There is a limit of (2) "Link" and/or "Link Modulo" variables for the Intelligent Drive.*

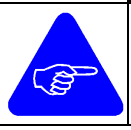
5.4.10 Link Modulo

This command creates a linked <*target variable*> which is equal to the <source variable> modulo <devisor variable>. Once this link is created, the <*target variable*> will always equal the defined relationship, regardless of program execution. Linked variables are recomputed once per MLP.

Example:

Link Modulo XYZ=FPOS/ONE

(XYZ fluctuates between 0 and 1)

	NOTE
	*There is a limit of (2) "Link" and/or "Link Modulo" variables for the Intelligent Drive.*

5.4.11 Quick Stop on Input Mask

This command causes a quick stop fault if a given input or sets of inputs are inactive (off) when the command is executed. The motor will hard decelerate to a stop causing a fault condition in the drive that stops program execution. The Input Mask is read as binary pattern of the input. An input mask of (9) would represent 00001001 or Inputs 1 & 4. (Binary is read from right to left)

Example:

Quick Stop on Input Mask 12

In the example above if Input 3 or Input 4 (00001100) are off the motor will Quick Stop and the drive will fault.

5.4.12 Set variable = expression

Sets a "System" or "User" variable equal to a given expression at the time the command is executed. The expression may be any valid mathematical expression using any of the arithmetic functions supported by the drive. The expression can be used to change an existing system variable or used to create a new user variable.

Example: (Also See Example 2, 7 & 8)

```
COUNT=ZERO
……..etc.
COUNT=COUNT+ONE
```

The above example changes the user variable to zero then adds a count of one each time through. This will insure the value is changed by exactly one count.

V8=V8-(TPOS*0.8)

This expression makes V8 equal V8 minus o.8 times the target position.

5.4.13 Slew Variable

This command changes a variable to a final value over a specified time period. The execution of the program is halted until the variable reaches it's final value. Only one active variable slew is allowed at one time. The user must enter the variable to slew, final value of variable and time in seconds.

Example:

Slew KP: 1.219, 30 [final value, time (sec)]

This expression changes the value of KP (Velocity Proportional Gain) from some starting value to 1.219 Nm/RPM over 30 sec. This is an important function for Web Tensioning applications, which use PID loops. The KP can be adjusted over time to allow for the long acceleration required. This will prevent jerking of the web, which would occur if the KP were instantly changed.

5.4.14 Software Disable

Sets the SWE (Software Enable) register to false. If the ***Enable Source*** (EM) is set for software enable, the amplifier will become disabled. To verify the Enable Source, click on "Parameters" then "Modes".

Example:

```
If (IN&2) then Go To Disable
…… etc.
DISABLE:
Software Disable [SWE=0]
```

In the above example, if Input 2 is on when the command is executed then the program will jump to the label DISABLE and software disable the drive.

5.4.15 Software Enable

Sets the SWE (Software Enable) register to true. If the ***Enable Source*** (EM) is set for software enable, the amplifier will become enabled. To verify the Enable Source, click on "Parameters" then "Modes".

Example:

```
TOP:
If (IN&6) then Go To Enable
Go To TOP
ENABLE:
Software Enable [SWE=1]
```

In the above example, if Input 6 is on the program will jump to label ENABLE and software enable the drive.

5.4.16 Trigger Capture

This command is used in conjunction with the scope by triggering the acquisition of data which will be uploaded for display on the on-screen oscilloscope. The user can plot a number of variables vs. time, including feedback position, velocity, position error, etc. After opening the oscilloscope make sure you set the begin plot on trigger capture.

Example: (Also See Examples 3, 4 & 8)

```
Trigger Capture
Index of 12 system-rev
```

At the point the ***Trigger Capture*** command is executed the oscilloscope begins sampling the data to display on the screen.

	NOTE
	*Remove the **Trigger Capture** command from the program after debugging in order to speed up the program execution.*

5.4.17 Wait (Dwell)

Range: 1 to 432,537 msec.

This command causes the program execution to pause for the specified interval of time. The user must enter the dwell time in milliseconds (1/1000 sec).

Example:

```
Index of 11.3 rev
Wait for in Position
Wait 2200 msec
        Index of –5.24 rev
```

The program will Index 11.3 rev, pause for 2.2 sec then index –5.24 rev.

5.4.18 Zero the Feedback Position

This command zeroes the internal Feedback Position counter (FPOS) while maintaining the following error. For example, if the Target Position is 11.0 rev and the Feedback Position is at 10.9 rev, after execution of this command the target position will be set to 0.1 rev and the feedback position to 0.0 rev.

Example: (Also See Examples 3 4 & 8)

```
HOME:
    Move at 3 inches/sec Until Input 7
    End Move at Zero
    Wait for in Position
    Move to Feedback Null (-)
    Wait for in Position
    Set Feedback Position to 0.0
    Go To TOP
```

The motor will rotate at 3 inches/second until Input 7 becomes active. The motor will stop, then move to the feedback null before setting the feedback position to zero.

5.4.19 Zero the Following Error

This command zeroes the following error as well as the target position and feedback position registers.

Example: (Also See Examples 1, 2, 5, 6, 7 & 8)

```
    Move At 0.1 inches/sec While Current < 40%
    End Move At: Move By 0 inches
    Zero the Following Error
```

This program causes the motor to relax by making the TPOS value equal to the FPOS value. Assume that you have moved into a HARDSTOP and want the motor to relax (else the amplifier will command 40% of drive peak current to try and get to position)

5.5 Gearing Commands

5.5.1 Begin Master Position Tracking

This command switches the drive to Master/Slave electronic gearing while in "Position" mode. The motion of the axis will follow a master encoder signal from an auxiliary encoder.

Example: (Also See Example 8)

```
Begin Master Position Tracking
Connect Master Encoder
Gear At 50000/10000
```

In the above example, the slave drive will rotate 5 system-rev for each system-rev the master axis rotates.

The following conditions must be met:

(1) Amplifier configuration must be in Position mode.
(2) Issuing any subsequent motion command will terminate the Master Position Tracking
(3) Queued Motion cannot be active.
(4) "Gear At" commands must be issued while Master Position Tracking is active to allow changes in the Electronic Gearing Ratio of the amplifier
(5) The "End Master Position Tracking" command can be used to return to position mode.

5.5.2 Capture Failure Parameters

This command defines when and how to handle repetitive high-speed failures. (A failure constitutes when the limit is exceeded) The <*failure count*> specifies how many failures must occur in a row before a capture fault occurs. When the fault is triggered, the macro specified in <*macro to execute*> is "Chained" to and run.

	NOTE
	*Failures are ignored unless the "Capture Failure Parameters" command us issued after execution of a "Begin Master Position " command.*

Example:

Begin Master Position Tracking
Connect Master Encoder
Capture Failure Parameters 10 failure, Macro 7
Macro Start 7 Repeat 0
Wait Macro Complete

The example above will execute Macro 7 if the "**Wait Capture**" command in the gearing macro "times out" 10 times in a row. If the command times out 9 times then receives a trigger, the failure counter is reset to zero.

5.5.3 Connect Master Encoder

This command connects the master position encoder input to the drive effectively reversing the effect of the "Disconnect" command. Identical to the "…Connect" macro step.

Example: In the example below, an arm is pulling material fed between two rollers, which prevents the material from stretching. The secondary encoder on the arm has a 2/1 gear ratio to the rollers. At the end travel for the arm, Input 6 becomes active which ends master position tracking. This prevents material from being back fed through the rollers as the arm retracts.

```
        Begin Master Encoder Tracking
        Connect
        Gear At 20000/10000
    HOLD
        If Input 6 is Off, Go TO HOLD
        Gear at ZERO/10000 in 100
        Disconnect
```

The above macro will connect the master position encoder so that the drive will perform electronic gearing with a 2:1 ratio. When Input 6 becomes active, the drive will disconnect the master position encoder, which essentially ignores any pulse transitions in the secondary encoder.

	NOTE
	*By default, the master position encoder input is **Connected** when Master Position Tracking is initiated.*

5.5.4 Disable PLS

This command disables all defined PLS (Programmable Limit Switch) outputs. (Outputs go off)

Example:

```
        Disable PLS
        PLS 3=FPOS/25/30
        Enable PLS
```

In the above example the PLS outputs are disabled in order to be defined. Output 3 is defined as a PLS output which will be set whenever the feedback position (FPOS) falls between 25 and 30 system-revs.

	NOTE
	*The PLS Outputs must be disabled before defining.*

5.5.5 Disconnect Master Encoder

This command is used to disconnect the master position encoder input by ignoring any pulse. Identical to the "…Disconnect" macro step.

Example: In the example below, an arm is pulling material fed between two rollers, which prevents the material from stretching. The secondary encoder on the arm has a 2/1 gear ratio to the rollers. At the end travel for the arm, Input 6 becomes active which ends master position tracking. This prevents material from being back fed through the rollers as the arm retracts

```
Begin Master Encoder Tracking
Connect
Gear At 20000/10000
HOLD
If Input 6 is Off, Go TO HOLD
Gear at ZERO/10000 in 100
Disconnect
```

The above macro will connect the master position encoder so that the drive will perform electronic gearing with a 2:1 ratio. When Input 6 becomes active, the drive will disconnect the master position encoder, which essentially ignores any pulse transitions in the secondary encoder.

5.5.6 Enable PLS

This command enables all defined PLS (Programmable Limit Switch) outputs. By default, (after power-up and resets) all PLS outputs are disabled (off).

Example:

```
Disable PLS
PLS 3=FPOS/25/30
Enable PLS
```

In the above example the PLS outputs are disabled in order to be defined. Output 3 is defined as a PLS output which will be set whenever the feedback position (FPOS) falls between 25 and 30 system-revs.

5.5.7 End Master Position Tracking

This command ends the Position Tracking mode. It allows the user to switch in and out of Master/Slave mode within a program.

Example: In the example below, the system is in Master/Slave mode until Input 6 becomes active at which time it disables the "Master Position Tracking and makes a Indexed move. **(Also See Example 8)**

```
Gear At 20000/10000
Begin Master Position Tracking
HOLD
    If Input 6 is Off, Go To HOLD
    End Master Position Tracking
    Indexed of 12.8 system-revs at 20 system-revs/sec
    Wait for in Position
```

By using the "***Begin Master Position Tracking***" and the "***End Master Position Tracking***" commands, the program is able to change from "Master/Slave" mode to "Position" mode and back again.

5.5.8 Gear At

This command is used to change the "Master Tracking" gearing. The numerator or denominator may be specified as a constant or a variable. The relationship for the ratio is

Slave = (Numerator x Master) /Denominator

Example:

```
Gear At 30000/100000
```

Sets the gear ratio between the master slave to 30000/100000 (3/10). The slave will move 3 counts for every 10 counts of the master encoder.

	NOTE
	*It is recommend that the Electronic Gear Ratio numerator and denominator are as large as possible to provide a smooth transition between gear ratios. (Numerator and Denominator < 1,000,000)* *Gear At 2/3 in 1000 will transition in one master count. Gear At 20000/30000 in 1000 will transition within 1000 master counts.*

5.5.9 PLS

This command creates a "Programmable Limit Switch" for a given output, which is "active" if the variable is within the upper and lower limits specified. PLS outputs are updated once per MLP. (2 milliseconds for Intelligent Drive)

Maximum PLS Outputs for Intelligent Drive (2)

Example:

```
Disable PLS
PLS 3=PLSX/1000/1600
Enable PLS
```

In the above example the PLS outputs are disabled in order to be defined. Output 3 is defined as a PLS output which will be set whenever the system variable "PLSX" (Slave Modulo "SMOD") falls between 1000 and 1600 counts.

	NOTE
	*Due to software conflicts, the same output cannot be effectively controlled by both a PLS and the "main" program or macro. Therefore use PLS outputs which are not used elsewhere in the program.* *All PLS outputs must be disabled before defining a new output.*

5.5.10 Zero SMOD

This command clears the "Slave Modulo" (SMOD). Used to set a known modulo position in a continuous master/slave system.

	NOTE
	*System variable SMOD is cleared synchronously in Master Tracking mode.*

5.6 Motion Commands

5.6.1 Absolute Move, Shortest Time

This command moves the motor to an absolute position without speed limits. The system generates a move profile using the last acceleration and deceleration rates. The motor velocity will be dependent on the maximum system velocity or maximum current.

Example: Set Acceleration Rate to 400 system-rev/sec^2
Absolute Move To 20.23 system-rev

In this example, if the motor position is greater than 20.23 rev the motor will rotate CW. If the motor position is less than 20.23 rev the motor will rotate CCW. The motor velocity is dependent on move distance, peak current and maximum motor velocity.

	NOTE
	*If an acceleration rate has not been previously set, the motor will accelerate ½ the move distance and decelerate the other ½ giving a triangular move profile. For long moves, this may result in an **OSPD** over speed fault or will cause a step motor to stall.*

5.6.2 Absolute Move, Speed-limited

This command moves the motor to an absolute position with the speed limited to the specified maximum. The system generates a trapezoidal move profile using the last acceleration and deceleration rates along with the specified maximum velocity.

Example: **(Also See Examples 6 & 7)**

Set Acceleration Rate to 65 system-rev/sec^2
Absolute Move To 43.0 system-rev At 30 system-rev/sec
Wait for In Position

In this example, if the motor position is greater than 43.0 rev the motor will rotate CW. If the motor position is less than 43.0 rev, the motor will rotate CCW. The motor velocity will be 30 system-rev/sec as long as the move distance is great enough to accelerate to maximum velocity.

	NOTE
	*If an acceleration rate has not been previously set, the motor will accelerate ½ the move distance and decelerate the other ½ giving a triangular move profile.*

5.6.3 Absolute Move, Time-limited

This command moves the motor to an absolute position without any speed limits. The system generates a triangular move profile based on the acceleration and deceleration rates required to meet the move time. (The previous acceleration rate remains unchanged.) The program will not execute the next command until the target position reaches the end. To insure the Feedback Position reaches the desired end position use the ***Wait for In Position*** command.

Example: (Also See Examples 7 & 8)

```
Absolute Move To 12.0 system-rev In 4.2 sec
Wait for In Position
```

In this example, if the current motor position is greater than 12.0 rev the motor will rotate CW. If the current motor position is less than 12.0 rev the motor will rotate CCW. The motor velocity and acceleration rates are calculated based on move distance and move time.

5.6.4 Change Move at Velocity

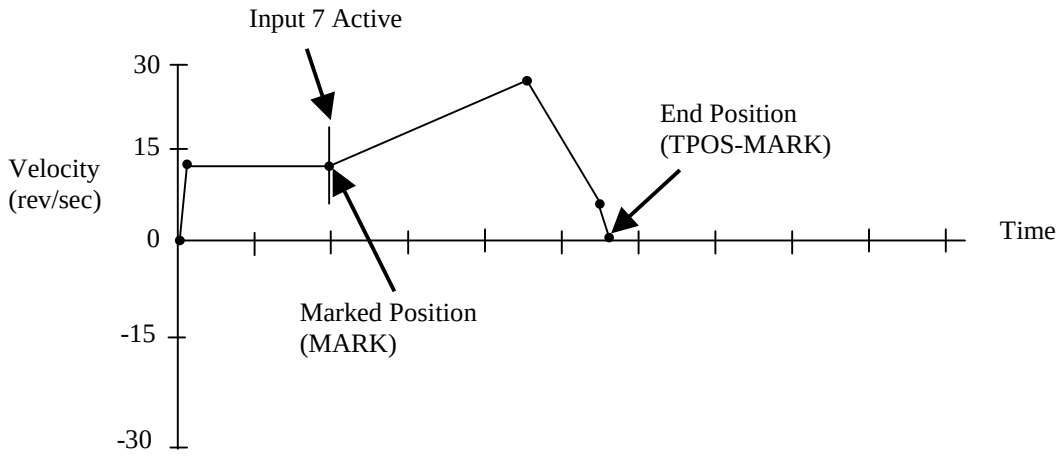
This command is only valid with a "**Move At …**" command and is used to change a velocity over a specified period of time. Program execution is paused until the command is completed and the new velocity is met. Several ***Change Move at Velocity*** commands can be used during a single move.

Example:

```
Set Acceleration Rate to 150 system-rev/sec^2
Move at 12.5 rev/sec Until Input 7
Change Move At Velocity 25 system-rev/sec in 5 sec
Change Move At Velocity 5 system-rev/sec in 2 sec
Change Move At Velocity 0.01 system-rev/sec in 0.3 sec
V1=TPOS-MARK
End Move At; Move By V1 rev
```

In this example the motor will rotate at 12.5 rev/sec CCW until input 7 becomes active, then accelerate to 25 rev/sec over 5 sec, decelerate to 5 rev/sec over 2 sec, decelerate to 0.01 rev/sec in 0.3 sec then end move at current position. (See Move Profile below)

Move Profile:



5.6.5 Don't Queue Motion Command

This command reverses the ***Queue Motion Command*** by changing the program back to a pause mode during motion commands. The program can be switched in and out of the Queue Motion mode.

5.6.6 End Move At

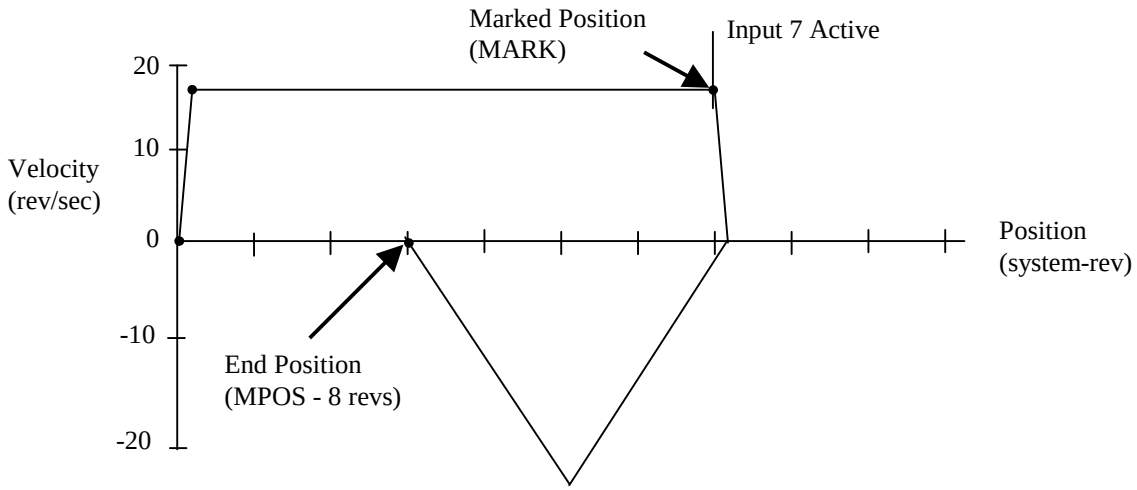
This command is required after a "**Move At …**" command is executed in order to end the move command. The move is ended at a specified relative position to the marked position (MARK).

Example: (Also See Examples 1, 2 & 8)

```
Set Acceleration Rate to 200 system-rev/sec2
Move at 18.145 rev/sec Until Input 7
If (IN=6) then END6
 End Move At; Move By –8.0 rev
..…ect.
END6:
End Move At; Move By –10.0 rev
```

In this example the motor will rotate at 18.333 rev/sec CCW until input 7 becomes active. Once the condition is met the position will be marked, the motor will reverse direction and rotate to an offset of -8.0 revs CW of the marked position. If Input 6 is on the program will jump to label END6 and will reverse direction and rotate to an offset of -10.0 revs CW of the marked position.

Move Profile:



	NOTE
	If the End Move At command is not issued, the motor will continue to move and a software fault will be generated with the next time a move command.

5.6.7 Hard Stop Queue Motion

This command when executed will decelerate the motor immediately using the fastest deceleration time possible. The command can be used with a Index, Absolute Move or Spline.

Example:

```
TOP:
    Queue Motion Command
    Index of 43.0 system-rev At 15 system-rev/sec
MOVING:
    If ABS(IRA)>1.2 Then Go To STOP
    If Queue Motion Complete Go To DONE
    Go To MOVING
DONE:
    Don't Queue Motion Command
    Wait for In Position
    Go To TOP
STOP:
    Queued Motion: Hard Stop
```

In the above example, the motor will make an indexed move of 43 system-rev at 30 system-rev/sec. During the move if the motor current exceeds +/-1.2 amps the program will jump to the label "STOP" and immediately decelerate at to 0 rev/sec at quickest time possible.

	NOTE
	*If the **Queue Motion** command is used with a **Move At…** command a software fault will be generated and the drive will shutdown.*

5.6.8 Index, Shortest Time

This command rotates the motor a fixed distance without speed limits. The system generates a move profile using the last acceleration and deceleration rates. The motor velocity will be dependent on the maximum system velocity and/or the maximum current.

Example:

Set Acceleration Rate to 400 system-rev/sec^2
Index of -32.5 system-rev At 10 system-rev/sec
Wait for In Position

In this example, the motor will rotate of 32.5 system-rev in the CW direction with the velocity being dependent on the acceleration rate, peak current and/or maximum motor velocity.

	NOTE
	*At the start of an indexed move the feedback position is reset to zero. If indexed moves are used in conjunction with absolute move the zero reference will change.*

5.6.9 Indexed, Speed-limited

This command moves the motor a fixed distance with the speed limited to the specified maximum. The system generates a move profile using the last acceleration and deceleration rates along with the specified maximum velocity.

Example:

Set Acceleration Rate to 190 system-rev/sec^2
Index of 34.5 rev At 25 rev/sec
Wait for In Position

In this example, the motor will rotate 34.5 rev in the CCW direction at a maximum velocity of 25 rev/sec.

	NOTE
	*At the beginning of an indexed move the feedback position is reset to zero. If an indexed move is used in conjunction with an absolute move the zero reference will change.*

5.6.10 Move At… Until Input Active

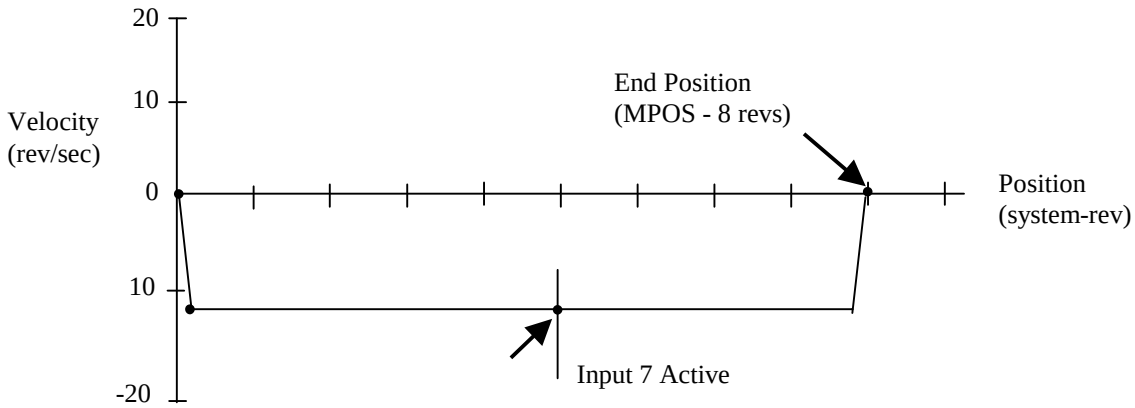
The motor will rotate at a given velocity until a specified input becomes active. Once the input becomes active, the position is captured by the variable MARK and program execution continues. The direction of motor rotation can be changed by using a + or - sign. The user must enter a velocity and specify an input. .

Example: (Also See Examples 1, 2 & 8)

Set Acceleration Rate to 400 system-rev/sec^2
Move at -12.333 rev/sec Until Input 7
End Move At; Move By –8.0 rev

In this example the motor will rotate at -12.333 rev/sec CW until input 7 becomes active. Once the condition is met the, position will be marked and the motor will rotate to an offset of 8.0 revs CW of the marked position.

Move Profile:



	NOTE
	*The **Move at** command needs to be followed by an **End Move at** command in order to terminate the move. After the Move At condition is met the motor will continue rotating at the same velocity until the **End Move At** command is issued.*

5.6.11 Move At… While Current <

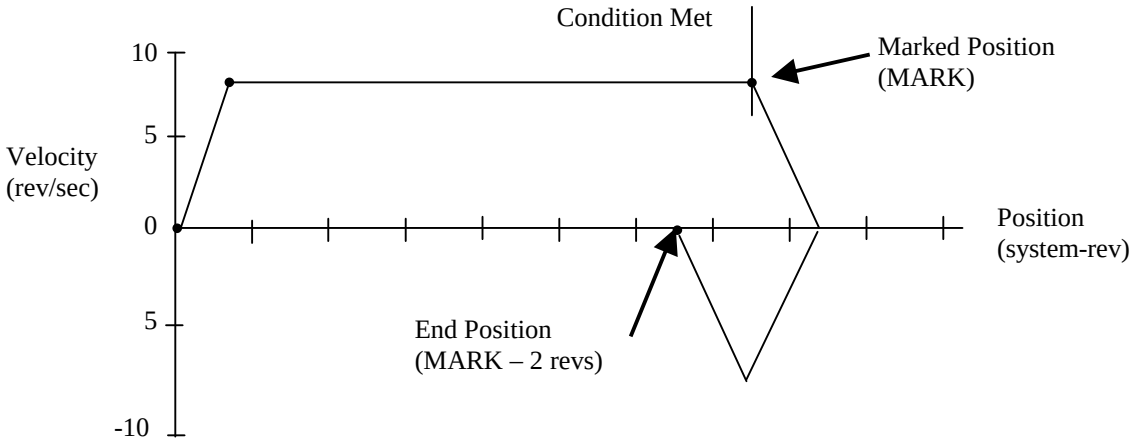
The motor will rotate at the specified velocity until the % of peak drive current reaches the specified value. The program flow is paused until the condition is met. Once the specified percent of peak current is reached the position is captured by the variable MARK and program execution continues. The direction of motor rotation can be changed by using a + or - sign. The user must enter a velocity and the percentage of commanded peak drive current.

Example:

Set Acceleration Rate to 20 system-rev/sec^2
Move at 8.25 rev/sec While Current < 12%
End Move At; Move By –2.0 rev

Assuming the drive is 6 Amp RMS (12 Amp Peak), the motor will rotate CCW at 8.25 rev/sec until the current reaches 1.44 Amps. Once the condition is met the position will be marked and the motor will rotate to a relative position of 2.0 revs CW of the marked position.

Move Profile:



	NOTE
	*The **Move at** command needs to be followed by an **End Move at** command in order to terminate the move. After the Move At condition is met the motor will continue rotating at the same velocity until the **End Move At** command is issued.*

5.6.12 Move At… While Input Inactive

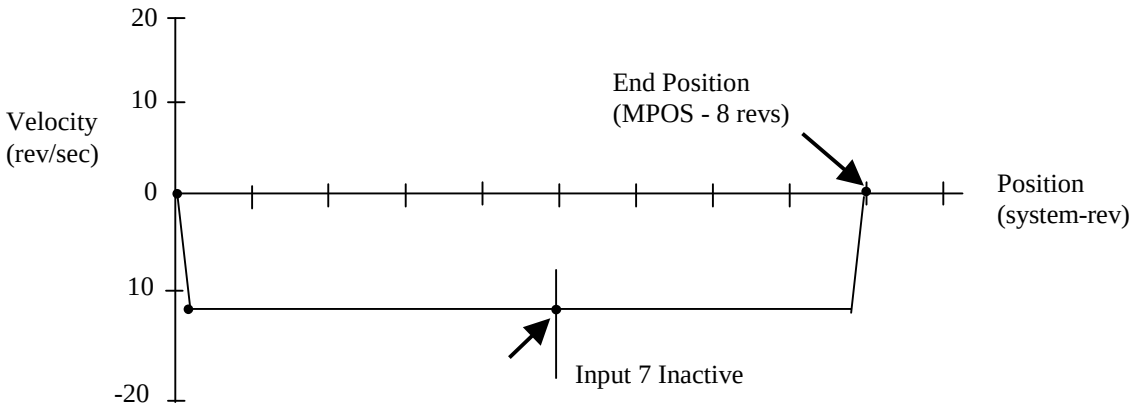
The motor will rotate at the given velocity until the specified input becomes inactive. The program flow is paused until the condition is met. Once the input becomes inactive, the position is captured by the variable "MARK" and program execution continues. The direction of motor rotation can be changed by using a + or - sign. The user must enter a velocity and specify the input.

Example: (Also See Examples 1)

Set Acceleration Rate to 600 system-rev/sec^2
Move at -12.333 rev/sec While Input 7
End Move At; Move By -8.0 rev

In this example, the motor will rotate at 12.333 rev/sec CW until input 7 becomes inactive. Once the condition is met the, position will be marked and the motor will rotate to an offset of 8.0 revs CW of the marked position.

Move Profile:



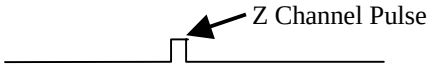
	NOTE
	*The **Move at** command needs to be followed by an **End Move at** command in order to terminate the move. After the Move At condition is met the motor will continue rotating at the same velocity until the **End Move At** command is issued.*

5.6.13 Move to Feedback Null (-)

This command rotates the motor in the negative (CW) direction until the feedback null is reached. The feedback null is as follows:

(Also See Examples 1, 2 & 7)

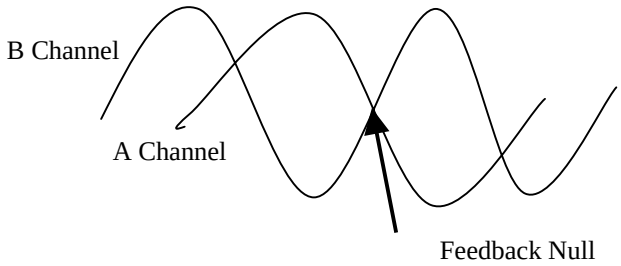
Encoder Feedback - The motor will rotate CW until Z channel pulse is reached. (This will occur once per motor rev.)



2 Pole Resolver - The motor will rotate CW until the B channel rises and crosses the A channel which is falling. (This will occur once per motor rev.)

4 Pole Resolver - The motor will rotate CW until the B channel rises and crosses the A channel which is falling. (This will occur twice per motor rev.)

8 Pole Resolver - The motor will rotate CW until the B channel rises and crosses the A channel which is falling. (This will occur four times per motor rev.)



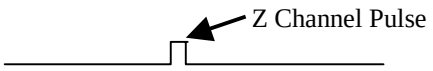
	NOTE
	In an encoder system, the motor must rotate one complete motor revolution before homing to the Z-channel pulse

5.6.14 Move to Feedback Null (+)

This command rotates the motor in the Positive (CCW) direction until the feedback null is reached. The feedback null is as follows:

(Also See Examples 1, 2 & 7)

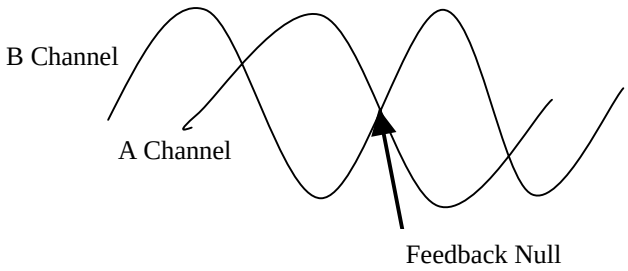
Encoder Feedback - The motor will rotate CCW until Z channel pulse is reached. (This will occur once per motor rev.)



2 Pole Resolver - The motor will rotate CCW until the B channel rises and crosses the A channel which is falling. (This will occur once per motor rev.)

4 Pole Resolver - The motor will rotate CCW until the B channel rises and crosses the A channel which is falling. (This will occur twice per motor rev.)

8 Pole Resolver - The motor will rotate CCW until the B channel rises and crosses the A channel which is falling. (This will occur four times per motor rev.)



	NOTE
	In an encoder system, the motor must rotate one complete motor revolution before homing to the Z-channel pulse

5.6.15 Queue Motion Command

Generally, a motion command (Such as Index, Absolute Move, Spline Move) causes program execution to pause until the target position is reached. This command allows the program flow to continue immediately after the motion command begins. A fault will occur if a "***Move at***" command is used or another motion command is attempted before the first motion command is completed. Use the "***Don't Queue Motion Command***" to turn off queued motion.

Example:

```
TOP:
    Queue Motion
    Index of 43.0 system-rev At 15 system-rev/sec
MOVING:
    If ABS(IRA)>1.2 Then Go To STOP
    If Queued Motion Complete Go To DONE
    Go To MOVING
DONE:
    Don't Queue Motion Command
    Go To TOP
STOP:
    Queued Motion: Soft Stop
```

In the above example the motor makes an index move of 43 system-rev at 15 system-rev/sec. During the move if the motor current exceeds 1.2 amps the program will jump to the label "STOP" and decelerate.

5.6.16 Relative Move, Shortest Time

This command makes a fixed move distance without speed limits. The system generates a move profile using the last acceleration and deceleration rates. The motor velocity will be dependent on the maximum system velocity or maximum current. The direction of the move is defined by a (+/-) sign. Unlike the "**Indexed Move**" the feedback position is not reset to zero.

Example:

Set Acceleration Rate to 400 system-rev/sec^2
Relative Move Of 24.63 rev
Wait for In Position

In the previous example, the motor will rotate 24.63 revs in the CCW direction.

	NOTE
	*If an acceleration rate has not been previously set, the motor will accelerate ½ the move distance and decelerate the other ½ giving a triangular move profile.*

5.6.17 Relative Move, Speed Limited

This command makes a fixed move distance with the speed limited to the specified maximum. The system generates a trapezoidal move profile using the last acceleration and deceleration rates along with the specified maximum velocity. Unlike the "**Indexed Move**" the feedback position is not reset to zero.

Example:

Set Acceleration Rate to 65 system-rev/sec^2
Relative Move Of -43.0 system-rev At 30 system-rev/sec
Wait for In Position

In this example, if the motor position is greater than 43.0 rev the motor will rotate CW. If the motor position is less than 43.0 rev, the motor will rotate CCW. The motor velocity will be 30 system-rev/sec as long as the move distance is great enough to accelerate to maximum velocity.

5.6.18 Relative Move, Time Limited

This command makes a fixed move distance without any speed limits. The system generates a triangular move profile based on the acceleration and deceleration rates required to meet the move time. (The previous acceleration rate remains unchanged.) The program will not execute the next command until the target position reaches the end. To insure the Feedback Position reaches the desired end position use the "***Wait for In Position***" command.

Example:

Set Acceleration Rate to 1000 system-rev/sec^2
Relative Move Of -12.0 system-rev In 4.2 sec
Wait for In Position

In this example, the motor will rotate CW 12 system-rev.

5.6.19 Set Acceleration Rate

This command sets the acceleration and deceleration rate for the position mode. If the acceleration rate is not defined before a move command, the default rate is 1000 units/sec/sec.

Example: **(Also See Examples 1, 2 & 6)**

Set Acceleration Rate to 65 system-rev/sec^2
Absolute Move To 43.0 system-rev At 30 system-rev/sec
Wait for In Position

The above example sets the acceleration rate to 65 system-rev/sec^2 before making the absolute move.

5.6.20 Set Move at Limit

Sets the maximum displacement allowed during a "Move At" command. If the condition is not met within the maximum move limit the drive will "fake" the condition and cause the move to complete. This command can be issued if a "Move At" command is used to locate a registration mark. If the machine jams and the registration mark does not appear within a fixed move distance the motor will stop.

Example:

Set Move At Limit Displacement to 45 system-rev
Move at 10.5 rev/sec Until Input 7
End Move At; Move By 0.2 rev

In the above example, the motor will rotate 10.5 rev/sec until Input 7 becomes active. If Input 7 does not become active within 45 system-revs the drive will fake the condition and stop the motor at 0.2 revs.

5.6.21 Soft Stop Queued Motion

This command when executed will immediately decelerate the motor using the last acceleration time. The command can be used with a Index, Absolute Move or Spline.

Example:

```
TOP:
    Queue Motion Command
    Index of 43.0 system-rev At 15 system-rev/sec
MOVING:
    If ABS(IRA)>1.2 Then Go To STOP
    If Queue Motion Incomplete Go To MOVING
    Go To MOVING
DONE:
    Don't Queue Motion Command
    Wait for In Position
    Go To TOP
STOP:
    Queued Motion: Soft Stop
```

In the above example the motor will make an index move of 43 system-rev at 15 system-rev/sec. During the move if the motor current exceeds 1.2 amps the program will jump to the label "STOP" and decelerate.

	NOTE
	*If the **Queue Motion** command is used with a **Move At…** command a software fault will be generated and the amplifier will shutdown.*

5.6.22 Wait for In Position

This command is used after a move command to stop the program flow until the feedback position and target position are with-in a specified Position Error Tolerance. The range is set by the program parameter PET (Position Error Tolerance). This command is recommend after each move command.

Example: (Also See Examples 1, 2, 6 & 8)

```
Absolute Move To 14.1 system-rev At 30 system-rev/sec
Wait for In Position
Index of 23.4 rev
Wait for In Position
```

In this example, the program will wait until the feedback position and target position are with-in the position error tolerance before executing the next move command. A single sample of Position Error (PERR) is compared to the Position Error Tolerance (PET) to determine if the motor is in position. The sample rate of PERR is based on program speed.

5.7 Go To Commands

5.7.1 Call

The ***Call*** command alters the flow of the program by jumping to the specified label and executing the next command. When the ***Return*** command is reached the program execution will return to the line below the Call command and continue.

Example:

```
        Call POSITION
        Set Output(s) 3
        ……..
POSITION:
        Absolute Move to 20.34 rev at 20 rev/sec
        Wait for In Position
        Return
```

The above program will jump to label POSITION execute the absolute move the return to the command under call and set output 3

5.7.2 Go To

The ***Go To*** command alters the flow of the program by jumping to a specified label and continuing to execute. Unlike the ***Call*** command the program does not return to the original line.

Example: (Also See Example 6, 7 & 8)

```
        Go To HOME
        ……..
HOME:
        Absolute Move to 0.0 rev at 20 rev/sec
        Wait for in Position
        Set Output(s) 3
```

The above program will jump to the label HOME and move to absolute position 0.0 revs before setting output 3. The program flow continues from this point.

5.7.3 If Queued Motion Complete

This command allows the program to branch to a specified label if the Queued Motion has been completed. If the motion has not been completed the program will continue executing the program.

5.7.4 If Queued Motion Incomplete

This command allows the program to branch to a specified label if the Queued Motion has not been completed. If the motion is complete the program will continue to execute.

5.7.5 If…Then…Call

This command is a conditional ***Call*** statement. If the given condition is true the program will jump to the call label. When the program hits the ***Return*** command it jumps back to the command below the ***If…Then…Call.*** If the condition is not true the program will continue to execute the next line of code.

Example:

```
TOP:
    Relative Move of 1.53 rev at 5 system-rev/sec
    If FPOS>64.5 Then Call HOME
    Go To TOP
HOME:
    Absolute Move to 0.0 rev at 20 system-rev/sec
    Return
```

The above program will make a number of 1.53 rev index moves until the feedback position is greater than 64.5 revs at which time it will jump to label HOME, position at 0.0 rev, return to command “Go To TOP” and continue the cycle.

5.7.6 If…Then…Go To

This command is a conditional *Go To* statement. If the given condition is true the program will jump to the given label and continue executing. If the condition is not true the program will continue to execute the next line of code. The user must enter both the condition and the Go To label.

Example: (Also See Example 6)

```
        Queue Motion
        Relative Move of 31.23 rev at 10 system-rev/sec
        Wait 600 msec
        If ABS(IRA)>2.3 Then Go To STOP
        ……..
STOP:
        Queued Motion: Hard Stop
        Set Output(s) 2
```

The above program will make an indexed move of 31.23 revs. The program will wait 600 msec, then if the motor current exceeds +/-2.3 amps it will jump to label STOP, hard stop the motor and set output 2

5.7.7 Label

This command creates the labels for "***Go To***" and "**Call**" commands. A label can consist of 1-8 alphanumerical characters.

5.7.8 Return

The return command is used in conjunction with a "***Call***" or "***If…Then…Call***" command to alter the program flow and return to the line below the Call statement.

5.8 I/O Commands

5.8.1 Clear an Output

Clears (Resets) one or more outputs. If more than 1 output is cleared, the output numbers must be separated by commas.

Example: (Also See Example 7)

Clear Output(s) 1, 3, 4

	NOTE
	*The Extended I/O when used in a program command are Inputs (17-32) and Outputs (9-16). Example:* ***Clear Output(s) 3,9,16*** *represents Standard Output 3 (OUT=4) and Extended Outputs 9 & 16 (OUTX=129).*

5.8.2 If Input is Off…Go To

This command is a conditional ***Go To*** statement based on the state of a given input. If the input is off, (inactive) the program will jump to the specified label. If the input is on, (active) the program will continue executing the next line. The user must specify the input number and label.

Example: (Also See Examples 7 & 8)

TOP:
If Input 2 is Off, Go To TOP
Index of 12.25 rev

If input 2 is inactive the program will jump to the label TOP otherwise it will execute the indexed move of 12.25 revs.

	NOTE
	*The Extended I/O when used in a program command are Inputs (17-32) and Outputs (9-16). Example:* ***If Input 20 Off, Go To Top*** *represents Extended Input 20 (INX=8).*

5.8.3 If Input is On…Go To

This command is a conditional ***Go To*** statement based on the state of a given input. If the input is on, (active) the program will jump to the specified label. If the input is off, (inactive) the program will continue executing the next line. The user must specify the input number and label.

Example: (Also See Example 6)

```
TOP
    If Input 2 is On, Go To POS2
    Go To Top
    POS2:
    Index of 6.71 rev
```

If input 2 is inactive the program will jump to the label POS2 otherwise it will execute the indexed move of 6.71 revs.

	NOTE
	*The Extended I/O when used in a program command are Inputs (17-32) and Outputs (9-16). Example:* ***If Input 20 On, Go To Top*** *represents Extended Input 20 (INX=8).*

5.8.4 Set Output

This command sets one or more outputs. If more than 1 output is set, the output numbers must be separated by commas.

Example: (Also See Example 7)

```
Set Output(s) 2, 3
```

	NOTE
	*The Extended I/O when used in a program command are Inputs (17-32) and Outputs (9-16). Example:* ***Set Output(s) 3,9,16*** *represents Standard Output 3 (OUT=4) and Extended Outputs 9 & 16 (OUTX=129).*

5.9 Spline Commands

5.9.1 CAM Move, Forward

This command begins the CAM motion in the positive direction with respect to the target position of the spline table. A CAM table is a series of master positions and relative position points. (Master position points are based on a single CAM rotation from 0° to 360°) The target position (TPOS) will pass through the spline table position at the specified master position. Since there are 4 spline tables available, the user must identify the spline table number and set the scale. The scale is used to adjust the CAM position. One rev of Master rotation is based on the variable SPPR. Changing SPPR will change the number of master pluses per revolution.

Range: 16 - 98 data points Intelligent Drive, (0,0) is always the starting point for a CAM move.

Scale = 2 - The CAM position is multiplied by 2 (The CAM motor will move twice as far per one rotation of master)

Scale = 0.5 - The CAM position is multiplied by 0.5 (The CAM motor will move 1/2 as far per one rotation of master)

Example: (Also See Example 5)

CAM Spline table (See Curve Below)

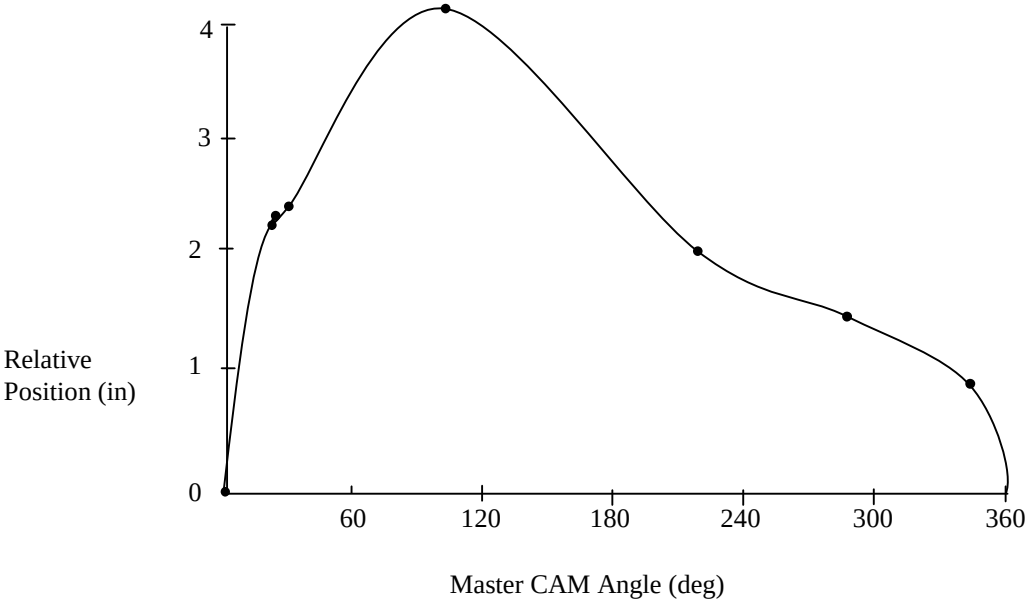
CAM Position (degrees)	0°	20°	20.5°	22°	110°	220°	290°	360°
CAM (Rotation) CPOS	0	0.055	0.057	0.061	0.306	0.611	0.806	1
Motor Position (in)	0	2.2	2.3	2.4	4.1	1.8	1.5	0.8

Clear Spline Table 2
Add to Spline Table 1 [0,0,0.055,2.2,0.057,2.3,0.061,2.4] (in)
Add to Spline Table 1 [0.306,4.1,0.611,1.8,0.806,1.5,1,0.8] (in)
Loop Spline Table 2 Forever
CAM Move FWD, Table 2, Scale=1

The ***"Input 3 and 4 are QUAD ENCODER inputs in Position Mode"*** must be checked in I/O Wizard for the CAM to function.

	NOTE
	*The drive employs cubic-spline interpolation between points. Cubic-spline interpolation fits (in real time) a third order polynomial contour between each set of points to insure smooth continuous motion The speed of the CAM is directly related to the frequency of the step pulse as set in SPPR.*

The above example clears the spline then adds the new values for Master-CAM-position and relative position. The move profile looks as follows:



5.9.2 CAM Move, Reverse

This command begins the CAM motion in the negative direction with respect to the target of the spline table. A CAM table is a series of master positions and relative position points. (Master position points are based on one CAM rotation from 0° to 360°) The target position (TPOS) will pass through spline table position at the specified master position. Since there are 4 spline tables available, the user must identify the spline table number and set the scale. . The scale is used to adjust the CAM position. One rev of Master rotation is based on the variable SPPR. Changing SPPR will change the number of master pluses per revolution.

Range: 16 -98 data points Intelligent Drive, (0,0) is always the starting point for a CAM move.

Scale = 2 - The CAM position is multiplied by 2 (The CAM motor will move twice as far per one rotation of master)

Spline = .5 - The CAM position is multiplied by 0.5 (The CAM motor will move 1/2 as far per one rotation of master)

Example:

CAM Spline table

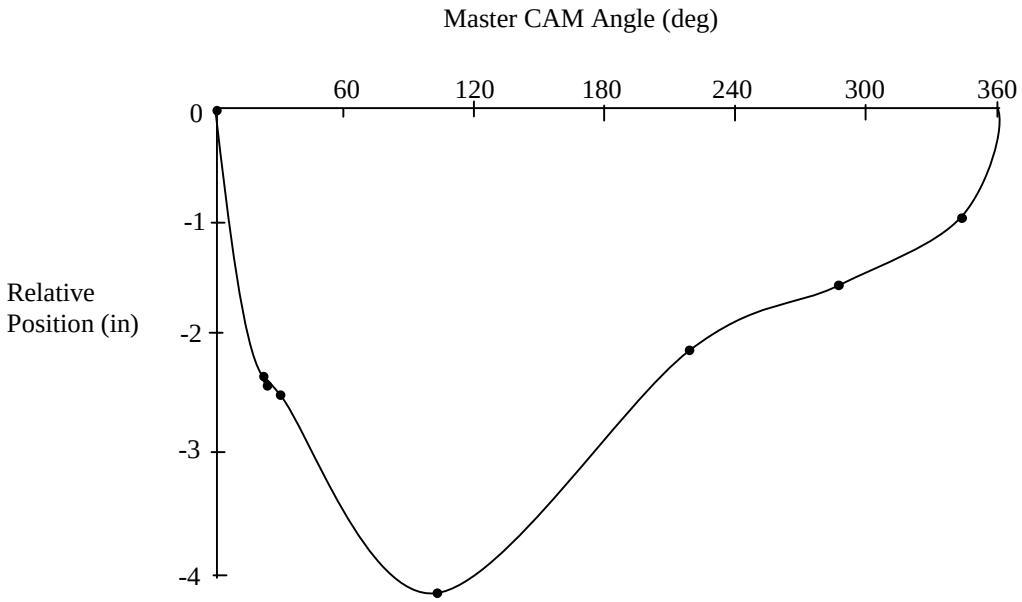
CAM Position (degrees)	0°	20°	20.5°	22°	110°	220°	290°	360°
CAM (Rotation) CPOS	0	0.055	0.057	0.061	0.306	0.611	0.806	1
Motor Position (in)	0	2.2	2.3	2.4	4.1	1.8	1.5	0

Clear Spline Table 1
Add to Spline Table 1 [0,0,0.055,2.2,0.057,2.3,0.061,2.4] (in)
Add to Spline Table 1 [0.306, 3.3,0.611,1.8,0.806,1.5,1,0.8] (in)
Loop Spline Table 2 Forever
CAM Move REV, Table 1, Scale=1

The ***"Input 3 and 4 are QUAD ENCODER inputs in Position Mode"*** must be checked in I/O Wizard for the CAM to function.

	NOTE
	*The drive employs cubic-spline interpolation between points. Cubic-spline interpolation fits (in real time) a third order polynomial contour between each set of points to insure smooth continuous motion The speed of the CAM is directly related to the frequency of the step pulse as set in SPPR.*

The above example clears the spline then adds the new values for Master-CAM-position and relative position. Profile as follows:



5.9.3 Loop Spline Table

This command loops the specified Spline/CAM table a given number of times. Normally a Spline is executed affixed number of times while a CAM is repeated forever. The ***Loop Spline Table*** command will execute the spline table a specified number of times. The user must enter the spline table number and number of loops.

Example:

Clear Spline Table 3
Add to Spline Table 3 [0,0,.1,2.2,.3,4.6,1.2,-12.3] (system-rev)
Add to Spline Table 3 [1.5,-8,3, 22.4,3.5,12] (system-rev)
Loop Spline Table 1, 3 times
Spline FWD, Table 3, Scale=1
Wait for In Position

The above example clears the spline then adds the new values for time/master-position and position before executing the spline move 4 times. (One time through plus three loops.)

5.9.4 Loop Spline Table Forever

This command loops the specified spline table forever. A Spline/CAM table can be executed a fixed number of times or repeat for ever. The ***Loop Spline Table Forever*** command will execute the spline table continuously until the program is ended. This command is generally used with the ***Queue Motion Command*** in order to monitor I/O and outside events for program termination.

Example: (Also See Examples 4 & 5)

```
        Clear Spline Table 3
        Add to Spline Table 3  [0,0,.1,2.2,.3,4.6,1.2,-12.3] (system-rev)
        Add to Spline Table 3  [1.5, -8, 3,22.4,3.5,12] (system-rev)
        Queue Motion Command
        Loop Spline Table 3 Forever
        Spline FWD, Table 3, Scale=1
TEST:
        If Input 5 is ON, Go To AA
        Go To TEST
AA:
        Queue Motion: Soft Stop
        End Program (0)
```

The above example clears then adds spline table 3 and puts the program in queue motion mode. The spline table will be looped continuously unless Input 5 becomes active at which point the motor will decelerate and the program will end.

5.9.5 Set Linear Spline Mode

This command sets the spline move using linear interpolation with respect to the target position in the spline table. The drive target position (TPOS) will pass through spline table position at the specified time using a linear interpolation move profile as opposed to a contoured third order polynomial as with a standard Spline or CAM move.

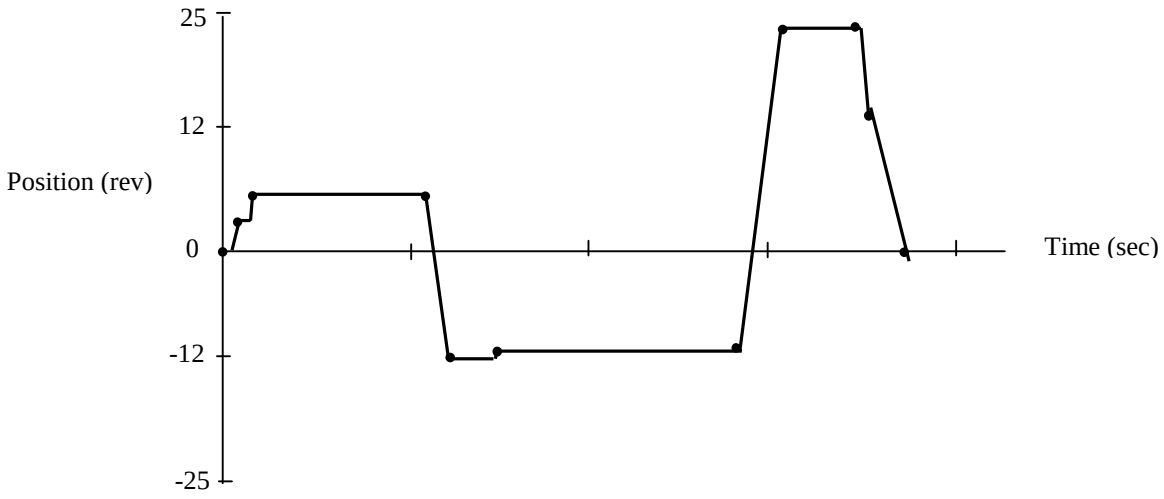
Example: (Also See Examples 4)

Spline table

Time (sec)	0.0	0.1	0.3	1.1	1.2	1.5	2.8	3.0	3.3	3.5	3.7
Position (in)	0.0	2.2	4.6	4.6	-12.3	-12	-12	22.0	22.0	12	0.0

Reallocate Spline Table 3, 30 entries
Clear Spline Table 3
Add to Spline Table 3 [0,0,.1,2.2,.3,4.6,1.1,4.6,1.2,-12.3] (in)
Add to Spline Table 3 [1.5,-12,2.8,-12,3,22,3.3,22 3.5,12,3.7,0]
Set Linear Spline Mode, Table 1
Spline FWD, Table 3, Scale=1
Wait for In Position

The above example clears the spline then adds the new values for time/master-position and position, before executing the Linear Spline. The move profile looks as follows:



5.9.6 Spline Move, Forward

This command begins the spline motion in the positive direction with respect to the target position in the spline table. The drive target position (TPOS) will pass through spline table position at the specified time. Since there are 4 spline tables available, the user must identify the spline table number and set the scale. The scale is used to adjust the speed at which the spline table is executed.

Scale = 2 - The times in the spline table are divided by 2. (Move time half as long)

Scale = 0.5 - The times in the spline table are divided by 0.5. (Move time is twice as long)

Example: (Also See Examples 3 & 4)

Spline table

Time (sec)	0.0	0.1	0.3	1.2	1.5	3.0	3.5	3.8
Position (revs)	0.0	2.2	4.6	-12.3	-8.0	22.4	12.0	0.0

Spline table (Scale=2) Actual commanded motion.

Time (sec)	0.0	0.05	0.15	0.6	0.75	1.5	1.75	1.9
Position (revs)	0.0	2.2	4.6	-12.3	-8.0	22.4	12.0	0.0

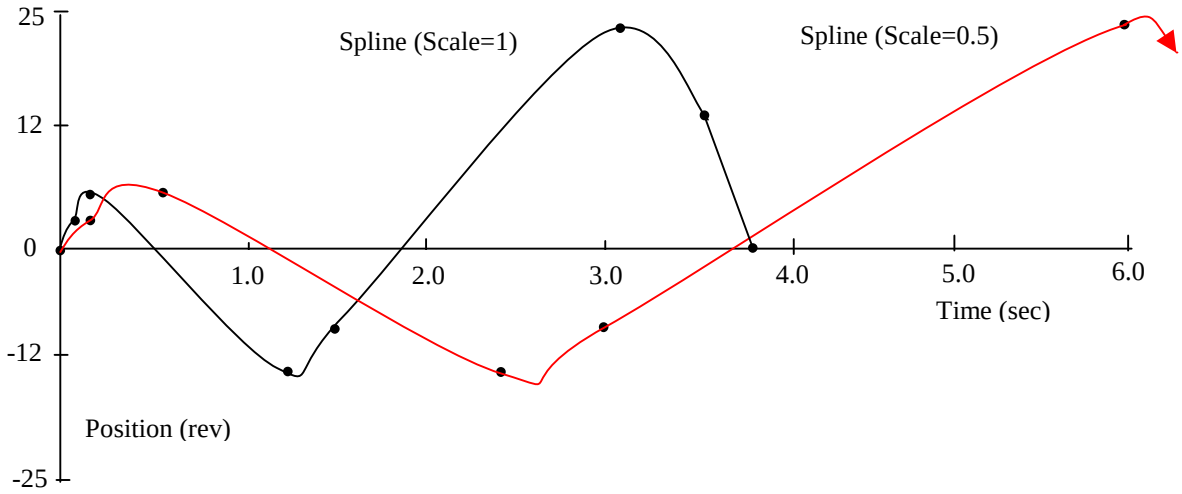
Spline table (Scale=0.5) Actual commanded motion.

Time (sec)	0.0	0.2	0.6	2.4	3.0	6.0	7.0	7.4
Position (revs)	0.0	2.2	4.6	-12.3	-8.0	22.4	12.0	0.0

Clear Spline Table 3
Add to Spline Table 3 [0,0,.1,2.2,.3,4.6,1.2,-12.3] (system-rev)
Add to Spline Table 3 [1.5,-8 ,3,22.4,3.5,12,3.8,0] (system-rev)
Queue Motion Command
Spline FWD, Table 3, Scale=1
Wait for In Position

	NOTE
	*The drive employs cubic-spline interpolation between points. Cubic-spline interpolation fits (in real time) a third order polynomial contour between each set of points to insure smooth continuous motion..*

The above example clears the spline then adds the new values for time/master-position and position. The move profile looks as follows:



5.9.7 Spline Move, Reverse

This command begins the spline motion in the negative direction with respect to the target position in the spline table. The drive target position (TPOS) will pass through negative of the spline table position at the specified time. Since there are 4 spline tables available, the user must identify the spline table number and set the scale. The scale is used to adjust the speed at which the spline table is executed.

Scale = 2 - The times in the spline table are divided by 2. (Move time half as long)

Scale = 0.5 - The times in the spline table are divided by 0.5. (Move time is twice as long)

Example: (Also See Example 3)

Spline table

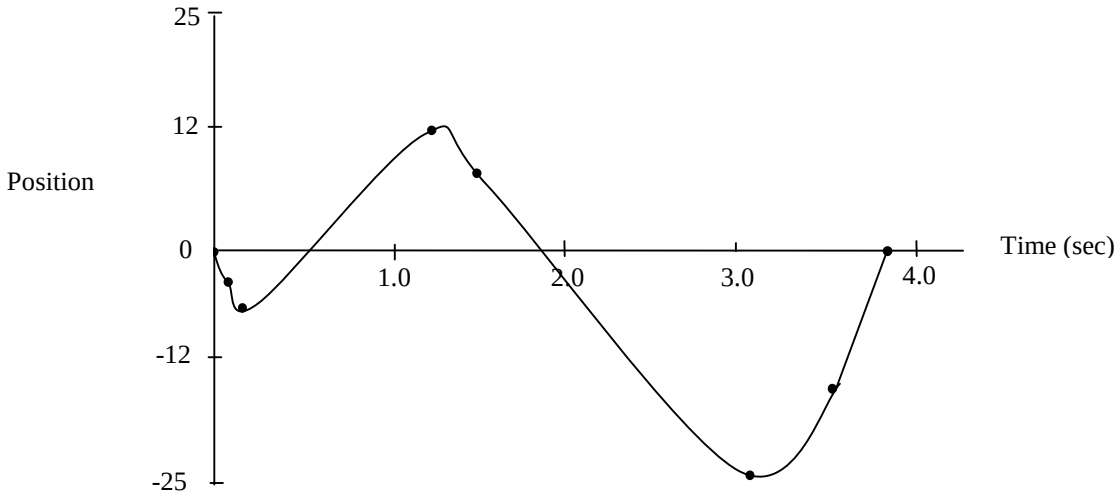
Time (sec)	0.0	0.1	0.3	1.2	1.5	3.0	3.5
Position (revs)	0.0	2.2	4.6	-12.3	-8.0	22.4	12.0

Reverse Spline Table (Scale=1) Actual commanded motion.

Time (sec)	0.0	0.1	0.3	1.2	1.5	3.0	3.5
Position (revs)	0.0	-2.2	-4.6	12.3	8.0	-22.4	-12.0

```
Clear Spline Table 0
Add to Spline Table 0  [0,0,.1,2.2,.3,4.6,1.2,-12.3 ] (system-rev)
Add to Spline Table 0  [1.5, -8, 3,22.4,3.5, 12,3.8,0]
Queue Motion Command
Spline REV, Table 0, Scale=1
Wait for In Position
```

The above example clears the spline then adds the new values for time/master-position and position. The move profile looks as follows:



	NOTE
	The drive employs cubic-spline interpolation between points. Cubic-spline interpolation fits (in real time) a third order polynomial contour between each set of points to insure smooth continuous motion.

5.9.8 Spline Pre-Compute

This command pre-computes the internal interpolation values for the spline table. Normally, the first time a spline table is executed there is a slight delay in order to calculate the interpolation constants. Pre-computing the spline table will eliminate any delays the first time through the program. The user must enter the spline table number to pre-compute.

5.9.9 Spline Table Add

This command adds time/master-position or CAM position data to the spline table. There are 4 spline tables available (0-3). This command appends data to the specified table; therefore, more than one ***Spline Table Add*** command can be used with-in a program. The time/master-position and/or CAM position can also be entered as expressions.

[T1*0.5,P1*1.5+8,T2*4.2,P2^2+4, ect]

Example: (Also See Examples 3, 4 & 5)

Add to Spline Table 3 [0,0,.1,2.2,.3,4.6,1.1,4.6,1.2,-12.3] (in)
Add to Spline Table 3 [1.5,-12,2.8,-12,3,22,3.3,22 3.5,12,3.8,0]

	NOTE
	*The time /master position values must be added to the table in ascending order. (It is not possible to enter the position for 3.0 sec after entering the position for 5.1 sec.) The **Spline Table Add** command is limited to 50 characters between brackets"[]". You will need to use multiple add spline tables to enter a large table*

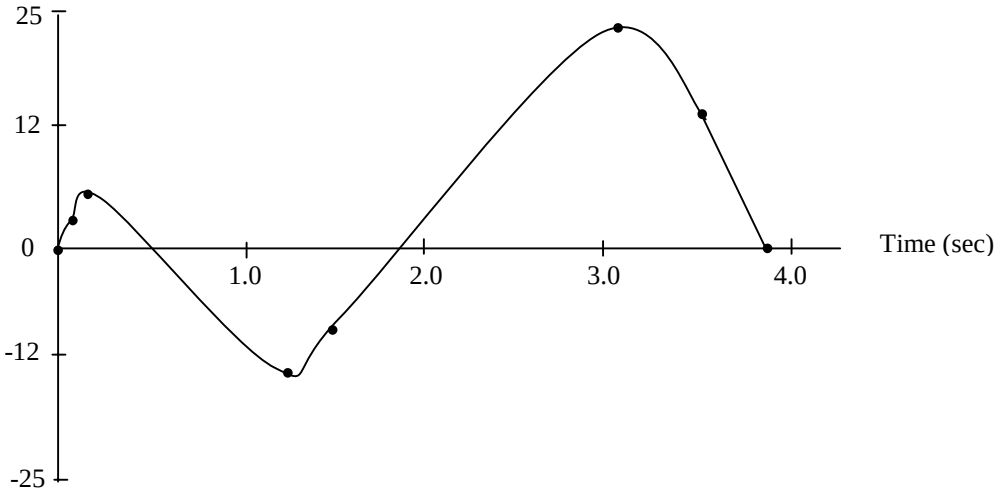
Example: (Also See Examples 3, 4 & 5)

Spline table

Time (sec)	0.0	0.1	0.3	1.2	1.5	3.0	3.5	3.8
Position (revs)	0.0	2.2	4.6	-12.3	-8.0	22.4	12.0	0.0

```
Clear Spline Table 0
Add to Spline Table 0  [0,0,.1,2.2,.3,4.6,1.2,-12.3] (system-rev)
Add to Spline Table 0  [1.5,-8 3,22.4,3.5, 12] (system-rev)
Spline FWD, Table 1, Scale=1
Wait for In Position
```

The above example clears the spline then adds the new values for time/master-position and position. The move profile looks as follows:



5.9.10 Spline Table Clear

This command clears the specified spline table. There are 4 spline tables available (0-3). The spline table should always be cleared before adding to it. If the table is not cleared, the new time/position values are added to the end of the table resulting in a software error when the amplifier's spline table limit is exceeded. The user must enter the spline table number to clear.

Example: (Also See Examples 3, 4 & 5)

```
Clear Spline Table 2
Add to Spline Table 2  [0,0,0.1,2.2,0.3,4.6,1.2,-12.3] (in)
Add to Spline Table 2  [1.5,-8.0,3.0,22.4,3.5,12.0,3.8,0.0] (in)
Spline FWD, Table 2, Scale=1
Wait for In Position
```

The above example clears the spline table before entering new values and executing the move.

5.9.11 Spline Table Re-Allocate

This command modifies the size of the spline table. Normally the spline table will default to 16 entries for Intelligent Drive (8 Master-time entries, 8 position entries). This command allows the programmer to increase the size of table so that larger tables can be entered. A fault will occur if the drive cannot allocate enough memory to hold all the points in the table. The user must specify the table number and table size.

Example: (Also See Examples 3, 4 & 5)

```
Re-allocate Spline Table 98 entries
```

When executed, this line will increase the spline table size to 98 entries. (49 Master-time or CAM position entries, 49 position entries)

6. ASCII Protocol

This section covers the ISAP (Intelligent Servo ASCII Protocol) for use with the Intelligent Servo drive. The protocol allows the user to send pure ASCII commands through the RS-232, 422 or 485 serial port directly to the servo drive. By using a node address, multiple axes can be commanded through one serial port without using a "Checksum" to verify if the command was properly received. ASCII Drive Interface

This method is not intended as a high-speed network solution but rather a method to transfer information to the drive. Although it is ideal for updating variable or parameters on the fly as well as loading and running stored programs, the baud rate of 19.2 KB and ASCII protocol limits the speed information can be transferred.

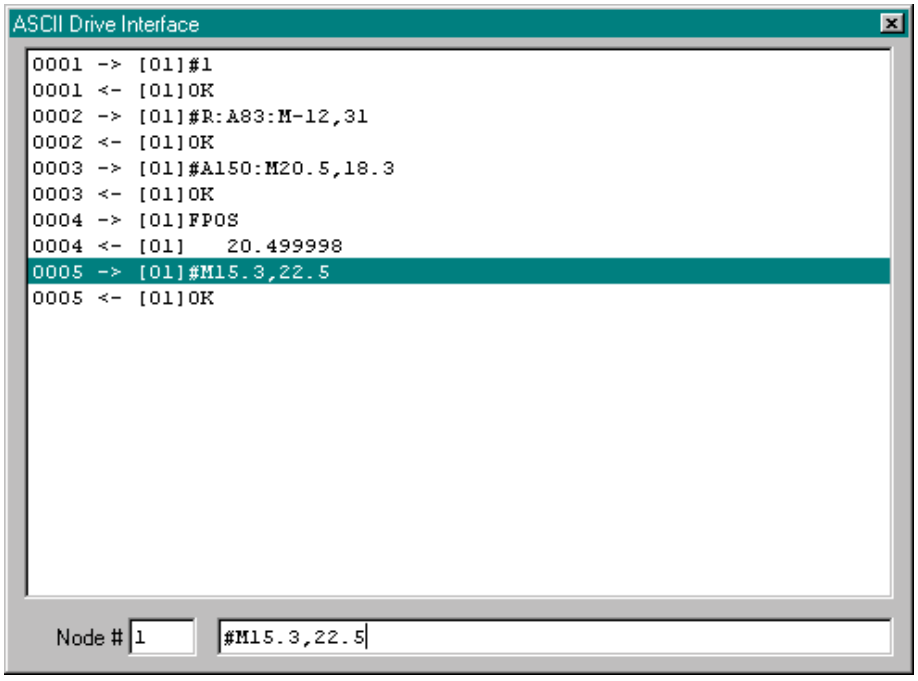
There are four basic command formats, which will be discussed:

1. **Parameters** - Retrieve value of a parameters
2. **Parameters = Value** - Sets the value of a parameter
3. **#Motion Command** - Executes a specific IML (Intelligent Motion Language) "Motion" command
4. **/Special Command** - Executes special commands, which mirror some specific IML commands.

The preferred method to interface with the servo drive is to use the "Intelligent Motion Language" with the DLL files that are included on the **Tec Tools** disc. This protocol provides a checksum that verifies the command was correctly received. The ISAP protocol was added for applications such as PLCs and terminal interfaces where a checksum could not be utilized.

6.1 ASCII Drive Interface

To open the ASCII drive interface click “Special” button on the menu bar. The interface window provides a simple interface to test the ASCII commands. The ASCII drive interface provides a simple easy to use terminal to test ASCII strings. The Node number is entered in the bottom left corner of the window and is not required to be sent with the ASCII string when using the interface.



An ASCII terminal interface screen is provided with **Tec Tools** and can be used to test ASCII commands. The drive node number [nn] is set at the bottom left corner of the ASCII terminal window and does not need to proceed the command as required with other ASCII interfaces. To talk to multiple axes the node number [nn] must be changed before sending the command.

	NOTE
	*Once the ACSII Drive interface is opened,* **Tec Tools** *no longer functions. The program pointer will stop moving and the monitor window will not update. Since the ASCII command structure does not provide a checksum for handshaking,, a communication error can cause data to be misinterpreted.*

6.2 Message Format

6.2.1 Outgoing ISAP packet

The outgoing packet from the controller to the drive is in the following format:

Format: [nn]command<CR>

[nn] is the ASCII representation of the drive node number.
"**command**" is the ASCII string of characters for the specified command
<CR> represent the "carriage return" ASCII character

Note: The command string must be received within 1 sec or the buffer is cleared and the command is not accepted.

Example:

SEND	[01]FPOS	- Request Feedback Position
RESPONSE	[01]4.127197	- Response is Feedback
SEND	[01]#1	- Turn on immediate mode.
RESPONSE	[01]OK	- Response OK

Response: OK – Command accepted
?? – Command **not** accepted

6.2.2 Incoming ISAP packet

The incoming packet from the drive to the controller is in the following format:

Format: [nn]response<CR>

[nn] is the ASCII representation of the drive node number.
"**response**" is the ASCII string of characters for the specified response.
<CR> represent the "carriage return" ASCII character

Example:

SEND	[01]/LTEST1	- Load program TEST1 (Node 1)
RESPONSE	[01]OK	- Response OK
SEND	[04]MOVE1= 10.15	- Set custom variable MOVE1 (Node 4)
RESPONSE	[04] 10.149999	- Response is value of variable

Response: OK – Command accepted
?? – Command **not** accepted

6.3 Parameters

6.3.1 Get Value of Parameter

The following format is used to get the value of an internal drive parameter or custom variable.

[nn]parameter

[nn] is the ASCII representation of the drive node number.

"**parameter**" is the ASCII string representing the drive parameter or variable.

Example:

SEND	[01]FPOS	- Request Feedback position (Node 1)
RESPONSE	[01]4.139987	- Response in position units *
SEND	[02]V1= 12	- Set custom variable V1 (Node 2)
RESPONSE	[02]12.00000	- Response is value of variable

***** The response for the feedback position is in the units that the drive is configured for in the program properties. (system-revs, inches, cm, etc.)

6.3.2 Set Value of Parameter

The following format is used to set the value of an internal drive parameter or custom variable. Parameters and variables can be changed on the fly while a program is running within the drive.

[nn]parameter=value

[nn] is the ASCII representation of the drive node number.

"**parameter**" is the ASCII string representing the drive parameter or variable.

"**value**" is the ASCII string which the parameter is set to

Example:

SEND	[01]V1=12.24	- Set variable V1=12.24
RESPONSE	[01]12.240001	- Response is new value of variable
SEND	[01]V2=V1+20	- Set V2 equal to V1+20
RESPONSE	[01]32.24000	- Response is new value of variable
SEND	[01]SWE=0	- Software Disable drive
RESPONSE	[01]SWE=1	- Software Enable drive

Changing the drive parameters can be used to enable/disable the drive, change drive modes, commanded velocity, commanded current etc. Below are a few useful parameters that can be set.

SWE=(0 or 1)	- Enables and Disables drive.
DCC=(0-IRMS)	- Changes the command drive current when in "Current" mode and "Command Source" (DCM) is digital.
DCV=(+/-15000)	- Changes the command drive velocity when in "Velocity" mode and "Command Source" (DCM) is digital.
CM=(0 – 6)	- Changes the command mode. Drive must be disabled before changing modes. (See ToolPAC manual for modes)
DCM=(0 or 1)	- Command source (Analog/Digital) when in velocity or current mode.
FSV=(0.01 –100000)	- Velocity at 10V Analog when in analog velocity mode.
IMAX=(Per Motor)	- Change maximum allowed current to motor. (Peak Current)
TR=(+/-16384)	- Changes the translation ration between system-revs and user units.
IOCW=(0-4096)	- Changes I/O configuration.
OUT	- Output status
IN	- Input status
TBAS= (0 – 1)	- Changes the "Time Base"
SPHZ=(0 – SMC)	- Changes "Slave" phase adjust.

For a complete list of variables refer to the **Section 4**.

6.4 Motion Commands

The motion commands are a set of IML (Intelligent Motion Language) commands that can be sent in ASCII format for use in the "Immediate" mode. The commands are standard drive commands and are preceded by a "#" sign. The drive ***must*** be in immediate mode (***not*** running a program) before any command preceded by a "#" can be issued.

Format: [nn]#motion command

[nn] is the ASCII representation of the drive node number.

"**#**" is an ASCII "pound sign" character

"**motion command**" is the ASCII command string.

	NOTE
	*The "IML" commands are case sensitive. Some commands use uppercase letters while other commands are in lowercase. Motion commands cannot be executed while a program is running.*

Eample:

SEND	[01]#M15.6,20	- Begins absolute move then resets feedback.
RESPONSE	[01] OK	- Response OK

The above example will begin an absolute move to 15.6 *units* at 20 *units*/sec.

SEND	[01]#A4000	- Set Acceleration
RESPONSE	[01]OK	- Response OK
SEND	[01]#M23.45,10	- Make Absolute move .
RESPONSE	[01]OK	- Response OK

The above example Sets Acceleration to 4000 *units*/sec/sec then makes an absolute move of 23.45 *units* at 10 *units*/sec.

Correct String

SEND	[01]#M12.0,20	- Issue Absolute move command.
RESPONSE	[01]OK	- Response OK
SEND	[01]/Q	- Query drive status
RESPONSE	[01]E+F-R-M+	- Drive Enabled and Motor moving.
SEND	[01]/Q	- Query drive status
RESPONSE	[01]E+F-R-M+	- Drive Enabled and Motor moving.
SEND	[01]/Q	- Query drive status
RESPONSE	[01]E+F-R-M-	- Drive Enabled and Motor not moving.
SEND	[01]#M-5.6	- Set Acceleration the makes Absolute move.
RESPONSE	[01]OK	- Response OK

The above example begins an absolute move to 12.0 *units* at 20 *units*/sec. The drive status is queried to determine if motion is complete (M-). Once motion is complete the next move command is issued.

6.4.1 Immediate Mode (ON)

This command turns on the "Immediate Mode" which allows the drive to accept the ASCII motion commands. While in immediate mode the program cannot run a stored program. While running a program the drive can not be switched to immediate mode.

Format: [nn]#1

Response: [nn]OK

6.4.2 Immediate Mode (OFF)

This command turns off the "Immediate Mode" which does not allow the drive to accept the ASCII motion commands. While in immediate mode stored programs can not run. While running a program the drive can not be switched to immediate mode.

Format: [nn]#0

Response: [nn]OK

6.4.3 Immediate Mode Status

This command turns check the status of the “Immediate Mode” to see if it

Format: [nn]#=

Response: [nn]0 or [nn]1 0 = Off, 1 = On

6.4.4 Set Acceleration

This command set the motor acceleration and deceleration rates for “Position mode”. The <value> can be an expression and is in ***units***/sec/sec where units are defined as system-revs multiplied by the translation ratio (TR).

Format: [nn]#A<value>

Response: [nn]OK

Example:

SEND	[01]#A4000	- Set acceleration to 4000 units/sec/sec
RESPONSE	[01]OK	- Response OK

or

SEND	[02] #AV1	- Set acceleration to variable V1 (Node 2)
RESPONSE	[02]OK	- Response OK

6.4.5 Move to Position

This command rotates the motor to an absolute position. The move profile is triangular (Constant acceleration for ½ the distance, followed by constant deceleration for the remaining ½.) The value can be an expression and is in ***units*** where units are defined as system-revs multiplied by the translation ratio (TR).

Format: [nn]#M<absolute position>

Response: [nn]OK

Example:

SEND	[01]#M20.5	- Moves to position 20.5 revs.
RESPONSE	[01]OK	- Response OK

or

SEND	[02] #MTPOS+20.5	- Moves 20.5 revs from current position.
RESPONSE	[02]OK	- Response OK

or

SEND	[02]M20.5	- Moves to 20.5 revs.
RESPONSE	[02]OK	- Response OK

The first example moves to an absolute position of 20.5 revs. The second example makes a relative move of 20.5 revs while maintaining the feedback position. The third example moves a distance of 20.5 revs.

	NOTE
	*Before sending a second move command the drive must be queried to determine if motion is complete (M-). A software fault (SF) will result if a second move command is issued while motor is moving*

6.4.6 Move to Position, Speed Limited

This command rotates the motor to an absolute position at the specified velocity. The move profile is trapezoidal using the last preset acceleration. If the acceleration rate and end position is specified such that the maximum velocity is not reached, then the move profile is triangular. The <absolute position> and <max velocity> can be an expression and are in ***units*** and ***units***/sec respectively, where units are defined as system-revs multiplied by the translation ratio (TR).

Format: [nn]#M<absolute position>,<max velocity>

Response: [nn]OK

Example:

SEND	[01]#M-12.5,25	- Moves to position –12.5 revs at 25 revs/sec
RESPONSE	[01]OK	- Response OK

or

SEND	[01]#MTPOS-12.5,V1	Moves -12.5 revs from current position at V1 rev/sec .
RESPONSE	[01]OK	- Response OK

or

SEND	[01]#M-12.5,25	- Moves to position -12.5 revs at 25 rev/sec
RESPONSE	[01]OK	- Response OK

The first example moves to an absolute position of -12.5 revs at 25 rev/sec. The second example makes a relative move of -12.5 revs at V1 rev/sec while maintaining the feedback position. The third example then moves a distance of -12.5 revs at 25 rev/sec.

	NOTE
	*Before sending a second move command the drive must be queried to determine if motion is complete (M-). A software fault (SF) will result if a second move command is issued while motor is moving*

6.4.7 Move to Position, Time Limited, Triangular

This command rotates the motor to an absolute position within the specified time. The move profile is triangular (Constant acceleration for ½ the specified time, followed by constant deceleration for the remaining ½.). If the end position and time is specified such that the maximum drive current is reached, then the move profile is trapezoidal The <absolute position> in ***units*** and <max velocity> in seconds can be an expression, where units are defined as system-revs multiplied by the translation ratio (TR).

Format: [nn]#N<absolute position>,<time>

Response: [nn]OK

Example:

SEND	[01]#N16.8,25	- Move absolute position 16.8 revs in 2.5 sec
RESPONSE	[01]OK	- Response OK

or

SEND	[01] #NTPOS+16.8,V1	Moves 16.8 revs from current position in V1 seconds.
RESPONSE	[01]OK	- Response OK

or

SEND	[01]N16.8,2.5	- Moves to position 16.8 revs in 2.5 seconds.
RESPONSE	[01]OK	- Response OK

The first example moves to an absolute position of 16.8 revs in 2.5 seconds. The second example makes a relative move of 16.8 revs within V1 seconds while maintaining the feedback position. The third example moves a distance of 16.8 revs in 2.5 seconds.

	NOTE
	*Before sending a second move command the drive must be queried to determine if motion is complete (M-). A software fault (SF) will result if a second move command is issued while motor is moving*

6.4.8 Set Output

This command sets a specified output or outputs. An output <list> consists of a series of numbers separated by commas (,).

Format: [nn]#OS<list>

Response: [nn]OK

Example:

SEND	[03]#OS3,5,6	- Sets Outputs 3, 5 and 6 (Node 3)
RESPONSE	[03]OK	- Response OK
SEND	[03]OUT	- Checks status of Outputs (Node 3)
RESPONSE	[03] 32.000000	- Value of Set Outputs

	NOTE
	*To check the status of the outputs get drive parameter "OUT" which represents the binary output value in decimal format.*

6.4.9 Clear Output

This command resets a specified output or outputs. An output <list> consists of a series of numbers separated by commas (,).

Format: [nn]#OC<list>

Response: [nn]OK

Example:

SEND	[03]#OC3,5,6	- Clears Outputs 3, 5 and 6 (Node 3)
RESPONSE	[03]OK	- Response OK
SEND	[03]OUT	- Checks status of Outputs (Node 3)
RESPONSE	[03]0.000000	- Value of Set Outputs

	NOTE
	*To check the status of the outputs get drive parameter "OUT" which represents the binary output value in decimal format.*

6.4.10 Zero and Relax

This command zeros the feedback (FPOS), target position (TPOS) and following error (PERR).

Format: [nn]#Z

Response: [nn]OK

Example:

SEND	[01]#Z	- Zero the FPOS, TPOS and PERR
RESPONSE	[01]OK	- Response OK
SEND	[01]FPOS	- Request Feedback Position
RESPONSE	[01] 0.000000	- Feedback position
SEND	[01]TPOS	- Request target Position
RESPONSE	[01] 0.000000	- Target Position
SEND	[01]PERR	- Request Position Error
RESPONSE	[01] 0.000000	- Position Error

6.4.11 Zero

This command zeros the target position (TPOS) but maintains the following error (PERR) while selecting the appropriate feedback position. (FPOS)

Format: [nn]#R

Response: [nn]OK

Example:

SEND	[01]#R	- Zero the FPOS and TPOS
RESPONSE	[01]OK	- Response OK
SEND	[01]FPOS	- Request Feedback Position
RESPONSE	[01] -0.325647	- Feedback position
SEND	[01]TPOS	- Request target Position
RESPONSE	[01] 0.000000	- Response OK
SEND	[01]PERR	- Request Position Error
RESPONSE	[01] 0.325642	- Position Error

6.5 Special Commands

The special commands are preceded by a forward slash (/) and can be executed when a program is running. These commands mirror some specific IML commands that allow the user to load, run and stop program execution as well as querying the drive for status.

6.5.1 Program Library Directory

This command lists all the programs with-in the "Program Library". Each program in the directory is separated by a semicolon (;).

Format: [nn]/D

Response: [nn]$$SAVE$$;<pgm1>;<pgm2>;<pgm3>;<pgm4>;etc

Example:

SEND [01]/D - Get Program Library
RESPONSE [01]$$SAVE$$;TEST1;TEST2;PART1;PART22;

	NOTE
	*Spaces in program names will result in the directory cutting off all text after the space.* Example: Part 1 and Part 2 becomes [01]$$SAVE$$;PART;PART $$SAVE$$ - Represents the active program in the program library.

6.5.2 Load Program

This command loads a specified program from the program library to the active program.

Format: [nn]/L<program name>

Response: [nn]OK

Example:

SEND [01]/LPART1 - Loads program "PART1" from library
RESPONSE [01]OK - Response OK

6.5.3 Run Program

This command runs the "Active" program in the specified node.

Format: [nn]/R

Response: [nn]OK

Example:

SEND	[01]SWE=1	- Enable drive
RESPONSE	[01]OK	- Response OK
SEND	[01]/R	- Run active program in Node 1.
RESPONSE	[01]OK	- Response OK

6.5.4 Stop Program

This command stops the execution of the "Active" program in the specified node.

Format: [nn]/S

Response: [nn]OK

Example:

SEND	[04]/S	- Stop active program in Node 4.
RESPONSE	[04]OK	- Response OK

6.5.5 Calabrate Analog Inputs

This command calibrates the analog inputs to zero by creating a calibration offset, which is added to the analog signal. Remember to set you analog source to zero before calibrating.

Format: [nn]/A

Response: [nn]OK

Example:

SEND	[04]/A	- Stop active program in Node 4.
RESPONSE	[04]OK	- Response OK

6.5.6 Query Drive Status

This command queries specified node for drive status.

Format: [nn]/Q

Response: [nn]E<code>F<code>R<code>M<code>

E - Drive enable status
F - Drive fault status
R - Drive running program
M - Motor Moving (Position Mode Only)

Where <code> is: + true/yes
- false/no

Example:

SEND	[04]/Q	- Query drive status
RESPONSE	[04]E+F-R+M-	- Drive Enable, No Faults, Program Running, Motor not moving.

6.5.7 Reset Command

This command resets the drive to clear faults. If the fault condition exists, the fault will be cleared, drive will be disabled and program will no longer be running.

Format: [nn]/F

Response: [nn] OK

Example:

SEND	[04]/F	- Reset Drive
RESPONSE	[04] OK	- Reset command executed.

7. Arithmetic Functions

This section discusses the math functions available with-in the programable drives.

The Arithmetic Functions are available for creating math expressions within a program. Multiple math functions can be used on one command line.

+	- Addition
-	- Subtraction
*	- Multiplication
/	- Division
=	- Equals
!=	- Not Equal
<	- Less Than
>	- Greater Than
<=	- Less Than or Equal To
>=	- Greater Than or Equal To

^ - Raised to the power of

Example: X1=Y1^5
(X1 will equal the Y1 to the fifth power)

( ) - Parenthesis for enclosing mathematical arguments,

Example: V1=FPOS*(PERR+1.4)

SIN(x) - Sine Function **(Radians)**

Example: CAPTURE=***SIN***(FPOS *0.2)

(Capture will equal the sine of 0.2 times the Feedback Position)

SINH(x) - Hyperbolic Sine Function **(Radians)**

Example: CAPTURE=***SINH***(FPOS*0.2)

(Capture will = the hyperbolic sine of 0.2 times the Feedback Position)

COS(x) - Cosine Function **(Radians)**

Example: V1=***COS***(0.8456)*FPOS

(V1 will equal the Feedback Position times cosine of 0.8455)

COSH(x) - Hyperbolic Cosine Function **(Radians)**

Example: V1=***COSH***(0.8456)*FPOS

(V1 will equal the Feedback Position times hyperbolic cosine of 0.8455)

TAN - Tangent Function **(Radians)**

Example: V1=***TAN***(0.23146)*PERR

(V1 will equal the Position Error times tangent of 0.2314)

TANH(x) - Hyperbolic Tangent Function **(Radians)**

Example: V1=***TANH***(0.23146)*PERR

ABS(x) - Absolute Value

Example: V2=***ABS***((0.23146)*FPOS)

(Assume FPOS= 25 then V2=|0.23146*25|=6)

INT(x) - Returns the integer potion

Example: V2=***INT***((0.23146)*FPOS)

(Assume FPOS= 25 then V2=|0.23146*25|=0.7865)

& - Logical AND

Example: If ON(2)**&**ON(4) Then TOP

(If Inputs 2 and 4 are on the program will jump to label TOP)

- Bitwise AND

Example: V1=IN&120 Looks only at Inputs 4-7

(If Inputs 2, 4 and 7 are on V1= 72)

| - Logical OR

Example: If ON(2)|ON(4) Then TOP

(If Input 2 or Input 4 are on the program will jump to label TOP)

~ - Logical NOT

Example: If (~IN &10) Then Go To END

(If Inputs 2 & 4 not active, go to label END)

Can also be expressed as If OFF (2) & OFF(4) Go To END

LOG(x) - Calculates the Natural Logarithm (base e)

Example: V3=***LOG***(FPOS)

(Assume FPOS= 25 then V2=Log(25)=3.2188758)

LOG10(x) - Calculates common Logarithm

Example: V3=***LOG10***(FPOS)

(Assume FPOS= 25 then V2=Log10(25)=1.397940)

EXP(x) - Calculates the Natural Antilogarithm (base e)

Example: V4=***EXP***(3. 218875 8) (V4=25)

TBL(x) - Returns the interpolated point from 'x' using the last spline table.

Example:

Spline table

Time (sec)	0.0	0.3	1.0	1.1	4.6	5.2	7.0
Position (revs)	0.0	2.2	-4.8	-6.7	25.6	12.7	2.1

1. Add to Spline Table 1: (0,0,0.3,2.2,1.0,-4.8,4.6,25.6,5.2,12.7,7.0,2.1) [rev] V1=TBL(0.4)

2. V1 would equal the interpolated value at 0.4 seconds, approximately 1.2 rev

3. Add to Spline Table 1: (0,0,0.3,2.2,1.0,-4.8,4.6,25.6,5.2,12.7,7.0,2.1) [rev] V1=TBL(Input&7)

V1 would equal the interpolated value based on the state of inputs 1-3. If inputs 2 and 3 are active then V1=TBL(6) which would be approximately 7.9 revs. else the value is 0

8. Amplifier LED Status Codes

This section covers the both "Fatal" and "Non-Fatal" LED drive faults which result in drive LED blinking the status codes.

8.1 Fatal Status Codes

Overspeed (oS)

The motor's velocity has exceeded the value in the OSPD (Overspeed) parameter. May be the result of issuing an Absolute or Indexed Move Shortest Time.

E/Quick-Stop (ES)

An event, communications or an input has caused a quick-stop or e-stop condition.

End Program Fault (SF)

The program has ended with a non-zero "End Program" value.

Bad Var/Label (SF)

- A variable specified in an expression does not exist.
- A variable specified in an expression is read-only, and cannot be changed.
- The target label of a "Go To" does not exist.
- The target label of a "Call" does not exist.

Bad Expression (SF)

There was an error attempting to evaluate an expression. The usual cause is an ill-formed expression, an expression with spaces or other illegal characters, or possibly missing, illegally specified or undefined variables.

Feedback Loss (FL)

A signal is missing for motor feedback.

Bad I/O Number (SF)

An illegal value was specified for an input or an output within a command. Each drive has a specified limited number of I/O points that are available for general use.

Bad Event I/O Number (SF)

An illegal value was specified for an input or output within an event. Each drive has a specified limited number of IO points that are available for general use.

Invalid Argument (SF)

An expression has evaluated to a value not legal for the command or context it is used. For example, a negative time in a command which needs a positive, non-zero time?

Spline Error (SF)

- The number of points in a spline table is less than 3; a minimum of three points are needed to make a spline table.
- The specified spline table does not exist. Depending on the drive's configuration, there are a limited number of spline tables available to load data. Typically, there are four such tables, numbered zero (0) through three (3).
- The number of points specified in a "Spline Table Re-Allocate" exceeds the available memory dedicated to spline tables.
- There is no space available in a spline table to add any more data points; this error occurs in the "Add to spline table" command. Either reduce the number of points, or use the "Re-Allocate" command to increase the size of the spline table.
- The "X" value in an "Add…" command is less than or equal to the last point which was added to the table. Spline table "X" values (either time or CAM position) must be increasing in value.

The "Run Time Faults" are generated by an "End Program" command or by a software fault and stored in system variable 'RERR'. Faults generated by the End Program command will always be forced to a negative value. The following error codes are generated due to software faults.

7000 Attempted to define a PLS with PLS's enabled
Attempted to define too many PLS's

7001 Arguments are incorrect in a PLS definition

7002 Attempted made to utilize a "Gear At" while in a "Gear At In" or a "Gear For In"

7003 Specified PLS output number is invalid

8000 Attempted to “Queue Motion” with a macro running

8001 Attempted to define a macro with a macro running
Attempted to define too many macro’s

8002 Illegal “time” specified in a “Wait” macro step
Attempted to start a macro while one is already running
Attempted to start a macro while one is being defined
Attempted to run a macro that doesn’t exist
Illegal “repeat count” specified in a “Macro Start”

8003 A variable was specified where a constant is required
A constant was specified where a variable is required
Illegal code specified for the input number in an “Arm” operation Illegal code specified for the action in a “Wait Capture” operation

8004 “Wait Capture” fault has been raised

8005 An illegal input or output number was specified

8006 Attempted to execute a motion command when not in position mode
Attempted to execute a motion command while in a “Move At”
Attempted to begin a “Relative” or “Absolute” move while currently executing another motion step

8007 Attempted to “Phase Delay”, “Phase Adjust”, “Gear At”, “Gear For At” or “Gear At In” while presently finishing execution of another “Phase Delay”, “Phase Adjust”, “Gear For At” or “Gear At In” operation

8008 Attempted to “Call” or “Chain” to an illegal or undefined macro

8009 “Call” stack overflow. The macro call stack was greater than 16 levels deep.

9001 General failure in a “Gearing” operation

9002 Occurs in a “Gear For At” command when slave displacement exceeds ½ of the master displacement.

Following Error (FE)

An event, which monitors following error, (the difference between the target position and the feedback position) has shut down the drive because the limit specified in the event has been exceeded. Either increase the value programmed in the event, or reduce the acceleration rate, speed or time-requirement for the programmed motion. The Following Error fault can also be caused by the Target or Feedback position registers over flow during operation.

Regen Fault (rF)

This fault will be displayed when the duty cycle of the regen resistor has been exceeded.

Illegal Operation (SF)

- An attempt was made to start or stop motion when the drive is not in position mode.
- A specified time period is negative or zero.
- A variable which was specified in a PID or “Slew Variable” command does not exist.
- An attempt was made to stop motion when the drive has executing a “Move At”. Such commands need to be terminated with a corresponding “End Move At” command.
- An attempt was made to use a “Hard/Soft Stop Queued Motion” command when the drive was not queuing motion.
- An attempt was made to enter “Master Position Tracking Mode” when the drive was not in position mode, or the drive was in a continuous move, or the drive was executing a queued motion command.
- Another “Slew Variable” instruction was encountered before the first had completed. Only one such command may be active at a time.
- “ Wait for In Position” was executed within a “Move At” block.
- An attempt was made to execute a spline or CAM command when the system was already executing one.
- Arguments specified in a “Move At” or related command is not valid.
- Arguments specified in a “Move” command do not make sense.
- Too many calls have occurred.

- A Return has occurred without a Call.
- A variable specified for DAC related commands does not exist.

Short Circuit Fault (SC)

- An amplifier fault has occurred in the power transistor stage.
- Verify motor wiring.
- Bus Fault (bF)
- A bus-related amplifier fault has occurred.
- Verify the incoming AC power. Are all three phases present?
- Is AC voltage above 90 VAC?
- Is the bus dropping below 90VDC during operation?
- Is bus voltage exceeding 390VDC during deceleration?
- Is external regeneration resistor present?

Amplifier Overtemp (At)

The drive shutdown because internal thermal limits have been exceeded.

Motor Overtemp (ot)

The motor over temperature thermal switch has been tripped. If the motor does not have such protection, change the value of the COT parameter, which controls whether or not the thermal switch input should be examined.

Over Current (OC)

The drive current exceeded 1.9 times the drive's peak current. Can be caused if the current loop gains are set too high and becoming unstable.

Firmware Fault (FF)

This error indicates a fault with the microprocessor in the drive.

Real-Time Loop Overrun (LO)

The execution of the real-time motor control exceeds the maximum limit for motor loop control.

Hall Loss (HL) (Encoder version)

The Hall Effect feedback is miswired or not connected. Occurs if the Hall Effect Device (HED) reads back a number below one or greater than six.

Power Supply Under Voltage (PS)

The drive bus voltage dropped below 80 VDC for the BDA-34xx or 18 VDC for the BDD-34xx Servo drives.

Watchdog Timeout (rd)

Firmware is not operating correctly. If occurred during a "***Firmware Upgrade***" reset the drive using the status screen and continue. **DO NOT** remove power during a firmware upgrade.

8.2 Nonfatal Status Codes

Pause (P)

The “Pause” input is active. (Non-blinking)

Limit (L)

The “Limit” input is active. (Non-blinking)

NN

Displays the node ID at Power-up. (Scrolling)

Disabled (d)

Indicates the drive is disable. The enable input is “Off” and/or the drive is disabled in software. (Non-blinking)

Enabled (E)

The “Pause” input is active. The enable input is “On” and/or the drive is software enabled. (Non-blinking)

9. Saving, Printing and Uploading Firmware

9.1 Uploading Firmware

- Connect communication cable, power up controller and start **Tec Tools** ™
- Disable the drive before beginning Firmware upgrade. If the controller is enabled when uploading new firmware, the drive will become enable, which may cause unexpected motion.
- Highlight the axis to be updated and click on "Special", then "Upload New Firmware…"
- Locate the firmware .BIN file to be uploaded and click on Open.

	NOTE
	*The I-servo firmware between the encoder and resolver versions is distinct and can not be interchanged. Loading the wrong version firmware will result in a feedback fault.*

Example:

I and d Servo Firmware is -R40D-00.bin *

* The firmware .BIN file name differs depending on revision level.

The firmware upload will take from 3 to 5 minutes.

If the upload takes more than five minutes verify the RS-232 connection. **Do Not** remove power from the controller.

	WARNING
	*Removing power during the upload process will result in the controller locking-up. The amplifier would need to be returned to the factory for repair. This is not covered under warranty.*

9.2 Saving Set-up Parameters to Disk

- Connect communication cable, power up controller and start **Tec Tools** ™

- Single click on plus sign (+) next to the axis to expand the view.

- To create a new application folder on your hard drive click on "File", "New Application" then enter the application name. The new application folder will be created under the **Tec Tools** ™ directory. (c:\program files\Tec Tools\application name.MDB) This file can then be saved to disk using Windows Explorer.

- Right mouse click on new application you created and select the correct drive type and enter an axis name.

- To save the setup parameters to the hard drive, right mouse click on

- "Parameters", then click on "Copy All Parameters To" the new axis you created.

	NOTE
	*It is recommended that the Program Libraries, Set-up Parameters and Events be backed up on disk in the event that the drive is not accessible.*

9.3 Saving Events from Controller to Disk

- Connect communication cable, power up controller and start **Tec Tools** ™

- Single click on plus sign (+) next to the axis to expand the view.

- To create a new application folder on your hard drive click on "File", "New Application" then enter the application name. The new application folder will be created under the **Tec Tools** ™ directory. (c:\programfiles\Tec Tools\application name.MDB) This file can then be saved to disk using Windows Explorer.

- Right mouse click on new application you created and select the "New I-Servo" application axis and enter an axis name.

- To save the events from the servo controller to the hard drive, right mouse click on "Events", then click on "Copy Events To" the new axis you created.

	NOTE
	*It is recommended that the **Program Library, Set-up Parameters** and **Events** be backed up on disk in the event that the drive is not accessible.*

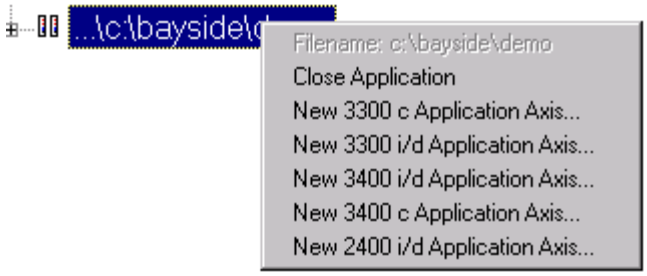
9.4 Saving Program Library to Disk

- Connect communication cable, power up controller and start **Tec Tools**

- Single click on plus sign (+) next to the axis to expand the view.

- To create a new application folder on your hard drive click on "File", "New Application" then enter the application name. The new application folder will be created under the **Tec Tools** ™ directory. (c:\program files\Tec Tools\ application name.MDB) This file can then be saved to disk using Windows Explorer.

- Right mouse click on new application you created and select the correct drive type and enter an axis name.



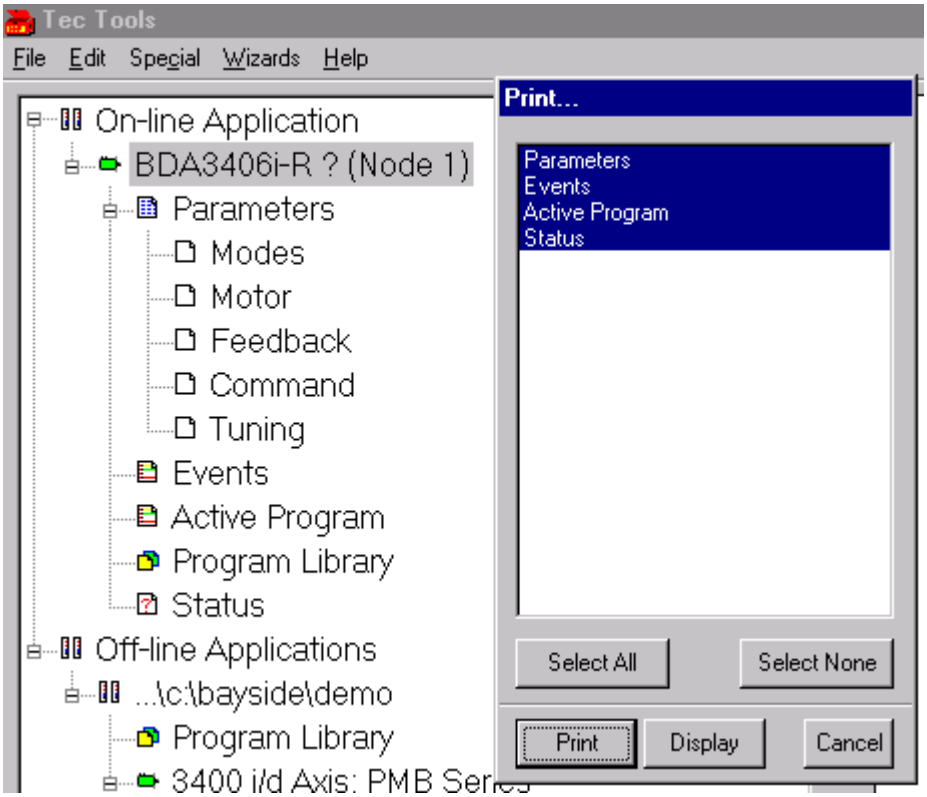
- To save the program library to the hard drive, right mouse click on "Program

- Library", then click on "Copy All Programs To" the new application you created.

	NOTE
	*It is recommended that the **Program Library, Set-up Parameters** and **Events** be backed up on disk in the event that the drive is not accessible.*

9.5 Printing

To print the ***Events***, ***Drive Parameters***, ***Drive Status***, or an existing program begin by highlighting the axis or application. Then on the menu bar click "***File***" then "***Print…***". A window will appear listing all the programs and drive information. Select all or select the individual programs to be printed.



By pressing the "***Display***" button, you can view the file before printing. The display can be used to copy the file to the Clipboard for pasting to another document. (Rich Text Format RFT)

10. Appendices

The information contained in the following appendices is provided for reference purposes.

10.1 Servo Axis Tuning

Closed-loop controllers must be tuned, a process, which several "gains" are adjusted to achieve optimal dynamic performance. The Digital Servo drives use a closed loop system consisting of three components for velocity control with forth variable (PPG) added when in position mode. The controllers' job is to compare the command and the feedback signals and after some processing, send the control current to the motor. The command and feedback processing is done in a Velocity loop which combines the proportional (K_P), integral (K_I) and feedforward (K_F) velocity signals.

The Velocity loop works off an error signal between the commanded velocity and the feedback velocity. Each gain in the loop has a different function.

The proportional gain (K_P) in correct measure provides stability as well as "growing room" for the other gains. If K_P is set too high the motor will become noisy and the system can become unstable.

The effect of the integral gain (K_I) is to make the system stiffer. In other words a large K_I makes it difficult for a torque disturbance to move a shaft. Be careful though, if K_I becomes too large it will cause over-shoot.

The feedforward gain (K_F) controls the response to instantaneous velocity changes. K_F is multiplied by K_P and added back into the velocity control loop. The feedforward gain is important in systems which, require step velocity changes like labeling applications

The position proportion gain (PPG) is used in "Position Mode" and controls the response to the position error then is factored into the velocity loop through the commanded velocity.

10.1.1 Tuning Wizard

To use the Axis Tuning Wizard:

- Highlight the axis,
- Click "Wizard" on the toolbar.
- Click "***Tuning…***". The tuning window will appear.

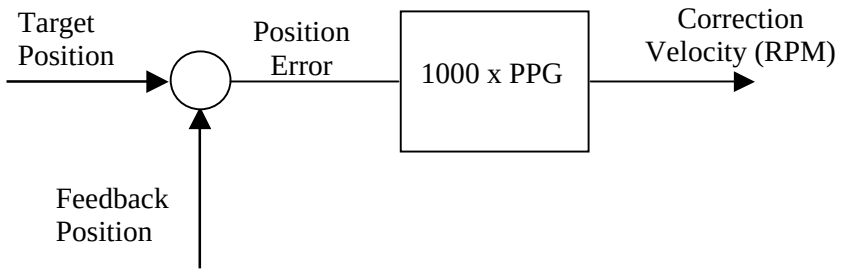
Command Modes

Description	34xxd Servo Drive	34xxi Servo Drive
Current Analog	X	X
Current Digital	X	X
Velocity Analog	X	X
Velocity Digital	X	X
Position		X
Step/Direction	X	X
Step Up/Down	X	X
Quad Encoder	X	X
Open Loop	X	X

Position Loop Gain

The "Position Loop Gain" is only used when in Position, Step/Dir., Step/Step or Quad modes. This variable effects the speed at which the drive reacts to a position error. If set too high the system will become unstable.

Position Loop Schematic:



Correction Velocity (RPM) = PPG (1/Min) x Position Error x 1000

Position Loop Gain (PPG)	Position Error (Rev)	Correction Velocity (RPM)
1	0.1	100
3	0.1	300
4	0.1	400
8	0.1	800
10	0.1	1000

Damping

The Damping slide bar controls the system damping, which effects the value of K_P. The amount of dampening required is determined by the system itself. If the system has a lot of friction less damping is required. An under damped system will cause oscillation, while over dampening will slow system response.

Feedforward Gain

The Feedforward Gain (K_F) slide bar adjusts the accel/decel response to step changes in velocity. The K_F value is multiplied by the proportional gain and factored back into the velocity loop. If set too high the motor will overshoot and if set too low can limit response to step changes in velocity.

Response

The Response slide bar uses an advance control algorithm which is a second order equation. The Response effects both the integral (K_I) and proportional (K_P) gains and controls the stiffness of the velocity loop and partially it's dynamic response. The higher the value means increased stiffness. Too high of a value could cause instability or oscillation. (Velocity Integral Gain)

External Inertia

The External Inertia slide bar allows the user to adjust the bar based on the Load:Motor inertia mismatch. Adjusting this bar changes the maximum/minimum range of the "Response" slide bar.Finding the optimum gain settings can be a tedious process because each variable influences the effect of the others. Optimum tuning vary from system to system. What works good for one mechanical system may cause instability in another. A labeling application requires a quick response to velocity changes, which is not desired on conveyor system, which causes jerking.

10.1.2 Tuning in Current Mode

The master controller does all system tuning when in current mode. The PID loop is done within the master controller, which then send the current command to the drive.

10.1.3 Tuning in Velocity Mode

The best way to tune your system is on the machine with actual inertia and loads. If this is not possible the drive can be setup in "Digital" Velocity mode and a short program can be created to help tune the motor/drive system.

- Command Mode (CM): **Velocity**
- Command Source (DCM): **Digital**
- Acceleration Rate (ACC): **(0.1 < ACC < 100,000 RPM/sec)**
- Deceleration Rate (DEC): **(0.1 < ACC < 100,000 RPM/sec)**

Below is an example of a short program that can be used to tune a system in velocity mode.

Example:

```
LOOP:
    Trigger Capture
    DCV=2000 (or machine maximum velocity)
    Wait 1000 msec.
    DCV=-2000 (or machine maximum velocity)
    Wait 1000 msec.
    Go To LOOP
    End Program (0)
```

The following steps can be followed to tune the system in Velocity mode.

- Start the Oscilloscope and select "***Current Reference***" and "***Feedback Velocity***" vs. time using 3.0 second intervals. Set the monitor for "Trigger", "Repeat" and "Capture".

- Start the "Tuning Wizard" then position the scope and the Tuning Wizard windows so they are both visible on the screen.

- Turn off the "Position Proportional Gain" by unchecking the box in the Tuning Wizard

- Using the default settings begin to adjust the Response slide bar to the right. Click on <Change> to have the new values take effect. (If the Response bar reaches it's limit, adjust the "External Inertia" bar to change the Response max/min limit)

- Monitor the velocity on the scope until it overshoots by about 15% then begin adjusting the "Damping" slide bar to remove the overshoot. (Remember to click on <Change for the new value to take effect)

- This process can be repeated if required. If the system becomes unstable, back off the "Response". Clicking on <Default> then <Change> will revert back to the default values.

	NOTE
	*In most applications it is not necessary to adjust the Feedforward gain. It will neither make nor break the tuning process although can be adjusted to enhance the overall system performance.*

10.1.4 Tuning in Position Mode

The best way to tune your system is on the machine with actual inertia and loads with the program running. If this is not possible, a short Tuning program can be created to tune the motor/drive system.

- Command Mode (CM): **Position**
- Position Error (PET): **(0 – 100,000 revs)**

Below is an example of a short program that can be used to tune a system in position mode.

Example:

```
        // Start low and increase Acceleration as your application requires
        Set Acceleration Rate to 8000 system-rev/sec^2
LOOP:
        Trigger Capture
        Index of 2.0 system-rev At 10 system-rev/sec
        Wait for in Position
        Wait 300 msec.
        Index of -2.0 system-rev At 10 system-rev/sec
        Wait for in Position
        Wait 300 msec.
        Go To LOOP
        End Program (0)
```

This program will trigger capture the scope, Index a 2 rev, wait 1 sec and then start again.

The following steps can be followed to tune the system in Position, Step/Dir., Step/Step or Quad modes. (Remember the "Position Proportional Gain" (PPG) is now used.

- Start the Oscilloscope and select "***Current Reference***" and "***Feedback Velocity***" vs. time using 0.3 second intervals. Set the monitor for "Trigger", "Repeat" and "Capture".

- Start the "Tuning Wizard" then position the scope and the Tuning Wizard windows so they are both visible on the screen.

- Make sure the "Position Proportional Gain" by is turned on by checking the box in the Tuning Wizard

- Using the default settings begin to adjust the Response slide bar to the right. Click on <Change> button to have the new values take effect. (If the Response bar reaches it's limit, adjust the "External Inertia" bar to change the Response max/min limit)

- Monitor the velocity on the scope until it overshoots by about 15% then begin adjusting the "Damping" slide bar to remove the overshoot. (Remember to view the effect of your change on the Oscilloscope screen after clicking <Change>)

- This process can be repeated if required. If the system becomes unstable, back off the "Response". Clicking on <Default> then <Change> will revert back to the default values.

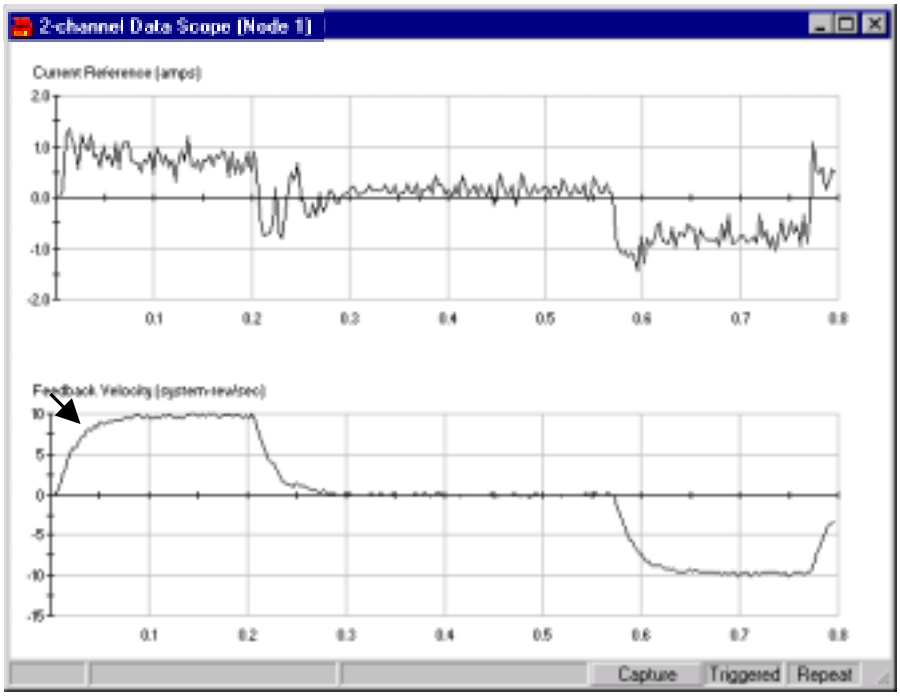
- You can adjust the "Position Proportional Gain" (PPG) to decrease settling time at the end of a move. You will then have to readjust the Response and Damping.

	NOTE
	*In most applications it is not necessary to adjust the Feedforward gain. It will neither make nor break the tuning process although can be adjusted to enhance the overall system performance.*

Position Loop Gain (PPG)

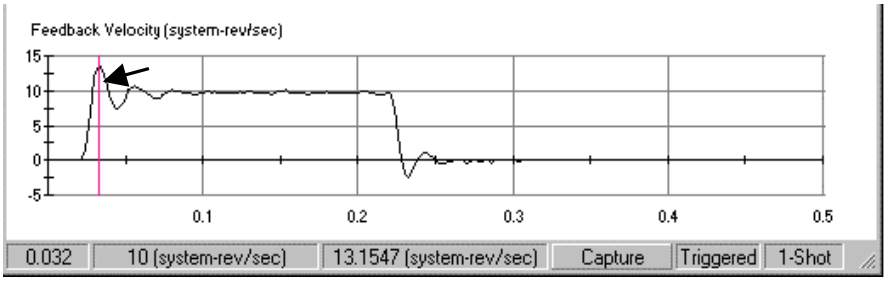
This parameter controls the gain of the position loop. Higher values reduce following errors. Too high a value can cause instability. (Position Proportional Gain)

Position Loop Gain Too Low



A Position Loop Gain too low will result in the drive having a large following error. The motor will lag behind the target position. As shown above the motor never reaches the maximum velocity of 10 system-rev/sec.

Position Loop Gain too High

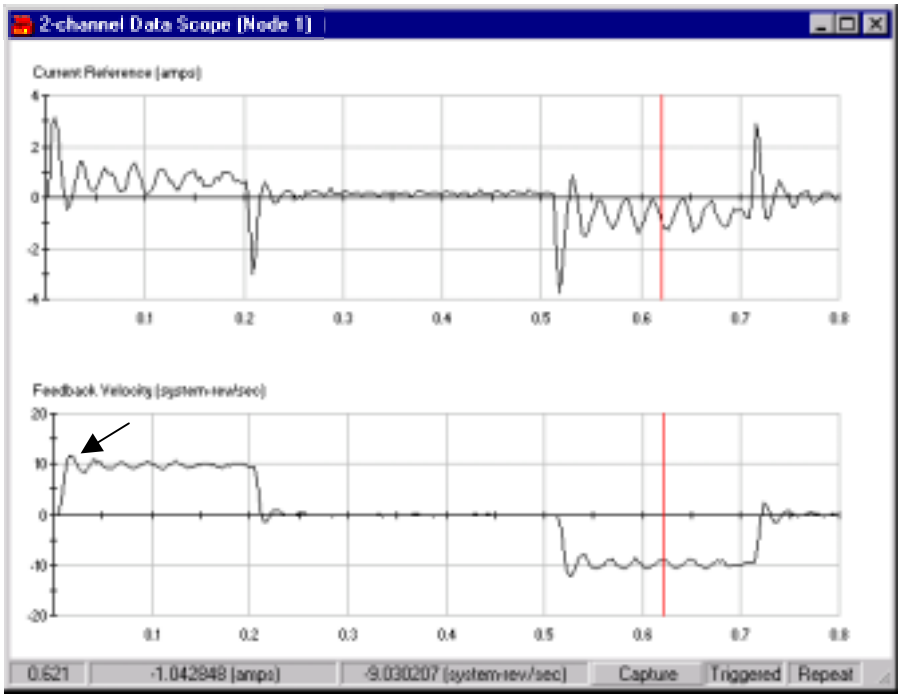


A Position Loop Gain too high will result in the drive trying to maintain a very small following error. The motor will not react well to velocity changes causing it to over react and over shoot. As shown above the motor over shoots the target velocity of 10 system-rev/sec. (May also require Damping)

Damping

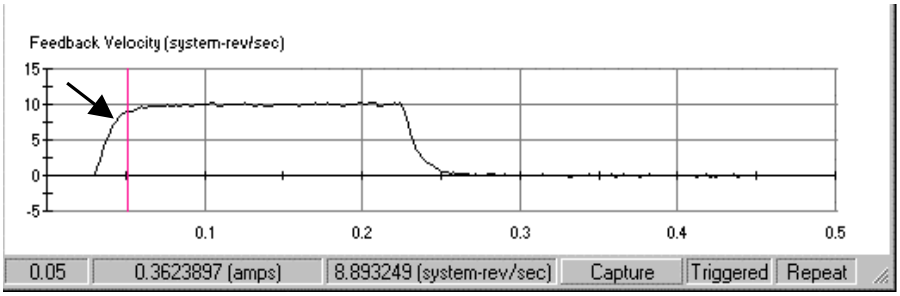
This slide bar controls the dynamic response of the velocity loop. It is present in both the velocity and position command modes. (Velocity Proportional Gain)

Under Damped



When the motor is under damped it will oscillate around the target velocity. Notice the large current oscillations as the axis attempts to control the motion. These current oscillations may be damaging to the motor and amplifier As shown above the motor feedback velocity never flattens out at 10 system-rev/sec.

Over Damped

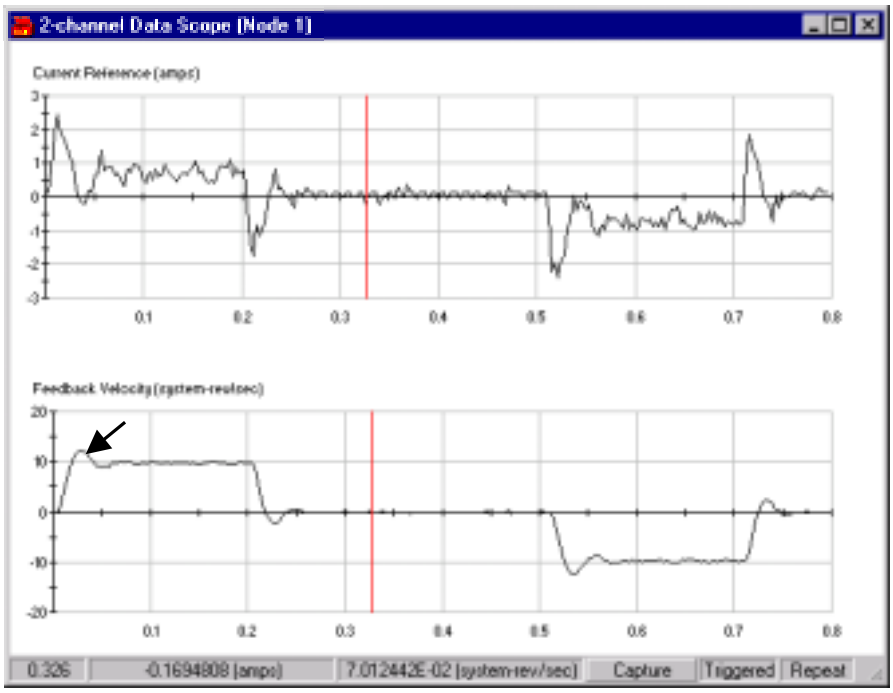


When a motor is over damped it will stop short of the target velocity. The plot above shows a hesitation in the motor acceleration. (May also require Feed Forward)

Feedforward Gain (K_F)

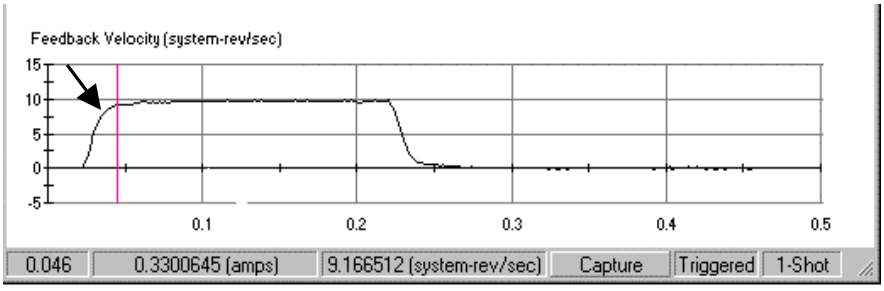
This slide bar controls the dynamic response of the velocity loop, especially during step changes in the command velocity. (Velocity Forward Gain)

Forward Gain too Low



If the Forward Gain is set too low the motor will over shoot.

Forward Gain too High

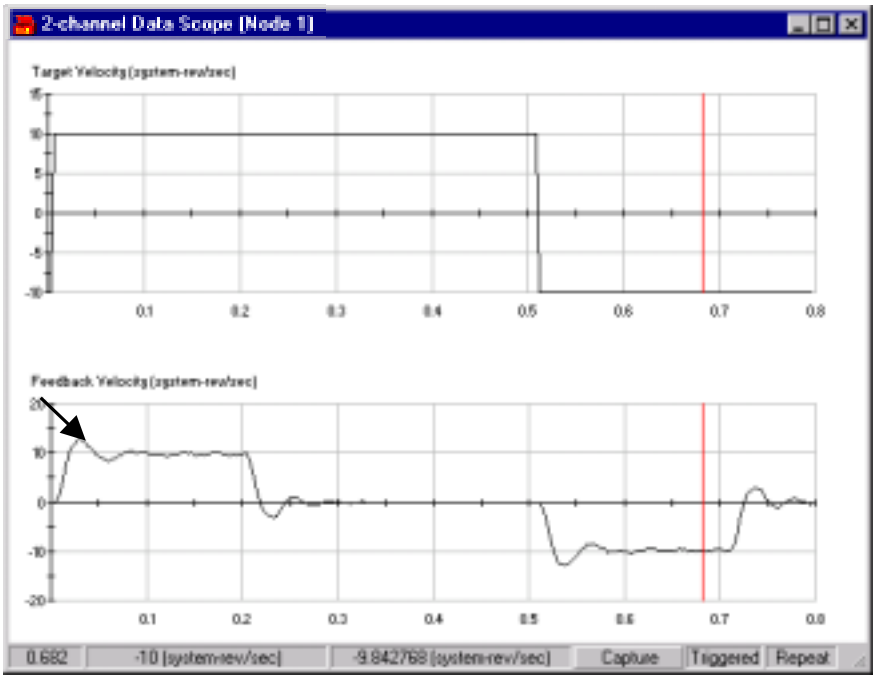


If the Forward Gain is set too high the motor will stop short before hitting the target velocity. (May also require Integral Gain)

Response (K_I)

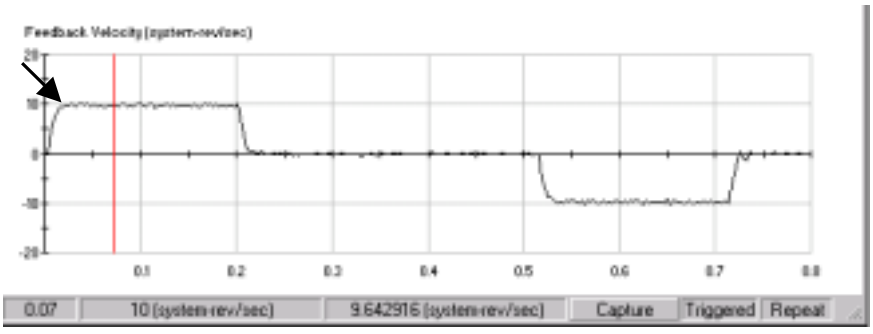
This slide bar controls the stiffness of the velocity loop and partially it's dynamic response. The higher the value means increased stiffness. Too high of a value could cause instability or oscillation. (Velocity Integral Gain)

Response too Low



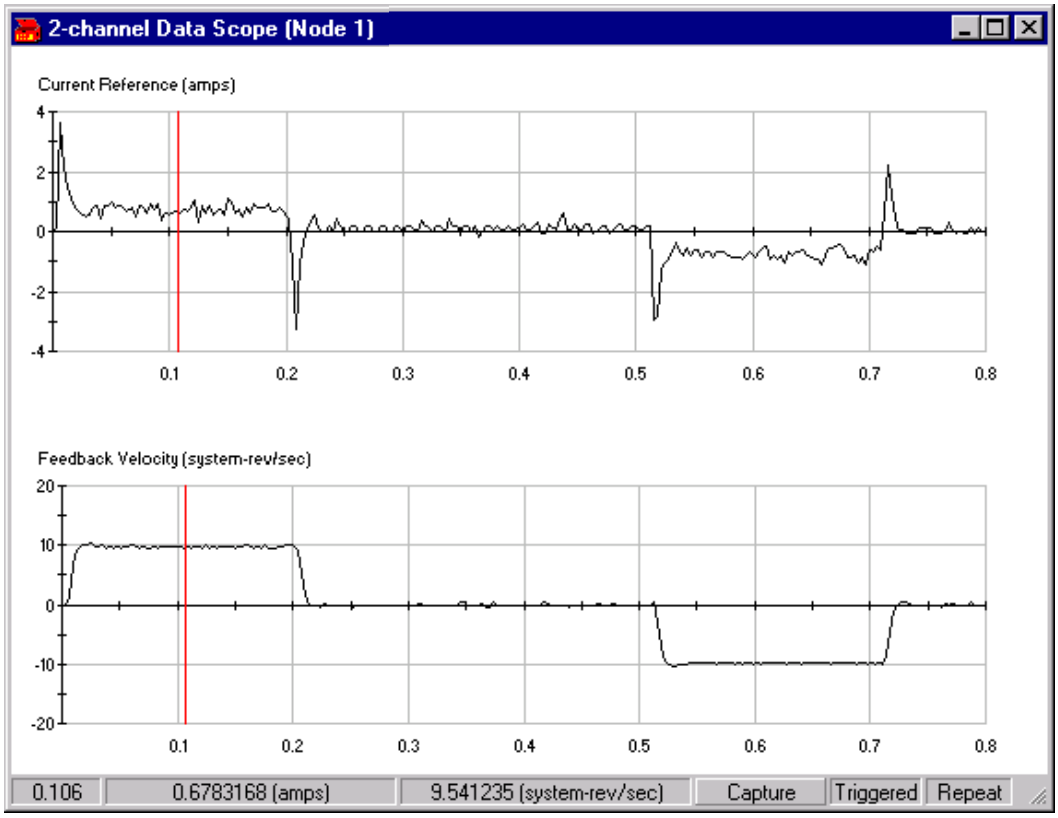
When the Response is too low the motor velocity lags behind the target velocity. The time to get to speed will be longer and the motor tends to overshoot the target velocity. (System does not react well to velocity changes.)

Response too High



When the response is too high the drive tends to over react to velocity changes. In the above graph when the target velocity reached speed and the drive over reacted causing the motor velocity to come up short.

Properly Tuned System



The above plot shows a properly tuned system. The motor accelerates to the target velocity without stopping short or over shooting. The velocity remains constant at 10 system-rev/sec until the motor begins its deceleration. The current peaks during acceleration then drops off and remains low during the constant velocity portion of the move.

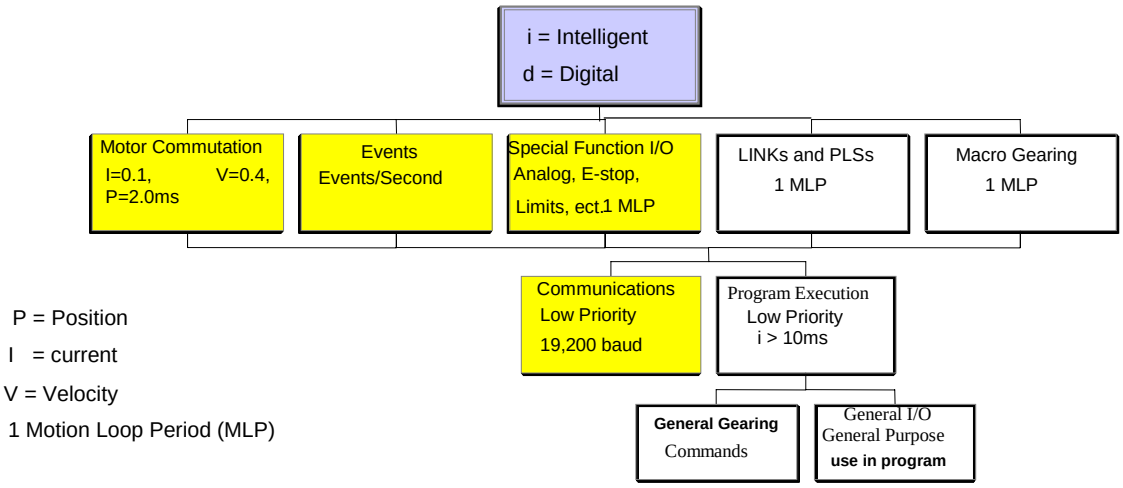
	NOTE
	*A properly tuned system is one that provides the required motion and is stable under all conditions.*

	WARNING
	*Tuning a system for its best possible response may result in exceeding the mechanical systems capabilities.*

10.2 Programming Technique

The goal in writing any program is to meet the control requirements of the machine using the most efficient programming method. The speed of execution is greatly dependent upon the programming technique. There are often many ways to write the same program but only one way which, provides the fastest program execution.

All 24xx and 34xx digital drives are multi-tasking allowing the processor to monitor, calculate and execute multiple functions at the same time. The highest priority is always given to controlling the current, velocity and position loops. The lowest priority is given to general program execution and RS-232/485 communication. See below:



NOTE: For additional programming examples log on to our website at: inside@baysidemotion.com

Bad Programming Technique

The example below shows bad programming technique. When jumping to a label the program begins scanning at the top of the program. To speed up execution, the main program should be located a close to the top as possible. Since the Homing sequence is only executed at power-up it should be located at the bottom of the program.

```
Translation Ratio: 1 motor revolution per system-rev

000:       Set Acceleration Rate to 5000 system-rev/sec²
001:       Move At -6 system-rev/sec Until Input 7
002:       End Move At; Move By  0 system-rev
003:       Wait for In Position
004:       Move to Feedback Null (Positive)
005:       Wait for In Position
006:       Zero the Following Error
007:       Wait 500 msec.
008:  TOP:
009:       If IN&3=1 Then Go To GEAR1
010:       If IN&3=2 Then Go To GEAR2
011:       If IN&3=3 Then Go To GEAR3
012:       Go To TOP
013:  GEAR1:
014:       Gear At 50000/100000
015:       If Input 6 is OFF, Go To GEAR1
016:       Gear At 0/100000
017:       Go To TOP
018:  GEAR2:
019:       Gear At 100000/100000
020:       If Input 6 is OFF, Go To GEAR2
021:       Gear At 0/100000
022:       Go To TOP
023:  GEAR3:
024:       Gear At 200000/100000
025:       If Input 6 is OFF, Go To GEAR3
026:       Gear At 0/100000
027:       Go To TOP
028:       End Program (0)
```

Good Programming Technique

The same program has been rewritten to speed up execution. The main program was moved to the top for faster execution. The subroutines were written as macros (precompiled program steps), which executes each macro step within one position loop. Program execution will speed up by a minimum of 20 times.

```
Translation Ratio: 1 motor revolution per system-rev

000:       Call HOME
001:  TOP:
002:       Macro Start (IN&3) Repeat 1
003:       Wait Macro Complete
004:       Go To TOP
005:  HOME:
006:       Set Acceleration Rate to 5000 system-rev/sec²
007:       Move At -6 system-rev/sec Until Input 7
008:       End Move At; Move By  0 system-rev
009:       Wait for In Position
010:       Move to Feedback Null (Positive)
011:       Wait for In Position
012:       Zero the Following Error
013:       Macro  0
014:       Macro  1
015:       ··· Gear At 50000/100000 in 400
016:       ··· Wait Input Rise 6
017:       ··· Gear At 0/100000 in 400
018:       Macro  2
019:       ··· Gear At 100000/100000 in 400
020:       ··· Wait Input Rise 6
021:       ··· Gear At 0/100000 in 400
022:       Macro  3
023:       ··· Gear At 200000/100000 in 400
024:       ··· Wait Input Rise 6
025:       ··· Gear At 0/100000 in 400
026:       Macro End
027:       Return
028:       End Program (0)
```

The program execution can be sped up further by performing the macro call from within another macro. This will cut down on the main program scanning for a label.

10.3 Example Programs

The following sample programs are intended for reference only. Each individual application has its own set of safety requirements that may or may not be included in these programs. These programs are provided as a teaching reference, and should not be used to control actual equipment.

	NOTE
	*As a courtesy, all example programs can be found on the* **Tec Tools** *CD for loading to the drive.*

Example 1 Homing without Limits Switches

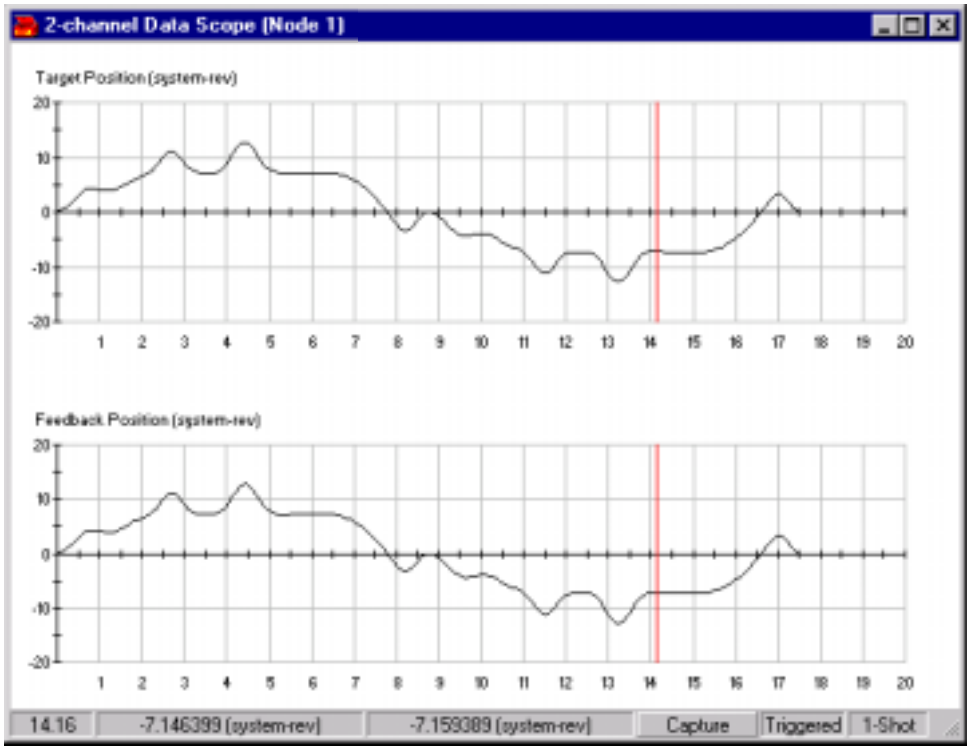
```
Translation Ratio: 1 motor revolution per system-rev
000:       // HOMING ROUTINE WITHOUT LIMIT SWITCHES
001:
002:       Set Acceleration Rate to 4000 system-rev/sec²
003:       Move At 2 system-rev/sec Until Input 7
004:       End Move At; Move By 0.0 system-rev
005:       Wait for In Position
006:       Move to Feedback Null (Negative)
007:       Wait for In Position
008:       Zero the Following Error
009:       End Program (0)
010:
011:       // INPUT 7 IS NORMALLY OPEN HOME SWITCH
012:       // THE SYSTEM WILL MOVE CCW AT 2 RPS UNTIL
013:       // INPUT 7 IS ACTIVE THEN STOP AT THE MARKED
014:       // POSITION BEFORE ROTATING CW TO FEEDBACK NULL
015:       // AND SETTING FPOS AND FOLLOWING ERROR TO ZERO
```

Example 2 Homing with Limits Switches

```
Translation Ratio: 1 motor revolution per system-rev
000:       // HOMING ROUTINE WITH LIMIT SWITCHES
001:       // THIS EXAMPLE IS BASED ON INTELLIGENT DRIVE
002:       // INPUTS 3 IS CCW LIMIT AND 4 IS CW LIMIT
003:       // THE LIMITS ARE TURNED OFF AT THE BEGINNING
004:       // OF THE PROGRAM THEN TURNED BACK ON
005:
006:        IOCW = 0
007:       Set Acceleration Rate to 4000 system-rev/sec²
008:       Move At -5 system-rev/sec While Input 4
009:       End Move At; Move By  .3 system-rev
010:       Wait for In Position
011:       IOCW = 2
012:       Move At 2 system-rev/sec Until Input 7
013:       End Move At; Move By  0 system-rev
014:       Wait for In Position
015:       Move to Feedback Null (Negative)
016:       Wait for In Position
017:       Zero the Following Error
018:       End Program (0)
019:
020:       // INPUT 7 IS NORMALLY OPEN HOME SWITCH
021:       // MOTOR WILL ROTATE CW AT 5 RPS UNTIL THE
022:       // CW LIMIT IS HIT THEN STOP AND ROTATE
023:       // CCW AT 2 RPS UNTIL THE HOME INPUT 7
024:      // IS ACTIVE      THE MOTOR WILL THEN STOP AT THE
025:      // MARKED POSITION BEFORE ROTATING CW TO FEEDBACK
026:       // NULL AND SET FPOS AND FOLLOWING ERROR TO ZERO
```

Example 3 Spline Table (Forward and Reverse)

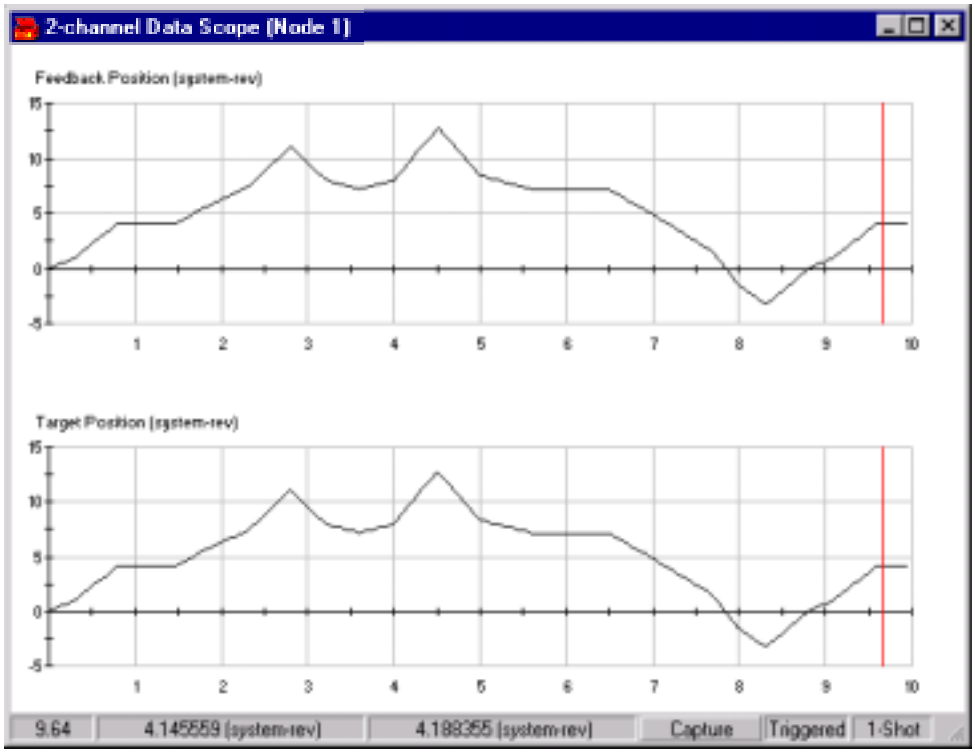
```
Translation Ratio: 1 motor revolution per system-rev
000:    // SPLINE TABLE
002:    Set Feedback Position to 0.0
003:    Clear Spline Table 2
004:    Re-allocate Spline Table 2, 50 entries
005:    Add to Spline Table 2: (0,0,0.3,1,.8,4.2,1.0,4.2,1.5,4.2)
006:    Add to Spline Table 2: (,1.9,6,2.3,7.4,2.8,11.1,3.2,8)
007:    Add to Spline Table 2: (3.6,7.2,4,8,4.5,12.8,5,8.4)
008:    Add to Spline Table 2: (5.6,7.2,6,7.2,6.5,7.2,7.7,1.5,)
009:    Add to Spline Table 2: (8,-1.5,8.3,-3.3,8.8,0) [in-rev]
010:    Precompute Spline Table 2
011:    Trigger Capture
012:    Spline FWD, Table 2, Scale = 1
013:    Spline REV, Table 2, Scale = 1
014:    End Program (0)
```



Example 4 Linear Spline Table

```
Translation Ratio: 1 motor revolution per system-rev

000:     // LINEAR SPLINE TABLE
001:     Set Feedback Position to 0.0
002:     Clear Spline Table 2
003:     Re-allocate Spline Table 2, 50 entries
004:     Add to Spline Table 2: (0,0,0.3,1,.8,4.2,1.0,4.2)
005:     Add to Spline Table 2: (1.5,4.2,1.9,6,2.3,7.4) [in-rev]
006:     Add to Spline Table 2: (2.8,11.1,3.2,8,3.6,7.2) [in-rev]
007:     Add to Spline Table 2: (4,8,4.5,12.8,5,8.4) [in-rev]
008:     Add to Spline Table 2: (5.6,7.2,6,7.2,6.5,7.2) [in-rev]
009:     Add to Spline Table 2: (,7.7,1.5,8,-1.5) [in-rev]
010:     Add to Spline Table 2: (8.3,-3.3,8.8,0) [in-rev]
011:     Loop Spline Table 2 Forever
012:     Set Linear Spline Mode, Table 2
013:     Precompute Spline Table 2
014:     Trigger Capture
015:     Spline FWD, Table 2, Scale = 1
016:     End Program (0)
```

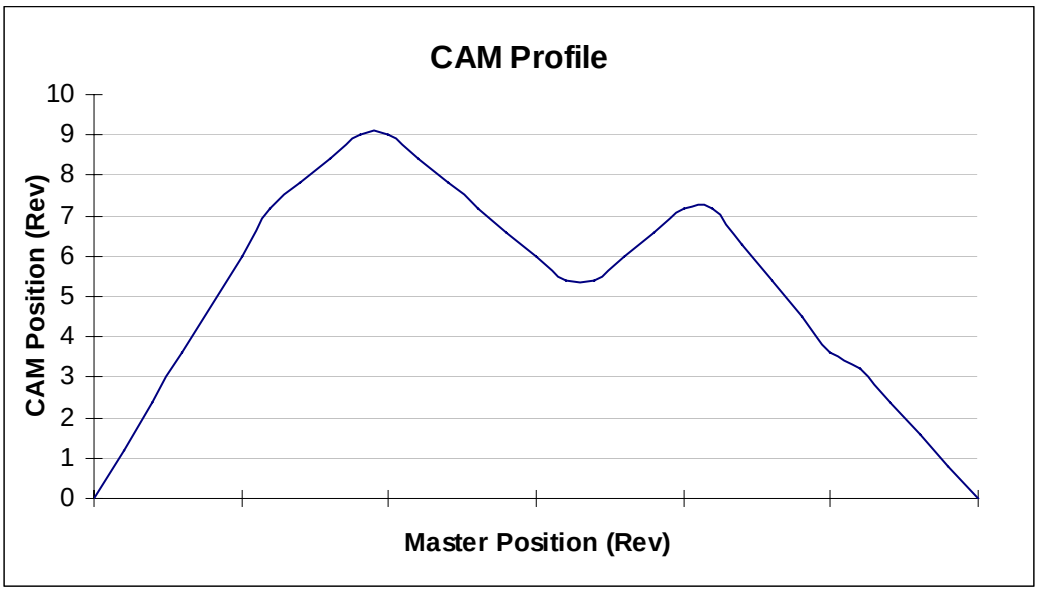


Example 5 CAM Profile (See Profile below)

```
Translation Ratio: 1 motor revolution per system-rev

000:     // CAM FOLLOWING PROGRAM
001:
002:     Zero the Following Error
003:     Re-allocate Spline Table 1, 50 entries
004:     Clear Spline Table 1
005:     Add to Spline Table 1: (0,0,.03,1.2,.07,2.4, .1,3.6)
005:     Add to Spline Table 1: (.13,4.8,.17,6, .2,7.2,.23,7.8)
006:     Add to Spline Table 1: (.27,8.4,.3,9,.33,9,.37,8.4)
007:     Add to Spline Table 1: (.4,7.8,.43,7.2,.47,6.6,.5,6)
008:     Add to Spline Table 1: (.53,5.4,.57,5.4,.6,6,.63,6.6)
008:     Add to Spline Table 1: (.67,7.2,.7,7.2,.73,6.3,.77,5.4)
009:     Add to Spline Table 1: (.8,4.5,.83,3.6,.87,3.2,.9,2.4)
010:     Add to Spline Table 1: (.93,1.6,.97,.8,1,0)
011:     Loop Spline Table 1 Forever
012:     CAM Move FWD, Table 1, Scale of 1
013:     End Program (0)
014:
015:     // SET SPPR EQUAL TO SECONDARY ENCODER RESOLUTION
016:     // DRIVE SHOULD BE IN POSITION MODE
017:     // I DRIVE NEEDS TO HAVE INPUTS 3 AND 4 AS QUAD
018:     // ENCODER INPUTS FOR POSITION MODE
```

Master (Deg)	0	12	24	36	48	60	72	84	96	108	120
Master (Rev)	0.00	0.03	0.07	0.10	0.13	0.17	0.20	0.23	0.27	0.30	0.33
Slave (Rev)	0	1.2	2.4	3.6	4.8	6	7.2	7.8	8.4	9	9
Master (Deg)	132	144	156	168	180	192	204	216	228	240	
Master (Rev)	0.37	0.40	0.43	0.47	0.50	0.53	0.57	0.60	0.63	0.67	
Slave (Rev)	8.40	7.80	7.20	6.60	6.00	5.40	5.40	6.00	6.60	7.20	
Master (Deg)	252	264	276	288	300	312	324	336	348	360	
Master (Rev)	0.70	0.73	0.77	0.80	0.83	0.87	0.90	0.93	0.97	1.00	
Slave (Rev)	7.2	6.3	5.4	4.5	3.6	3.2	2.4	1.6	0.8	0	



The above CAM Profile shows the CAM position of the motor in system-revolutions based on one rotation of the master encoder. If the Master encoder reverses direction the CAM motor will follow the same profile in the reverse direction. If the application can not tolerate the CAM reversing, check the "***Restrict Master (+) [Anti-backup]***" in the I/O Configuration Wizard.

Example 6 Mixing Absolute and Indexed Moves

```
Translation Ratio: 1 motor revolution per inch

000:       // MIXING INDEX AND ABSOLUTE MOVES
001:
002:       Zero the Following Error
003:       Set Acceleration Rate to 5000 inch/sec²
004:  WAIT:
005:       If Input 3 is ON, Go To CHECK
006:       Go To WAIT
007:  CHECK:
008:       If IN&96=0 Then Go To HOME
009:       If IN&96=32 Then Go To TOP
010:       If IN&96=64 Then Go To MOVE UP
011:       If IN&96=96 Then Go To MOVE DOWN
012:  HOME:
013:       Absolute Move To ZERO inch At 20 inch/sec
014:       Wait for In Position
015:       Go To WAIT
016:  TOP:
017:       Absolute Move To 100 inch At 20 inch/sec
018:       Wait for In Position
019:       Go To WAIT
020:  MOVE UP:
021:       Absolute Move To TPOS+5 inch At 8 inch/sec
022:       Wait for In Position
023:       Go To WAIT
024:  MOVE DOWN:
025:       Absolute Move To TPOS-5 inch At 8 inch/sec
026:       Wait for In Position
027:       Go To WAIT
028:       End Program (0)
029:
030:       // IF INPUT 6 AND 7 OFF GOTO HOME
031:       // IF INPUT 6 ON AND 7 OFF GOTO TOP
032:       // IF INPUT 6 OFF AND 7 ON GOTO UP
033:       // IF INPUT 6 AND 7 ON GOTO DOWN
034:
035:       // SINCE INDEX MOVES RESETS FEEDBACK POSITION TO
036:       // ZERO AFTER EACH MOVE TO MIX INDEX AND
037:       // ABSOLUTE MOVE COMMANDS USE ALL ABSOLUTE MOVE
038:       // COMMANDS AND FOR A INDEX MOVE USE TPOS MINUS
039:       // OR PLUS THE INDEXED MOVE DISTANCE
```

Example 7 Link Command and TBAS (Time Base)
Translation Ratio: 1 motor revolution per inch

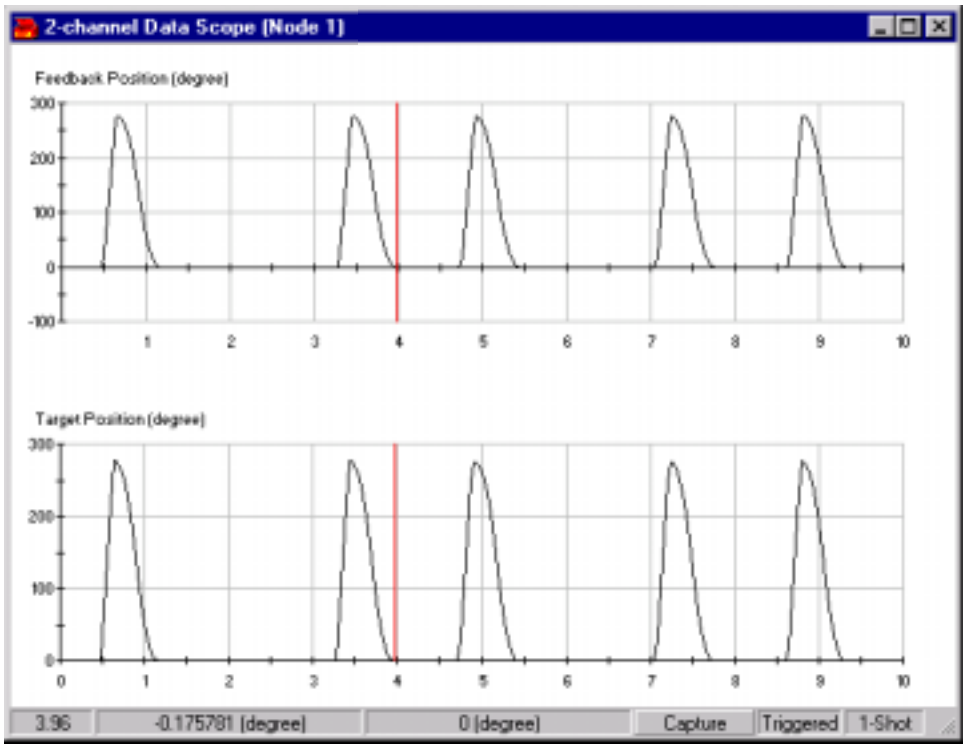
```
000:       // LINK TBAS
001:
002:       Clear Links
003:       TBAS = ONE
004:       PT1 = 0.1
005:       // HOME ROUTINE HERE
006:       Move to Feedback Null (Negative)
007:       Wait for In Position
008:       Set Feedback Position to 0.0
009:  CYCLE:
010:       Link TBAS=ADC1*PT1+ZERO
011:       Set Output(s) 1
012:       // READY TO START MOTION
013:  HOLDING:
014:       If Input 3 is OFF, Go To HOLDING
015:       Clear Output(s) 1
016:       Absolute Move To 100 inch At 10 inch/sec
017:       Wait for In Position
018:       Clear Links
019:       TBAS = ONE
020:       Absolute Move To 0 inch In 6 sec
021:       Wait for In Position
022:       Go To CYCLE
023:       End Program (0)
024:
025:       // ADC1 ACTS AS A SPEED OVERRIDE ON MOTION
026:       // 10 VDC IS 10 INCH/SECOND
027:       // 5 VDC IS 5 INCH/SECOND
028:       // 1 VDC IS 1 INCH/SECOND
031:       // ADC1<= 0 VDC IS 0.0 INCH/SECOND
```

In the above example the Time Base “TBAS” is linked to the Analog Input and is used to modify motor velocity and acceleration during an absolute move. The Link is then cleared and the motor moves back to position 0 in 6 seconds. Each time Input 3 becomes active this process will repeat.

Example 8 Electronic Gearing

```
Translation Ratio: 2.777778E-03 motor revolution per degree
000:       // //ELECTRONIC GEARING
001:
002:       // HOME ROUTINE
003:       Move At 240 degree/sec Until Input 7
004:       End Move At; Move By  0 degree
005:       Wait for In Position
006:       Set Feedback Position to 0.0
007:       Absolute Move To -10 degree In 1 sec
008:       SMC = EPPR
009:       ACCEL = 100
010:       MCOUNT = 8192
011:       SCOUNT = 8192
013:       Macro  1
014:       ··· Wait Input Rise 3
015:       ··· Connect Master Encoder
016:       ··· Gear At SCOUNT/MCOUNT in ACCEL
017:       ··· Wait on SMOD Within 6100/6200
018:       ··· Gear At ZERO/MCOUNT in ACCEL
019:       ··· Disconnect Master Encoder
020:       Macro End
022:       // MAIN ROUTINE
023:  TOP:
024:       If Input 3 is OFF, Go To TOP
025:       Begin Master Position Tracking
026:       Macro Start 1 Repeat 1
027:       Wait Macro Complete
028:       End Master Position Tracking
029:       // WE NEED TO USE 0 AS THE BEGIN-TRACKING
030:       // COMMAND RE-ZEROS THE FEEDBACK POSITION
031:       Absolute Move To 0 degree In .5 sec
032:       Wait for In Position
033:       Trigger Capture
034:       Go To TOP
035:       End Program (0)
036:
037:       // IT WILL GEAR FOR EXACTLY 270 DEGREES
038:       // THEN REVERSE TO 10 DEGREES AND WAIT
039:       // ON INPUT 1
040:       // THIS IS ASSUMING EPPR=8192
```

In this example, the system homes to Input 7 before rotating -10° and stopping. Once Input 3 becomes active the electronic gearing is connected and the motor will follow an external encoder signal through 270° of rotation before disconnecting the electronic gearing and returning to position 0°. The process is repeats each time input 3 is activated. The time between moves is dependent on input 3.



As the Oscilloscope shows the move profiles are similar when the master encoder runs at a constant speed. The dwell between moves is depends upon the time between Input 3 becoming active.

10.4 Table of Command Modes

Command Modes

Description	34xxd Servo Drive	34xxi Servo Drive
Current Analog	X	X
Current Digital	X	X
Velocity Analog	X	X
Velocity Digital	X	X
Position		X
Step/Direction	X	X
Step Up/Down	X	X
Quad Encoder	X	X
Open Loop	X	X

10.5 Table of Drive Variables

Analog Input Variables

Table 1

Code	Description	Unit	Lower Limit	Upper Limit	Read Only
ADC1	Analog input 1	volts	-10	10	X
ADC2	Analog input 2	volts	-10	10	X
CAL1	Offset for analog input 1	volts	-10	10	
CAL2	Offset for analog input 2	volts	-10	10	

Position, Velocity and Torque Varaibles

Table 2

Code	Description	Unit	Lower Limit	Upper Limit	Read Only
ACC	Acceleration Rate	RPM/sec	0.001	100,000	
CPOS	CAM Position	Revolutions	0	1	X
DCV	Digital Command Velocity	RPM	-15000	15000	
DCC	Digital Current Command	Amps	Model dependent		
DEC	Deceleration Rate, analog velocity mode only	RPM/sec	0.001	100,000	
FPOS	Feedback Position	Revolutions	-2^{31}	2^{31}	
JOGS	Jog Speed	RPM	Drive Dependent		
MARK	Marked Position	Revolutions	-2^{31}	2^{31}	X
PERR	Position Error	Revolutions			X
PET	Position Error Tolerance	Revolutions	0	32565	
PFLD	Percent Foldback Available	Percent	0	100	X
POS	Position in Feedback counts	Counts	-131,072	131,072	X
POSM	Motor Position	Revolutions			X
POSS	Position in of Step/Dir, Step Up/Down or Quad.	Steps			X
TPOS	Target Position	Revolutions	-2^{31}	2^{31}	X
UU	User Units (per Motor Rev)		0	6	
VEL	Feedback Velocity	RPM	Feedback/Motor Dependent		X

Configuration Variables

Table 3

Code	Description	Unit	Lower Limit	Upper Limit	Read Only
BUSU	Bus Undervoltage Set Point	VDC	0	400	
CDLY	Commutation Delay		None	None	X
CM	Command Mode; Velocity=0, Current=1, Position=2, Step & Dir=3, Step Up/Down=4, Quad=5 and Open Loop=6		0	6	
COFF	Commutation Offset	Degrees	-180	180	
COT	Check Motor OT Input		0	1	
CTMP	Drive Temperature	Degrees C	-------	-------	X
DCM	Command Source	Analog/Digital	0	1	
DM	Drive Mode; Variable Frequency=0, Brushless=1, Induction=2, Brush DC=3, Brush 6-step (trap)=4, Brushless (ingnor Halls)=5		0	5	
EM	Enable Source; Opto Input=0, Opto Input and-ed with SWE=1,Opto Input or-ed with SWE=2 and SWE=3		0	3	
EPPR	Feedback Encoder	pulses/rev	1	324768	
FBF	Feedback Filter (velocity)	Hz	0	10,000	
FEED	Motor Feedback device type		Model dependent		
FINV	Invert Feedback Direction		0	1	
FLT	Fault Code		-------	-------	X
FSV	Velocity @ 10V Analog Cmd	RPM	0.01	100000	
HED	Displays the current status of the HED inputs		1	6	X
HINV	Invert Hall (HED) Direction		0	1	
HSIF	High Speed Inpiut Filter	Hz	30 kHz	2 MHz	
IKID	Velocity Loop D-axis gain		0	None	
IKIQ	Velocity Loop D-axis gain		0	None	
IKPD	Current Loop D-axis gain		0	None	
IKPQ	Current Loop Q-axis gain		0	None	
IMAX	Maximum Allowed Current	Amps	Model dependent		
INER	Motor Inertia	kg-m²	0.000001	None	
IRA	Current ref in amps	Amps	Model dependent		
IRMS	Foldback RMS Current Limit	Amps	0.01	Drive Dependent	
KF	Velocity feedforward gain		0	10000	
KI	Velocity Integeral gain	Nm/RPM/sec	0	10000	
KP	Velocity proportional gain	Nm/RPM	0	10000	

Configuration Variables Continued

Table 3

Code	Description	Unit	Lower Limit	Upper Limit	Read Only
KT	Motor Torque Constant	Nm/Amp	0	10000	
MBR	Macro being run		0	64	
MPOL	Motor Pole Count	pole	3	64	
ONE	System variable == 1		One	One	X
OSPD	Overspeed Fault Setpoint	RPM	1	100000	
PPG	Position Proportional Gain	1000/min	0	100	
RERR	End of program Error Code		1	99	X
RGNI	Regen Integration Increment		None	None	X
RGNL	Limit on Integrated Regen		None	None	X
RPOL	Feedback Resolver	pole	2	16	
SPPR	Step Pulse Count used to set electronic gear ratio	pulses/rev	1	65536	
STEP	The program Step number		0	256	X
SWE	Software Enable; 0=Disabled, 1=Enabled		0	1	
SWF	Stop Pgm on Fault		0	1	
TBAS	System variable	Seconds	0	2028	
THET	Angular motor Position	Counts	0	EPPR	X
TR	Translation Ratio, ratio of distance to motor rev.		-16384	16384	
VDC	Bus voltage present (DC)	VDC	Model dependent		X
VFF	VF Mode Frequency	Hz	-1200	1200	X
VFI	VF Mode Current Command	Amps	Model dependent		X
ZCHN	Z Channel Pulse		-1	0	X
ZERO	System variable == 0		Zero	Zero	X

Factory Variables

Table 4

Code	Description	Units	Lower Limit	Upper Limit	Read Only
IFBK	Composite output current to the motor, I servo only.		0	Drive Dependent	X
RSOS	Resolver sine offset, factory		non zero	5	X
RCOS	Resolver Cos offset		non zero	5	X
ICO	Current loop offset		non zero	500	
IBO	Current loop offset		non zero	500	

I/O Variables

Table 5

Code	Description	Unit	Lower Limit	Upper Limit	Read Only
HSIF	High Speed Input Filter	MHz	2	250	
IN	Bit representation of inputs that are active		0	Drive depend	X
IN0	Force input bits "#" off		0	Drive depend	
IN1	Force input bits "#" on		0	Drive depend	
INX	Bit representation of Extended Inputs that are active		0	32,768	X
INX0	Force Extended Input bits "#" off		0	32,768	
INX1	Force Extended Input bits "#" on		0	32,768	
IOCW	I/O Configuration Word		0	Drive depend	
OFF	Function call, represents input bits that are inactive		0	Drive depend	X
ON	Function call, represents input bits that are active		0	Drive depend	X
ON(#)	Function call, represents input bits that are active		1	Drive depend	X
OUT	Bit representation of outputs		0	Drive depend	X
OUT0	Force bits output bit "#" off		1	Drive depend	
OUT1	Force bits output bit "#" on		1	Drive depend	
OUTX	Bit representation of outputs		0	Drive depend	X
OUTX0	Force bits output bit "#" off		0	32,768	
OUTX1	Force bits output bit "#" on		0	32,768	
OUTON	Check Status of Outputs On (-1) , Off (0)		1	Drive depend	X

Electronic Gearing Variables

Table 6

Code	Description	Unit	Lower Limit	Upper Limit	Read Only
MAST	Master Encoder Counts	Counts	0	MMC	X
MDEL	Master Displacement	Counts	0	MMC	X
MMC	Master Modulo Constant	Counts	2	1,000,000	
MPAI	Master pulses received during the last "Gear At In"	Counts	N/A	N/A	X
MPFI	Master pulses received during	Counts	N/A	N/A	X
MSP	Macro Step Pointer	Step Number	0	15	X
MVMS	Master Pulse per millisecond	Counts/ms	0	1000	X
PLSM	Floating point version of MAST	Counts	0	1,000,000	X
PLSS	Floating point version of SLAV	Counts	0	None	X
PLSX	Floating point version of SMOD	Counts	0	None	X
SLAV	Slave Position	Counts	0	SMC	X
SMC	Slave Modulo Constant	Counts	0	None	
SMOD	Slave Modulo	Counts	0	SMC	X
SMRK	Slave Mark variable	Counts	0	None	X
SPAI	Slave pulses commanded during the last "Gear At In"	Counts	N/A	N/A	X
SPFI	Slave pulses generated during the last "Gear For In"	Counts	N/A	N/A	X
SPHZ	Slave Phase Adjust	Counts	0	SMC	X
TMC	Master pulses received since variable last cleared	Counts	-2^{40}	2^{40}	
TSC	Slave pulses generated since the variable was last cleared	Counts	-2^{40}	2^{40}	
WLIM	Wait for input limit in slave counts	Counts	0	None	

Index

Bayside Motion Group
27 Seaview Blvd.
Port Washington, NY, 11050
Phone: 516 484 5353
Fax: 516 484 5496
Email is : inside@baysidemotion.com