

# SilverLode™ Servo Family User Manual

Revision 6.03

30 March 2013

For QuickControl Rev 6.03

# Table of Contents

- How to Use This Manual ..... 6
  - Notations ..... 6
- Warnings..... 6
  - What's New For QuickControl 6.03? ..... 8
  - Application Related Information ..... 8
  - Trademarks ..... 8
  - Copyright..... 8
- Chapter 1: Quickstart Motion Guide ..... 9
  - Connecting the Hardware ..... 9
  - USB Serial Ports and drivers ..... 9
  - Hardware Requirements..... 9
  - Install QuickControl ..... 9
- 10 Minutes To Motion ..... 10
  - Run QuickControl 1st Time And Setup Communications..... 10
  - Initialization Wizard ..... 11
  - Control Panel..... 11
  - Create and Run First Program..... 12
- Chapter 2: Expanding Your Program ..... 16
  - Basic Motion Commands ..... 16
    - Relative Motion ..... 16
    - Absolute Motion..... 16
    - Velocity Based Motion ..... 16
    - Velocity Control ..... 16
    - Time Based Motion..... 17
    - S-Curve Factor ..... 17
  - Register Based Motion Commands ..... 21
    - Data Registers..... 21
    - Register Moves..... 21
    - Extended Register Moves ..... 21
    - Profile Moves ..... 21

Registered Step and Direction (RSD) .....	21
Input Modes .....	21
Program Debugging .....	23
Debug Mode .....	23
Single Step/Break.....	23
Single Step Trace .....	23
Real-Time Trace .....	23
Breakpoints .....	23
Real-Time Breakpoints.....	23
View Thread .....	24
Test Line .....	24
Jump To Line.....	24
Chapter 3: Unique Features and Commands.....	27
Status Words .....	27
Polling Status Word (PSW) .....	27
Internal Status Word (ISW) .....	30
Internal Status Word 2 (IS2) Description .....	32
Extended IO Word (XIO) Description (SD).....	34
Error Limits and Drag Mode .....	35
Error Limits Command Parameters .....	35
Error Limits Operation .....	35
Drag Mode.....	35
Related Profile Move Commands.....	37
Profile Velocity Continuous (PVC) .....	37
ERL - Error Limits and Drag Mode Operation with MAV and PMV.....	38
Anti-Hunt™ Feature .....	39
Using Anti-Hunt™ .....	39
Anti-Hunt™ Commands.....	39
Anti-Hunt Operation .....	41
Multi-Tasking.....	42
Multi-Tasking Operation Rules .....	42
Multi-Tasking Operation .....	42
Multi-tasking Exceptions and Interactions.....	43
Multi-Tasking Examples .....	44

Multi-Thread .....	45
Using QuickControl To Launch Thread 2 .....	46
Specialty Commands .....	47
Calculation (CLC).....	47
The CLC command provides basic math and logic functions. ....	47
Calculation Two Word (CTW) .....	47
Calculation Extended (CLX) and Calculation Extended With Data (CLD) .....	47
WCW and WCL Commands .....	47
Input Mode Commands .....	49
Shutdown and Recovery .....	49
Advanced Topics.....	49
Please refer to the technical documents available on our website for further instruction and information on advanced topics, such as camming, torque control or interpolated motion control. Our Technical Support Staff are also available by phone or email during regular business hours. ...	49
Chapter 4: Input Output Functions .....	50
SMI and SIP interfaces .....	50
I/O Functions .....	50
Analog Inputs.....	52
Using Digital Inputs.....	52
General Digital Inputs .....	52
Kill Motor on Input .....	53
Digital Input Filter (DIF) Command.....	53
Using Digital Outputs .....	53
Using Analog Inputs.....	54
Using Encoder Signals with Digital I/O.....	54
Encoder Signal Types .....	54
Step and Direction Signals.....	54
Step Up/Step Down Signals (SilverNugget Only).....	55
A and B Quadrature Signals.....	55
Dual Loop Control .....	56
Chapter 5: Troubleshooting Guide .....	57
Troubleshooting Communications .....	57
SilverLode Indicator LEDs .....	57
Green LED Flash Code .....	58

Troubleshooting Frequently Asked Questions.....	59
Sample Programs.....	67
Main Init.....	67
Startup Recovery .....	67
Kill Motor Recovery .....	67
Power Low Recovery.....	67
Flash Seq .....	67
Factory Block Fault.....	67
Appendix II Initialization and Set Up .....	68
Initialization File.....	68
Initialization Wizard Details .....	68
Initialization Scenarios.....	70
Scenario 1: I-Grade SilverDust Controller With I-Grade Motor.....	70
Scenario 2: I-Grade SilverDust With Non I-Grade Motor.....	70
Scenario 3: SilverNugget or SilverDust MG .....	71
Scenario 4: SilverDust with Custom Motor or 3rd Party Motor .....	71
Scenario 5: I-Grade SilverDust Controller With I-Grade Motor To Be Attached in the Field .....	72
Download File To Device.....	74
Default Configuration .....	74
Typical Configurations .....	75
Setup Menu Details .....	76
Comm Port .....	76
Register Devices.....	76
Protocol.....	77
E-485 Bridge (Ethernet) .....	78
Phase Alignment.....	78
Appendix III Configuring Your PC .....	80
Auto Power-Down Options .....	80
Appendix IV QuickControl® Overview .....	81
QuickControl® Interface .....	81
Toolbar .....	82
Program Window.....	82
Control Panel.....	83
Device Status Monitor .....	85

Edit Details .....	86
Register Watch .....	86
Register Names.....	88
QuickControl Menu Bar .....	90
Program Line Right Click Pop-Up Menu .....	92
Program Info Toolbar.....	93
Appendix VI Techniques for Stopping Motion .....	95
Software Stop Options .....	95
Hardware Stop: Drive Enable feature .....	96
Using Inputs to Stop Motion .....	96
Standard Stop Conditions - QuickControl .....	96
Standard Stop Conditions – Serial Communications .....	97
Advanced Stop Conditions .....	97
Appendix VII Encoder Outputs .....	98
Internal Encoder Output (SilverNugget Only) .....	98
Internal Encoder Output (SilverDust-IGB Only).....	98
Internal Encoder Output (SilverDust-IG8 Only).....	98
Scaled Internal Encoder Output (Modulo Output)(SilverNugget Only) .....	98
Synchronous Serial Interface (SilverDust IG8 only).....	99

# How to Use This Manual

The SilverLode™ Servo Family User Manual is a technical reference designed to aid users in the operation and programming of the SilverLode™ product family which includes the SilverNugget™, SilverSterling™ and SilverDust™. QuickControl®, QuickSilver's software interface, is used for programming, initializing and testing all SilverLode products. The companion publication, SilverLode™ Servo Family Command Reference, provides details on all commands.

## Notations

### SDnn

This notation means the feature or command being described was introduced in SilverDust rev nn.

### SSnn

This notation means the feature or command being described was introduced in SilverSterling rev nn.

### SNnn

This notation means the feature or command being described was introduced in SilverNugget rev nn.

## Warnings

The QuickSilver Controls, Inc (QCI) SilverLode servos are high performance motion system. As with any motion system, it is capable of producing sufficient mechanical output to cause bodily injury and/or equipment damage if it is improperly operated or if it malfunctions. The user shall not attach a QCI product to any mechanism until its operation is fully understood. Furthermore, the user shall provide sufficient safety means and measures to protect any operator from misuse or malfunction of the motion system. The user assumes all liability for its use.



**Do Not Hot Plug The SilverLode Product!** Connecting or disconnecting hot wires or plugs is defined as Hot Plugging. A hot wire is a wire with voltage in it. When this occurs, the residual current in the power circuitry (motor windings, power supply, voltage clamp, 5 Volt supply, communication power...) attempts to find the path of least resistance to ground (before the proper ground connection is established). In most cases this path is through the communication lines (but is not limited to communication failure). The available protection devices are not rated for high transient power spikes, or repeated spikes. Repeated spikes can weaken communication slowly to the point of failure. In some cases, total communication failure can occur in the first and only instance of Hot Plugging. In applications, this can be overcome by connecting chassis ground to the power supply ground. With this direct ground implemented, the path of least resistance for residual power is through the added chassis ground. In applications where chassis ground is isolated from power ground, take EXTREME care not to Hot Plug. Contact QCI if necessary.



User must remove motor from load before initializing the servo or aligning motor index pulse to prevent potential injury or damage.



If Index Phase Alignment option is used, the user must re-run the Initialization Wizard after replacing either motor, encoder, and/or driver; motor must be removed from load prior to powering up system after changing any of these elements to prevent potential injury or damage. Start the wizard with the power turned off, and turn on power when instructed.



Units shall not be used in life critical applications without the signed authorization of the President of QuickSilver Controls.



User is responsible to provide safety interlocks for any application that may cause injury or damage in either normal or abnormal operation of the unit.



The SilverNugget N3 must be wired with a voltage clamp (i.e. QCI-CLCF-04) between the N3 and the Driver power supply; the SilverNugget N2 and SilverDust D2 MG may require a voltage clamp, according to the application. The voltage clamp must be placed close enough to the SilverLode controller/driver to guarantee that the voltage difference between the module and the clamp at maximum current never exceeds 1.5 Volts. (This includes the drop across both the power and ground conductors.) See Technical Document QCI-TD017 High Current Clamp Module - QCI-CLCF-04, QCI-CLOF-04 for information concerning the voltage clamps.



Do not mechanically back drive the motor of a SilverLode servo without a voltage clamp present. The voltage generated may damage the electronics.



User shall limit current to the SilverNugget N3 and SilverSterling S3 units to no more than 35A, or shall fuse power to the SilverNugget N3 using a slow acting fuse rated at not more than 35A. Other units (D2, N2, S2) should have their input current limited to no more than 8 amps.



These products are intended to be used with the appropriate power supplies, motors, electrical protection components and other equipment to form a complete end product or system. They must be installed by a professional assembler who is familiar with the requirements for safety and electromagnetic compatibility (“EMC”). The products are to be tested in the final application at the system level. The assembler is responsible for ensuring that the end product or system complies with all the relevant laws in the country where it is to be used.

## What's New For QuickControl 6.03?

QuickControl 6.03 has been compiled to work with XP, Vista, and Windows 7 (Trademarks of Microsoft). New firmware commands and registers may be accessed with this latest version.

## Application Related Information

Detailed application programming examples are available with the QuickControl software. They are found in a subfolder of the main QuickControl install folder named "QCI Examples". QuickControl can be downloaded from the QCI website [www.QuickSilverControls.com](http://www.QuickSilverControls.com). The website also contains QCI Application Notes that offer the user details of operation in specific applications.

## Trademarks

® QuickControl® and QCI® are Registered Trademarks of QuickSilver Controls, Inc.

SilverLode™, SilverNugget™, SilverDust™, SilverSterling™, PVIA™, QuickSilver Controls™, and AntiHunt™ are trademarks of QuickSilver Controls, Inc.

## Copyright

The SilverLode servo family's embedded software, electronic circuit board designs, embedded CPLD logic, and this User Manual are Copyright © 1996-2013 by QuickSilver Controls, Inc.

# Chapter 1: Quickstart Motion Guide

**WARNING**: Read Warnings section at the beginning of this manual before connecting any hardware. Failure to do so may result in injury or damage to the product. DO NOT connect the motor to any mechanism until you understand its operation. DO NOT connect to any mechanism which may present a hazard to the user in either normal or abnormal operation. Keep clear of any pinch points in mechanisms.

## Connecting the Hardware

Connect the motor to the controller using the motor interface cable (these typically have black overmolded connectors and a ferrite bead). Connect the serial connections to the controller. DO NOT connect any serial adapters to their USB ports at this time. Make connections to the power supply, but do not yet connect the power supply input power.

## USB Serial Ports and drivers

Make sure to use latest drivers. These are available on our web page for downloading. The default windows drivers, in some instances, are several years old and are known to cause problems. Install the drivers first and then only connect the USB port when prompted by the installation software. Connecting the USB ports before installing the drivers will cause the default drivers to be installed.

Most USB serial ports need to have the power down option turned OFF in the device manager. The USB hubs serving the serial port also need to have the power down option turned OFF in the device manager. Leaving the power down option turned on may allow the computer to turn off the serial port while QuickControl is using the serial port, which will cause faults. \*see Appendix III for details on disabling the auto power save.

## Hardware Requirements

- Personal computer with a Pentium (at least 500Mhz) or higher processor running Windows XP, Vista, Win 7.
- An unshared serial port capable of communicating at 57,600 Baud or better (USB to serial adapters ok).
- Some models include an Ethernet connection as well. Please See Ethernet Bridge section for more details.
- QuickControl Software, which is free to download from <http://www.quicksilvercontrols.com/SP/SW/QuickControl.html> This will need an internet connection to install the .NET libraries from MicroSoft.

## Install QuickControl

Install QuickControl as follows: Run the downloaded QuickControl installation program "Setup.exe". If it asks for administrator privileges, allow them. If the Microsoft run time libraries are not already installed on your system, you will be directed to download them from Microsoft.

Note: on Win 7 and Vista, QuickControl may need to be run as an administrator for proper functioning. Configuring the program for WIN XP compatibility mode is also suggested. This may be done by selecting QuickControl.exe , right clicking and selecting properties. From there, run as Administrator and XP compatibility check boxes may be set.

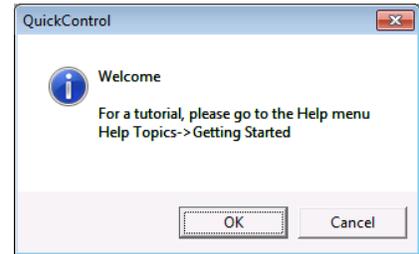
# 10 Minutes To Motion

## Run QuickControl 1st Time And Setup Communications

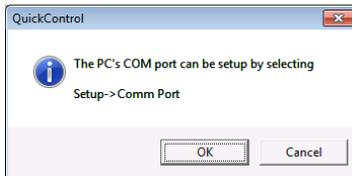
Run QuickControl: From the Start menu,

Start > Programs > QuickControl

If your comm port has not yet been enabled (true for any first time users), QuickControl will prompt you to enable the Comm Port and initialize your servo.

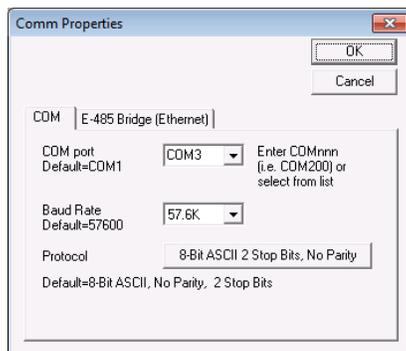


Press OK.



Press OK.

To change the Comm Port at a later time, go to Setup menu, select Comm port, and then press **Modify** to bring up the Comm Properties Window, which can be used to select the correct COM port or to change the Baud Rate.



Press OK when you are finished in either window.



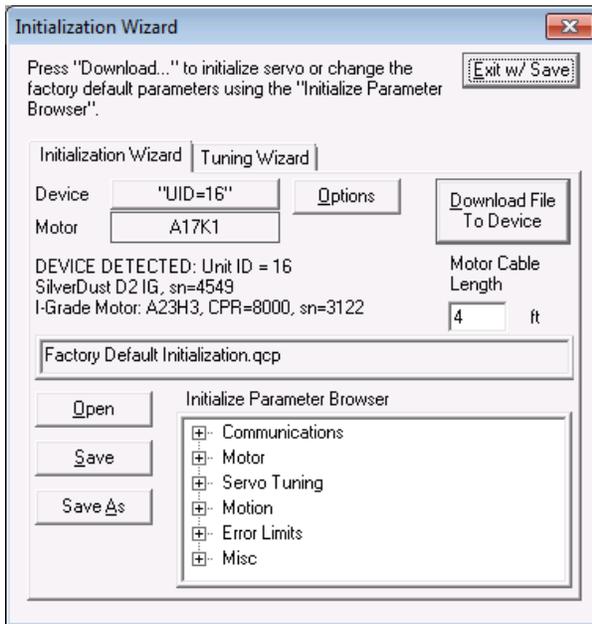
Verify your hardware is setup correctly. Do not connect the motor to a mechanical load until you are familiar with its operation and established any necessary safety interlocks. If your unit requires a clamp or clamp resistor, connect them now. Power up device and press OK.

This will launch the Initialization Wizard.

## Initialization Wizard

This wizard can also be launched from the Tools menu.

Select Open and select the appropriate initialization for your application and hardware. For these examples we are assuming a basic operation using one of our standard QCI servo motors. Different initializations are needed for open loop, and for DC motors and Voice Coils, as well as for CAN operation. See appendix 1 for alternative initialization scenarios.



Enter the Motor Cable Length in the feet. This can be found on the last two digits of the cable part number. For example, a QCI-C-D15P-D15S-04 is a 4' cable. If motor is directly connected to controller (such as a QCI-D2-MG) enter 1 for the length.

Press "Download File To Device" to initialize the servo using factory defaults.

A message will appear when the download is complete. Press OK to acknowledge message.

Press "Save As" to save the file with a name meaningful to you, if you have changed anything in the initialization file, including the ID.

Exit via the X box in the upper right hand corner.

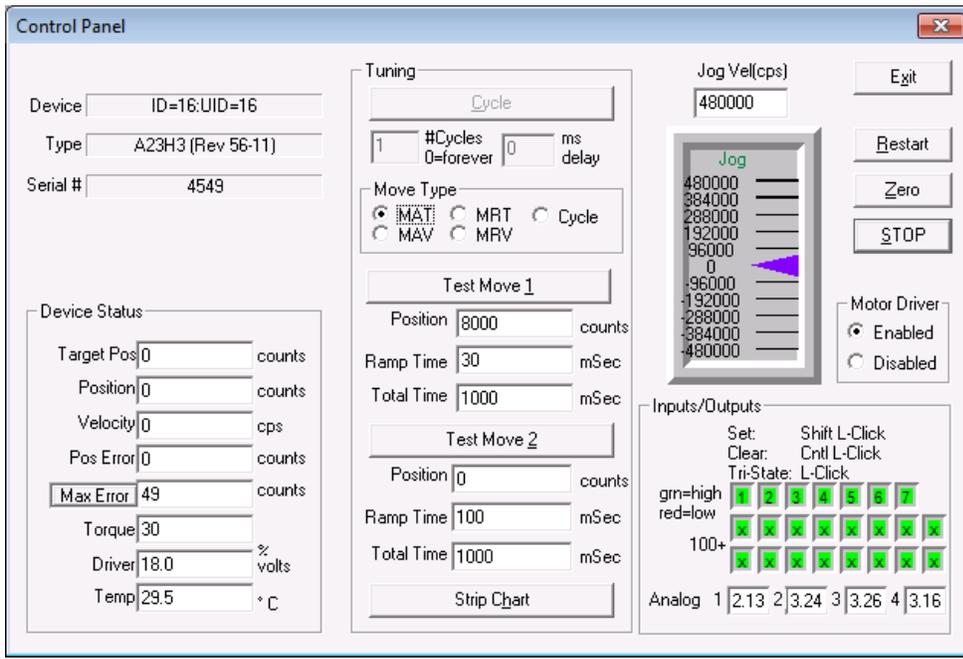
Initialization is complete!

Please note, the Initialization Wizard must be re-run anytime the encoder is loosened, or a different motor/encoder pair is connected (if the Index alignment option is used). If an I-Grade motor is used with an I-Grade SilverDust or SilverSterling controller, the motor can be changed with another of the same type without re-initialization.

Please see Appendix I: Sample Programs and Exercises for a list of possible Initialization scenarios that you may encounter, as well as sample programs included in QuickControl and for additional exercises to help you start programming your device from here.

## Control Panel

Open **Tools=>Control Panel** and a control panel window will open. To verify that the motor and controller are operational, left click and drag the purple slider bar up and down to produce forward and reverse motion. The speed scale box, just above the slider may be changed to allow a faster motion. The move commands may also be used to test particular motions.



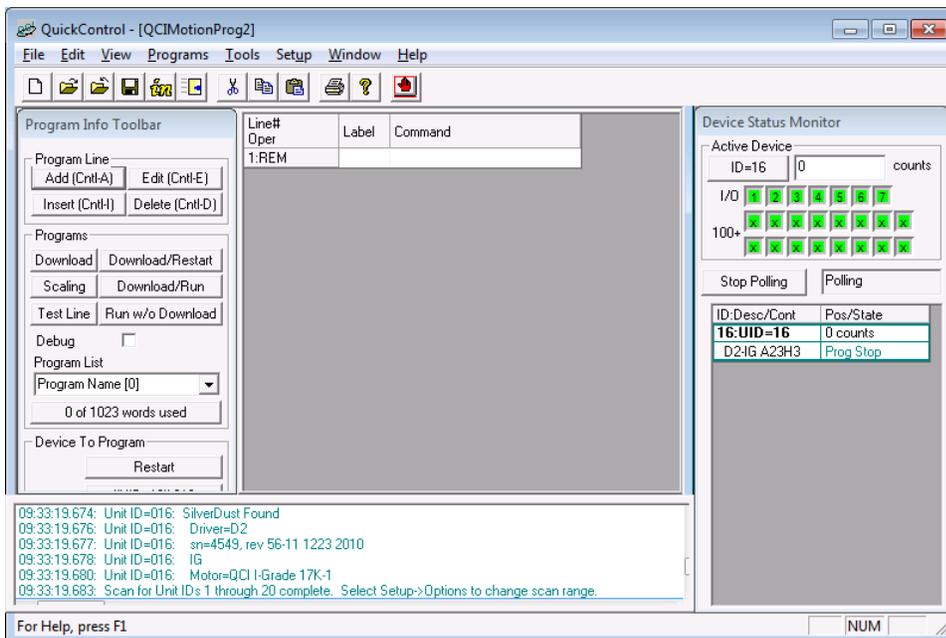
## Create and Run First Program

### A. Move 8000 Counts At Power Up

We will create a new program that causes the servo to move 8000 encoder counts when it powers up. You should be polling and have device showing in the left panel to enable you to see the position of the motor. If your status is not green, press Scan Network on the Device Status Monitor

Create a new program by selecting New Program File from the File menu.

### File -> New Program File



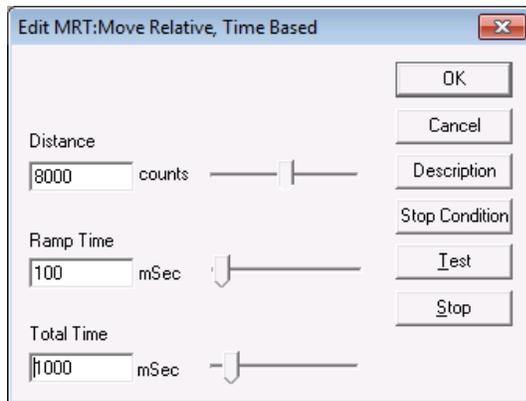
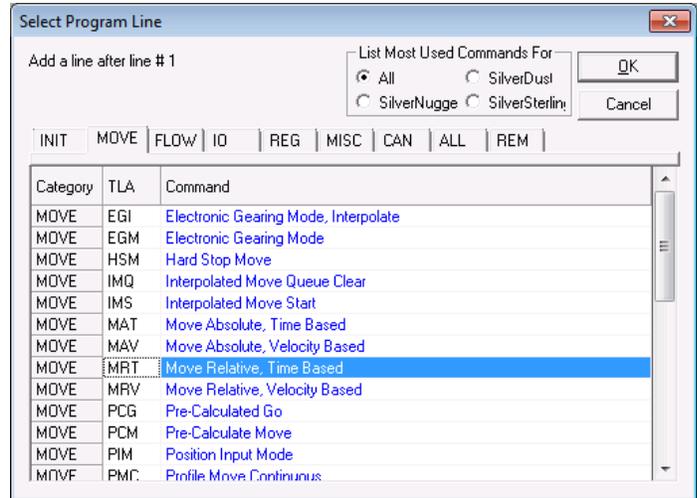
Your screen should look like this.

We will now add a Move Relative Time Based (MRT) command to the user program.

On the Program Info Toolbar (left side, approximately 1 inch down from the top), press **Add**.

The Select Program Line dialog box will appear. Select the MOVE tab.

Double click on MRT and the Edit Move dialog box will appear.



Edit the move data as shown. This will produce a relative move of 8000 counts in 1 second with a 100 millisecond ramp at the start and end of the motion. Press Test to preview the motion. The motor should spin.

Press the OK button to add this command to your program.

On the Program Info Toolbar, press Download/Restart. QuickControl will download your program to the servo's non-volatile memory. It will then restart the controller and go 8000 counts in 1 sec. You can press Restart to cause the program to execute again.

The program you just wrote will execute every time the servo powers up.

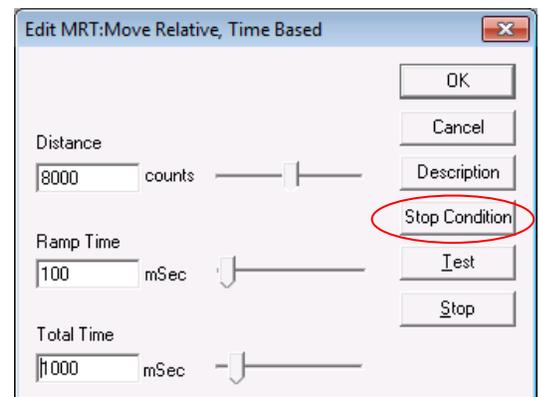
Select **Save As** from the **File** menu to name and save your program.

Congratulations! You are a programmer.

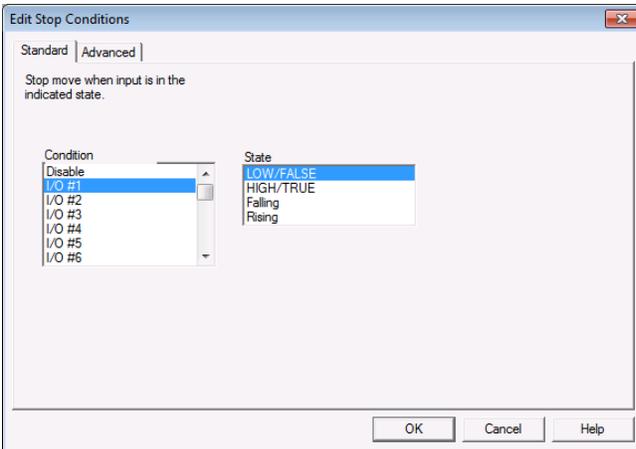
### B. Stop a Move Using a Digital Input

To make the move stop depending on the state of a digital input, the move needs to be edited.

- 1) Double click the MRT program line.
- 2) Press the Stop Condition button.
- 3) Select the Standard tab.



4) Select I/O #1 and LOW/FALSE. The dialog box should look this:



This selection will cause your move to stop when Input #1 is LOW.

5) Press OK to save and exit.

6) Press OK to exit the Edit Move dialog box.

7) Press the “Download/Restart” button on the Program Info Toolbar and verify the move stops when Input #1 goes LOW

8) Either press Restart, or power down and power up your servo, and verify the program runs again.

### C. Starting a Move Using a Digital Input

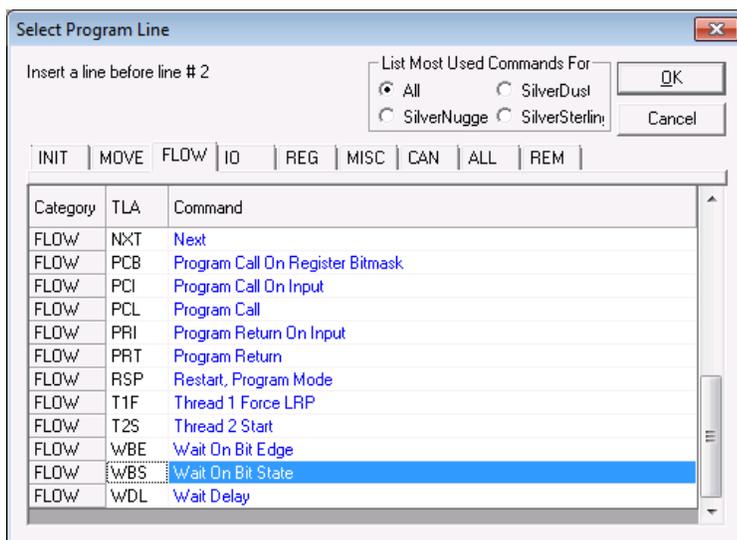
There are many ways to start a move using a digital input. Most of them involve adding a Program Flow command to the program. We will now add a command that will cause the servo to wait until Input #5 is LOW before executing our move. Note: if you do not have a physical switch, you can also simulate I/O using the Control Panel.

Open Control Panel, on the lower right side will be indicators for each of the IO bits on the controller. Clicking on the IO will set it to tristate (or off for the 24v IO). Holding the Shift button and clicking on the IO will set the IO (turn on the 24v IO), and holding the Ctrl key and clicking on the IO will clear IO low (turn off the 24 IO). The instructions are next to the indicators to help remind you.

1) With the MRT command selected, press the **Insert** button (top, left) to insert a line above the selected line, in this case, above line 2. (note: pressing Add will add a line below the selected line). The Select Program Line dialog box will appear.

2) Select the FLOW tab.

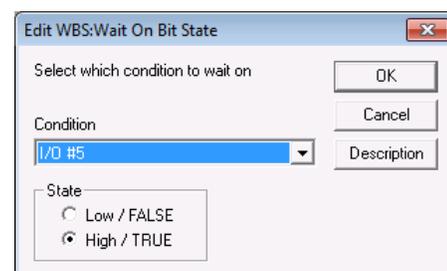
3) Double-Click on the WBS - Wait on Bit State - command. This command will cause the servo to wait at the program line until the Bit condition is met.



4) Enter the data as shown.

5) Press OK to add the command to the program.

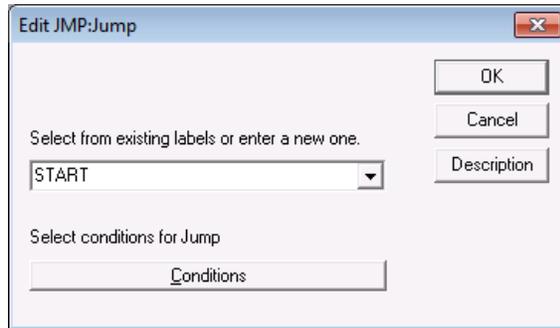
6) Set I/O 5 high, and then Press “Download/Restart” to run your program. Note, the servo does not move until Input #5 goes LOW and will stop early if Input #1 goes LOW.



#### D. Do Forever

To make the program repeat after it finishes the move, we will add a JMP command (FLOW) to the end.

- 1) Select the MRT program line.
- 2) Press the Add button.
- 3) Select the Flow tab.
- 4) Select the JMP command.
- 5) In the label field, enter the word "START" as shown on right.
- 6) Press OK to exit the Edit Jump Command dialog box.



- 7) You have just added a command to jump to the program line with the label "START". Add this label to line number 2 by clicking on the line 2 label cell and type "START". The JMP command will now loop to line 2 after it finishes the move.

Line#	Oper	Label	Command
1:	REM		
2:	WBS	START	Wait On Bit State Until "I/O #5" is HIGH/TRUE
3:	MRT		Move 8000 counts @ ramp time=100 mSec total time=1000 mSec Stop when "I/O #1" is LOW/FALSE
4:	JMP		Jump to "START"

- 8) With both Input #1 and Input #5 HIGH, run the program. Verify the motor does not spin until Input #5 goes LOW. Verify the motor executes its move until Input #1 goes LOW.

- 9) Save the program under a name of your choice. You can look at a fully documented version of the program by opening: QCI

Examples\ Using Inputs for Move Selection\Tutorial-Using an Input to Start and Stop a Move.qcp

# Chapter 2: Expanding Your Program

## Basic Motion Commands

There are four basic motion commands that are used to create simple motion profiles. These commands follow the same motion rules, but require different motion parameters. The result is a similar type of motion. The differences between the commands allow motion to be defined by relative or absolute distance, using velocity or time as the base. The commands pre-calculate the motion profile. The motion profile is subject to the maximum acceleration and velocity of the servo motor. A motion profile that the servo cannot numerically accomplish will cause a command error and the motion will not be executed. On the other hand, it is possible to program a motion profile that is numerically consistent, but requires more torque than available in the given servo.

### Relative Motion

The Move Relative, Velocity Based (MRV) and Move Relative, Time Based (MRT) commands are both relative moves. Relative motion is distance based, meaning that the first parameter is the overall distance to move. The starting position when the motion begins is irrelevant and the shaft will simply rotate the specified number of counts.

### Absolute Motion

The Move Absolute, Velocity Based (MAV) and Move Absolute, Time Based (MAT) commands are absolute moves. As long as the device is powered, it keeps track of its current location based on the zero point. The absolute position move is also based on the zero point. Please note that the zero point can be reset using the Zero Target and Position (ZTP) command, or moved/offset using the CLC “Sub Target Position” command. Absolute motion is position based, where the first parameter specified is the position to move to. The actual distance moved is the difference between the current position and the position specified in the command.

### Velocity Based Motion

The MRV and MAV commands are velocity based. The second and third parameters are velocity and acceleration. The servo will use the specified acceleration to achieve the velocity, and the same acceleration to bring the servo to a halt. If the acceleration is not sufficient to reach the specified velocity before deceleration must begin, the profile will be triangular rather than trapezoidal.

Note, acceleration and velocity must produce a ramp that is no more than a 7.86 seconds in duration for the move to be accepted. Slower ramps can be produced using the PMV command.

### Velocity Control

If an application requires only velocity control and no position designation, the Velocity Mode (VMP or VMI) commands should be used. VMP is the program type command, and VMI is the immediate type. The both Velocity Mode commands provide a “never-ending” motion. If no outside conditions interfere, the servo motor will continue to move at the specified velocity “forever”. (Note: if the final velocity is zero, the command will exit once zero velocity has been reached.) This command requires two parameters, velocity, and acceleration. A SilverLode servo will achieve the specified velocity using the given acceleration, and remain at that velocity until commanded otherwise. In a multitasking environment, this command can override any other motion currently in progress. This provides an easy

transition using a controlled deceleration to slow down or stop the servo. See Chapter 3 for more information on multitasking.

### Time Based Motion

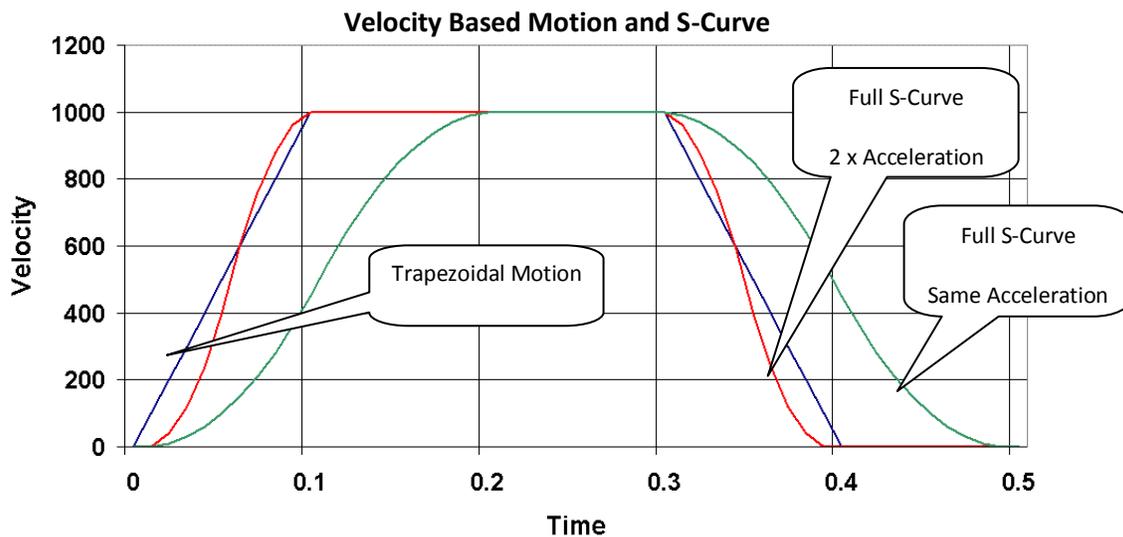
The MRT and MAT commands use time based parameters to create a motion profile. The second and third parameters are ramp time and total time. The total time specifies how quickly the move is to be completed. The ramp time sets the time to accelerate to velocity and the time to decelerate until it is stopped. The total time must be greater than twice the ramp time or errors will result and the move will not be executed.

Note: maximum ramp time is 7.86 seconds.

### S-Curve Factor

SCF determines the fraction of the acceleration/deceleration period that will be used to ramp up to and down from the needed acceleration (constant Jerk). A value of 0 defines a basic trapezoidal move, while a value of 32767 causes the entire acceleration /deceleration period to be used. A value of 3277 uses 10 percent of the acceleration/deceleration time to transition to/from the desired acceleration, lowering the required Jerk accordingly. A 10 percent value thus increases the maximum acceleration needed by a timed move by 10 percent, and the acceleration/deceleration time needed for a constant velocity/acceleration move by 10 percent.

A full s-curve will minimize the rate of change of acceleration (or jerk) for a trapezoidal motion. In a full s-curve time based move, the actual acceleration used is double that of a pure trapezoidal (zero s-curve) motion. In a velocity based move, the time to complete the move increases proportionally to the amount of s-curve specified. In order to have the full s-curve move complete in the same time as the trapezoidal move, the acceleration parameter must be doubled. The following chart shows a velocity based move with zero s-curve (trapezoidal), full s-curve, and full s-curve with the specified acceleration doubled.



Significant reduction in jerk in the system may be attained by a 20% or 50% s-curve (for example), where the acceleration ramps up over a portion of the curve, then is held constant for a portion, and then is ramped down again, with the same process occurring for the deceleration portion of the curve. A 20% s-curve factor value only adds 20% to the acceleration in a time based move or 20% to the time in a velocity based move, but can significantly reduce the jerk at the start and end of the move. The max value of the SCF command is 32767; a 20% value would thus be set to  $653 = 0.2 * 32767$ .



## Basic Motion Commands & Jump Commands

The purpose of this exercise is to get familiar with the basic MRV, JMP, and JOI commands. The servo will execute two moves depending on the state of the I/O #1 and I/O #3. The program runs in a continuous loop monitoring the two inputs.

1. Make sure QuickControl is polling the device. Press the Scan Network button in the middle of the Device Status Monitor to start polling the device if polling is stopped.
2. Select File → Open. Navigate to the "...\\QCI Examples\\Using Inputs for Move Selection\\" folder and select the file "Two Inputs Two Moves with MRV.qcp".
3. Put I/O #1 and #3 in the HIGH state. If you do not have physical switches, open the control panel and follow the instructions for the I/O portion on the control panel. Following the "set" instructions (hold the shift key down and left click the given IO indicator box) will set the I/O to a high state, while the "clear" instructions (hold down the CTRL key and left click) will set the I/O to a low state. Verify both inputs are in the high state by looking at the Input status boxes just above the Scan Network button. A Green box indicates the I/O line (in this case an input) is in the HIGH state. A Red box indicates a LOW state. Take a moment to toggle the I/O switches up & down to see the Input status boxes change color as the inputs change state.
4. Press the "Download/Restart" button on the Program Info Toolbar. Once the program is downloaded press the OK button. The program is now running.
5. Toggle I/O #1 LOW, then back to HIGH. The device will execute a simple move.
6. Toggle I/O #3 LOW, then back to HIGH. The device will execute a different move. Experiment with the I/O and notice the servo's motion. When finished close the active program.

Question: If BOTH inputs are LOW, which move gets executed? Why?



## Basic Velocity Mode

This exercise demonstrates the basic Velocity Mode, Program Mode (VMP) command. It illustrates how simple it is to have the device operate in a set velocity mode & stop on an Input trigger.

1. Select File → Open. Navigate to "...\\QCI Examples\\ Moves – basic\\Velocity Mode, Program Mode.qcp"
2. Press the "Download/Restart" button to download and begin execution of the program. Once the program is downloaded press the OK button.
3. The servo is now running in Velocity Mode. Notice the position counter window increasing the revolution count value as the device moves. Toggle I/O #5 LOW to stop motion. Press the Restart button to run again. When finished close the active program.

This program contains only one command. It has all the parameters needed for simple motion control.



## Creating, Downloading, and Running a Program in QuickControl®

In this exercise, a new program is created in QuickControl. It includes a short Move Relative, Velocity Based (MRV) move clockwise and a similar move counter-clockwise. I/O #1 can be triggered LOW to stop motion at any time.

### Creating a New Program

1. Select New Program File from the File menu to open a blank new program template.
2. Choose Add from the Program Info Toolbar to place a new program command in the list.
3. Choose the Move tab and double-click on MRV (Move Relative, Velocity Based) command.
4. Enter the following values for the first MRV move:  
Distance: -10000 counts  
  
Acceleration: 1000 cps/s (\*999.62 actual)  
  
Velocity: 2000 cps  
  
\*Acceleration parameters are rounded as QuickControl calculates the exact motion profile.
5. Press the Stop Condition button.
6. Choose the Standard tab of the Edit Stop Conditions box. Select I/O #1 for the Condition and use HIGH/TRUE as the State.
7. Press the OK button twice to get back to the main QuickControl screen.
8. Highlight the newly created Line 2 and choose Insert from the Program Info Toolbar. This will insert a new program command into the program list before the first MRV command on Line 2.
9. Choose the Move tab and double-click on MRV (Move Relative, Velocity Based) command.
10. Enter the following values for the second MRV move:  
Distance: 10000 counts  
  
Acceleration: 1000 cps/s (999.62 actual)  
  
Velocity: 2000 cps
11. Press the Stop Condition button.
12. Choose the Standard tab of the Edit Stop Conditions box. Select I/O #1 for the Condition and use LOW/FALSE as the State.
13. Press the OK button twice to get back to the main QuickControl screen.

### Downloading and Running the Program.

14. Before running the program, make sure I/O is ON or high.
15. Press "Download/Restart" on the Program Info Toolbar. This function will Download the program into the NV memory and restart the servo which will run the program.  
The program will only run once per Restart!
16. While the program is running, trigger I/O #1 to stop the move before the motion completes. Press the Restart button to run the program again.

### Saving the New QCP File

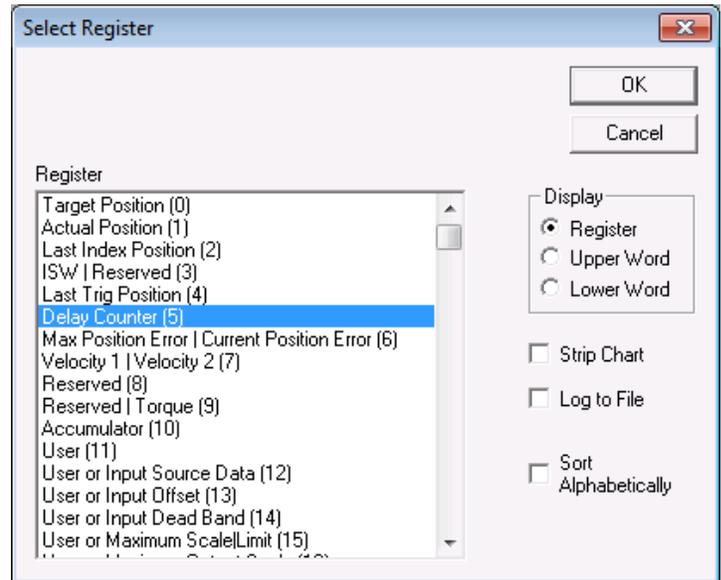
17. Using the File menu choose "File > Save As". This will open a standard file browser window where the program can be named and saved to the computer.  
Note: The browser window opened by "default" will be the last active folder accessed in QuickControl.
18. Choose a unique name for the new program and press the "Save" button.



## Using the Register Watch Tool

This exercise explores the operation and use of the Register Watch Tool.

1. QuickControl must be polling for the tool to function. If **Scan Network** button is visible in the right status panel, then polling is not active. To establish polling, press "Scan Network". The Register Watch Tool is started by selecting **Tools > Register Watch**. By default, no registers are displayed
2. To add a register to the display press the "Add Register" button. This brings up the Select Register window. Select the register to monitor/modify from the scrolling list. Some are combo-registers, where the upper 16 bits and lower 16 bits contain separate pieces of information. These registers are named (by default) 'high word | low word.' For example, register 7 contains two velocity values and is named "Velocity 1 | Velocity 2." To view only one half of register, select either 'Upper Word' or 'Lower Word' from the Display section.



3. Select register 5 'Delay Count' and 'Register' to view the timer register. Press OK to bring up the format window. Select the desired data format. Long and ULong display the exact contents of the register with no units. This is useful for inputting basic numbers, or for viewing SilverLode native units. The Hex option translates the contents from decimal to hexadecimal. This format allows easier analysis of individual register bits. Position, Acceleration, Velocity, and Time options all scale the register value using the current QuickControl units. Select 'Long' to display register 5 in native units.
4. Clicking OK a final time places the register into the tool. Click on the data box of register 5 (which currently contains a zero). Enter a numeric value of 10,000 or greater. Once the value has been typed, press enter. At that moment, the value typed in will be transmitted to the controller, and immediately begins decreasing. The Delay Count register is a specialized register that is automatically decremented each servo cycle. As demonstrated here, QuickControl is also constantly reading registers and reporting their contents in the register watch tool (while polling).
5. The register watch tool can be very powerful if applied well. Repeat the steps to add register 5 to the tool again, but select 'Time' in the format window.
6. The tool now displays register 5 twice. Data entered in either of the data fields will be transmitted to the servo. On the next QuickControl polling cycle, the other field will be updated and contain a scaled version of the edited field. This provides a convenient method for monitoring both SilverLode native units and the QuickControl scaled engineering units.

# Register Based Motion Commands

## Data Registers

Data Registers are used as data storage locations that may be used and modified by a host controller or by internal functions. There are a number of 32-bit data registers available within the device. They provide data storage for the parameters of register based commands. Some registers contain two separate pieces of data. These values are stored as two 16-bit numbers, one in the upper 16 bits of the register, and one in the lower 16 bits.

The standard motion commands (MRV, MRT, MAV, MAT) all have corresponding register and extended register versions. The trajectory for each of these moves is entirely pre-calculated; the contents of the register(s) are checked only once when the command is issued. Changing the value of the register(s) has no impact on a move already in progress. All values stored in the registers used by these commands must be in native units.

## Register Moves

(RRV, RRT, RAV, RAT) These register commands are the same as their standard counterparts, except that the first parameter (position/distance) is a register number, rather than a value. The value in the register, at the time of execution, is used to generate motion.

## Extended Register Moves

(XRV, XRT, XAV, XAT) The extended register version of the standard motion commands take three registers as parameters. The first specified register is used as the position/distance parameter. The following two registers are used for the second two parameters.

A SilverLode servo has other register based motion types. They are even more powerful, with changes in register values being applied on the fly. These commands are covered in detail in other chapters. Following is a quick listing of these command types.

## Profile Moves

All Profile Move commands (PMV, PMC, PMO, PMX) use the contents of registers 20-24 as parameters. See chapter 3 for more information on Profile Moves.

## Registered Step and Direction (RSD)

This command is operationally the same as Scaled Step and Direction (SSD), but the scale factor is retrieved from the specified register, to allow dynamic scaling. See chapter 4 for details on Step and Direction commands.

## Input Modes

Three commands, Position Input Mode (PIM), Velocity Input Mode (VIM), and Torque Input Mode (TIM), are used to access three special operating modes. The Input Modes use seven data registers for processing position, velocity, and torque information. They allow the servo to use data from an analog input or an external host to directly control motion. For example, this is a good way to configure the servo to respond to a joystick or analog input. These powerful commands are beyond the scope of this document. See Application Note " QCI-AN047 Input Mode - Joystick".



## Simple Register Based Motion

The servo will execute two moves depending on the state of the I/O #1 and I/O #3. The program runs in a continuous loop monitoring the two inputs. If data in the User Registers is modified, the motion profiles of the moves can be changed.

1. Select File > Open. Navigate to the “...\QCI Examples\Using Inputs for Move Selection\” folder and select the file “Two Inputs Two Moves with RRV.qcp”.
2. Open the Register Watch Tool (Tools > Register Watch). Press the “Add Register” button, select User [25], select Position for the Data Format, and press OK. Press the ‘Add Register’ Button again, select User [26], select Position for the Data Format, and press OK.
3. Press the “Download/Restart” button to download and begin execution of the program. Once the program is downloaded press the OK button.
4. Notice the values placed in the selected Data Registers by the program. Toggle I/O #1 LOW, then back to HIGH. The servo will execute a simple move.
5. Toggle I/O #3 LOW/HIGH. The servo will execute a different move.
6. Click once in the Data column of the User [25] Register. Enter the value 10 “revs” into the cell and push the Enter key on the keyboard. Toggle I/O #1 LOW/HIGH.
7. Click once in the Data column of the User [26] Register. Enter the value -100 “revs” into the cell and push the Enter key on the keyboard. Toggle I/O #3 LOW/HIGH.
8. Experiment with different values for the registers used in this position control example.
9. When finished close the active program.

Question: What type of applications can this program work in?



## Complete Register Based Motion

The purpose of this exercise is to get familiar with the basic XRV, JMP, and JOI commands. The servo will execute two moves depending on the state of the I/O #1 and I/O #3. The program runs in a continuous loop monitoring the two inputs. The complete motion profile of each move can be changed if data in the User Registers is modified.

1. Select File > Open. Navigate to the “...\QCI Examples\Using Inputs for Move Selection\” folder and select the file “Two Inputs Two Moves with XRV.qcp”.
2. Open the Register Watch Tool (Tools > Register Watch). Delete all listed Data Registers. Press the ‘Add Register’ Button, select User [25], select Position for the Data Format, and press OK. Press the ‘Add Register’ Button again, select User [26], select Acceleration for the Data Format, and press OK. Press the ‘Add Register’ Button again, select User [27], select Target Velocity for the Data Format, and press OK.
3. Press the ‘Add Register’ Button, select User [28], select Position for the Data Format, and press OK. Press the ‘Add Register’ Button again, select User [29], select Acceleration for the Data Format, and press OK. Press the ‘Add Register’ Button again, select User [30], select Target Velocity for the Data Format, and press OK.
4. Press the “Download/Restart” button to download and begin execution of the program. Once the program is downloaded press the OK button.
5. Notice the values placed in the selected Data Registers by the program. Toggle I/O #1 LOW/HIGH. The servo will execute a simple move.
6. Toggle I/O #3 LOW/HIGH. The servo will execute a different move.
7. Click once in the Data column of the User [25] Register. Enter the position value 100 “revs” into the cell and push the Enter key on the keyboard.
8. Click once in the Data column of the User [26] Register. Enter the acceleration value 150 “rps/s” into the cell and push the Enter key on the keyboard.
9. Click once in the Data column of the User [27] Register. Enter the velocity value 25 “rps” into the cell and push the Enter key on the keyboard. Toggle I/O #1 LOW/HIGH.  
Note: Acceleration Range is 0 to 277.78 rps/s. Velocity Range is 0 to 66.66 rps (4000 rpm).
10. Modify the Position User [28], Acceleration User [29] & Velocity User [30] data for the second move. Toggle I/O #3 LOW/HIGH
11. Experiment with different values.
12. When finished close the active program, delete all registers on the register list & close the Register Watch Tool.

# Program Debugging

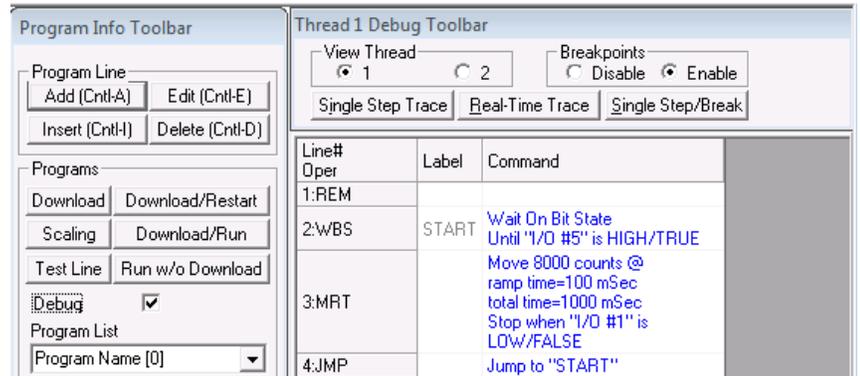
QuickControl provides several debug tools to aid in program development.

## Debug Mode

Enter Debug Mode by checking the Debug checkbox in the Program Info Toolbar.

While in Debug mode, QuickControl will highlight the program line as follows:

- Program Line Being Executed: Highlight Yellow and italic TLA
- Program Line Ready for Single Step: Highlight White and bold TLA



## Single Step/Break

Press the Single Step button to download the active program and single step (or execute) the first line. Every time Single Step is pressed, the next line will be executed.

If Single Step is pressed while a program is running, the program will break wherever it is and await a Single Step command.

## Single Step Trace

Press Single Step Trace to have QuickControl continually "push" Single Step button for you. This lets you observe the program executing at a reduce speed.

If Trace is pushed with no program running, the active program will be downloaded and QuickControl will start "tracing" the program.

## Real-Time Trace

Press this button to trace the program in real-time. Due to the delays in serial communications, the Real-Time Trace may appear to "skip" over lines.

## Breakpoints

Multiple Breakpoints can be set in the program by highlighting the line and pressing F9. If the program encounters a Breakpoint while doing a Trace, it will stop tracing and go into Single Step mode.

Select the Enable/Disable Breakpoints to Enable or Disable all Breakpoints.

## Real-Time Breakpoints

For SilverDust only, the Breakpoints will stop Real-Time Traces and go into Single Step mode.

Real-Time breakpoints can be temporarily set and cleared while in Single Step mode. The next time the program is loaded from non-volatile memory, the breakpoints will reset back to what they were a download time.

## View Thread

The View Thread radio buttons allow users to debug devices running both Thread 1 and Thread 2 programs. Each thread can be debugged independently of the other thread. For example, the user may select Real-Time Trace for Thread 1, use View Thread to switch to Thread 2, and start single stepping through the current Thread 2 program.

## Test Line

The Test Line button on the Program Info Toolbar is useful to send the selected program line to the device as an immediate command.

## Jump To Line

This feature is accessible from either the Programs menu or by right clicking on a line. If this command is selected while in Single Step mode, program execution will jump to the selected line. This is very useful if you want to “re-do” a specific section of code or test a section of code that is hard to get to.



## Debugging a (QCP) QuickControl Program

This exercise is an introduction to the program troubleshooting capabilities of QuickControl. These functions provide a powerful means to check the operation of a program, and understand how individual commands execute.

The default QCI Examples folder is located at: C:\Program Files\QuickControl\QCI Examples

The Red Stop Hand Icon on the Main Icon Toolbar will allow the user to quit operation of any Tool.

## Single Step

1. Using the File menu, open the following qcp program: “...\QCI Examples\Moves-basic\Move Examples.qcp”.
2. Check Debug on the Program Info Toolbar (left panel approximately half way down) to enter Debug Mode.
3. Press the **Single Step/Break** button on the Debug Toolbar. This will initiate the execution of the first command found in the program.
4. Continue pressing **Single Step/Break** until the end of the program file. The line which is next to be executed will be highlighted in yellow/blue, a line which is still executing will be highlighted in white/blue. Press the Red **Stop** Hand Icon to stop the program.

## Single Step Trace

Use the Single Step Trace tool on the same QuickControl program file.

1. While still in Debug Mode, click on the first line of the program, and then press **Single Step Trace**. This will initiate an automatic step-by-step execution of the program commands, one line at a time.
2. The Trace tool will continue through the program until it reaches either a Breakpoint, a command error or the end of the program. The program will run significantly slower than real time operation. When complete, press the Red Stop Hand Icon.

## Real-Time Trace

We will now use the Real-Time Trace tool on the same QuickControl program file.

1. While still in Debug Mode, click on the first line of the program, and then press **Real-Time Trace**. This will start the program, running at normal speed while highlighting the current line as time allows. As the program can proceed faster than the serial port can, report the events, not all lines will highlight unless the program dwells

on them for a sufficiently long time, such as delay commands, wait on inputs, or motion commands. The advantage of Real-Time trace is that it *is* real-time - it does not slow down the execution.

2. The Trace tool will continue through the program until it reaches either a Breakpoint, a command error or the end of the program. When complete, press the Red Stop Hand Icon.

## Breakpoints

Breakpoints are set on lines by highlighting the desired line and selecting Toggle Breakpoint from the Programs menu. The line should change color to Red. The function key F9 is a shortcut to toggle breakpoints on & off. Breakpoints are stored into non-volatile memory, but are only acted upon if breakpoints are enabled. Temporary breakpoints can be set when a program is paused in single step mode. The temporary breakpoints only remain while the same program is in the command buffer and go away if a new sub program is loaded. The breakpoints downloaded to non-volatile will be saved with the program. Breakpoints allow you to run a program in real-time until the section of interest is reached.

1. Set a breakpoint on line 6 and press **Single Step Trace**. QuickControl will step through the program and then stop at the selected breakpoint.

## Test Line

The Test Line tool allows any one line in a program list to be executed. It only Tests (executes) the selected line (command) and does NOT run the entire program.

1. Press the Red **Stop** Hand Icon.
2. Using the same open qcp program file, click on a MAT: Move Absolute, Time Based command and then press **Test Line** on the Program Info Toolbar.
3. QuickControl should have executed the chosen command without executing other commands.

## Jump to Line

When the program is paused in single step/break mode, right clicking on a desired execution continue point and selecting Jump to Line will cause the program to continue running from the new selected line.



## Cut, Copy & Paste Programming

This exercise provides the user a technique for building up an entire motion profile. Several QCI Example programs are opened up and the entire list of commands is copied into a “New” program.

1. Select File > Open. Navigate to the “...\QCI Examples\Homing\” folder and select the file “Homing against a hard stop.qcp”. Select File > Save As. Enter the filename testfile.qcp.
2. Select Programs > Program Details. Edit the program name to be “home”.
3. Select Programs > New Program. Enter the program name, “move”. Repeat again using the program name “PMV”.
4. Verify all three (3) programs are in the Program List of the Program Info Toolbar. Click on the small down arrow button to see the list contents. If all three are listed, save the file again. If all programs are not listed, repeat Step 3, then save the file.
5. Select File > Open. Navigate to the “...\QCI Examples\ Profile Moves\” folder and select the file “Profile Move Continuous.qcp”. Select Edit > Select All. Select Edit > Copy.
6. Select Window > testfile.qcp. From the Program List, choose the “PMV (2)” program.
  - a. Highlight line 1 - REM.
  - b. Select Edit > paste. The Profile Move Continuous program should be visible.
  - c. Highlight the last line of the program.
  - d. Press Add, choose the Flow tab, double click on the LRP command, press the button, and choose the “home” program. Select File > Save.
  - e. Click once in the Label column of the new LRP command. Enter the text “home” and push the Enter key on the keyboard.
  - f. Put a label on the first line of the program by clicking in the "Label" column and typing the label name in. Type “PMV”. Now, highlight the last line of the program.
  - g. Press Insert, choose the Flow tab, and double click on the JOI - Jump on Input command. Choose “HOME” from the Program List. Press the Conditions button, choose “I/O #3” from the Condition list and “LOW / FALSE” as the State. Press OK twice to get back to the program.
  - h. Highlight the last line of the program. Press Insert, choose the Flow tab, and double click on the JMP - Jump command. Choose “PMV” from the Program List. Press OK to return. Select File > Save.
7. From the Program List, choose the “home (0)” program. Highlight the last line of the program.
  - a. Press Add, choose the Flow tab, double click on the LRP command, press on the button, and choose the “move” program. Select File > Save.
8. Make sure all inputs are in the HIGH state and press the “Download/Restart” button to download and begin execution of the program. Once the program is downloaded press the OK button.

Review: The program begins by running the homing to hard stop program, followed by the Profile Move Continuous program. The program will continue to run until I/O #3 = LOW, then the entire process repeats. Press the Red Stop Hand Icon to end.

# Chapter 3: Unique Features and Commands

## Status Words

QuickControl comes with a diagnostic toolbar for real time status updates of the device, for ease in troubleshooting any potential programming or physical issues that may arise. Three special status words provide a wealth of information about the operating state of the device: the Polling Status Word (PSW), the Internal Status Word (ISW), and the I/O Status Word (IOS). The information contained in the 16 bits of these words can provide information not available anywhere else.

### Polling Status Word (PSW)

The Polling Status Word indicates the overall condition of the device. The meaning of each bit in the Polling Status Word (PSW) is explained in the table below. The description for each bit describes the device condition indicated by a “1” in that bit.

Bit#	Latched?	Definition	Description
15	Yes	I-Cmd Done	Immediate Command Done (i.e. Host Command).
14	Yes	NV Mem Error	There was a checksum error while reading data from or to the non-volatile memory. (SilverDust Rev 06 adds write protection to certain regions.)
13	Yes	P-Cmd Done	All commands active in the Program Buffer finished executing.
12	Yes	Command Error	There was an error associated with the command execution. Unreasonable parameter values or not support in this firmware
11	Yes	Input Found	The motion ended when the selected exit/stop condition was met.
10	Yes	Low/Over Volt	A low or over voltage error occurred.
9	Yes	Holding Error	Holding error limit set by the Error Limits (ERL) command was exceeded during a holding control state.
8	Yes	Moving Error	Moving error limit set with the ERL command was exceeded with the device in a moving control state.
7	Yes	Rx Overflow	Device serial receive (UART) buffer overflowed.
6	Yes	CKS Cond Met	A condition was met while executing a CKS command. One of the conditions set with the Check Internal Status (CKS) command was met.
5	Yes	Msg Too Long	The received message was too big for the Serial Buffer. Device rx packet > 31 bytes
4	Yes	Framing Error	There was a packet framing error in a received byte. Device rx byte with missing bit.
3	Yes	Kill Motor	The device was shut down due to one or more conditions set with the Kill Motor Condition (KMC) command (or KMX command for SilverDust Rev 06). Latched yes, but after CPL it clears even if there is still a KMC till present.
2	Yes	Soft Limit	A soft stop limit was reached as set by the Soft Stop Limit (SSL) command.
1	Yes	Rx Checksum	Device rx packet with an invalid checksum. Valid for 9 Bit Binary and Modbus only.
0	Yes	Aborted Pkt	There was a data error or a new packet was received before the last packet was complete.

The PSW is used to give the host a quick status update. A host may be programmed to act on changes in the PSW, or the word could simply be used to alert the host to changes in the operating condition of the device. The Polling function (POL Command) continually checks the PSW and is a common way to update an external host on the condition of the device. Two Immediate commands are associated with the Polling Status Word: the Poll (POL) command and the Clear Poll (CPL) command. The SilverDust (Rev 06) adds an additional command, the Poll with Response (POR).

### ***Poll (POL) Command***

The POL command is used to determine the condition of the device, by accessing the information in the bits of the Polling Status Word (PSW). A POL command can be executed at any time, including while the device is in motion. Executing this command will cause the device to return an ACK message if all of the bits in the word are cleared, or return the word itself in hexadecimal if any of the bits are set. An ACK is an Acknowledgement signal from the device that indicates that the command was received correctly with no errors but no reply was required. The POR command (SilverDust Rev 06) operates similarly to the POL command, but always returns a hexadecimal word, even if it is zero. See Technical Document QCI-TD053 Serial Communications on our website for more information on Serial Communications.

### ***Clear Poll (CPL) Command***

All of the bits are latched, meaning that they must be cleared manually; they are not cleared when the condition they are tied to clears. The CPL command is used to clear the bits of the Polling Status Word (PSW). This command is the only way to clear the bits in the PSW since all the bits are latched. Additional conditions that occur after a POL command is issued will show up when the next POL command is issued, even if those bits have been cleared by an intervening CPL command (i.e. the data is double buffered and the bits cannot be cleared until they have been read).

See Technical Document QCI-TD053 Serial Communications on our website for more information.

## I/O Status Word (IOS)

The I/O Status Word (IOS) indicates the states of the I/O lines, as well as several specific internal conditions.

The meaning of each bit in the I/O Status Word (IOS) is explained in the table below. The description for each bit, except Bit 7, indicates the condition indicated by a “1” in that bit.

Bit#	Latched?	Definition	Description
15	No	I/O #7	Status of I/O line #7 (normally high, low when triggered).
14	No	I/O #6	Status of I/O line #6 (normally high, low when triggered).
13	No	I/O #5	Status of I/O line #5 (normally high, low when triggered).
12	No	I/O #4	Status of I/O line #4 (normally high, low when triggered).
11	Yes	Reserved	
10	No	Delay Counter	Status of the Delay Counter. High when counting down, low when count is expired.
9	No	Holding Error	Holding error limit set with the Error Limits (ERL) command has been exceeded with the device in a holding control state.
8	No	Moving Error	Moving error limit set with the ERL command has been exceeded with the device in a moving control state.
7	No	Temperature Ok	Internal sensor determines over temperature condition and <u>clears</u> this bit when true. Bit is shared by drive enable lines on 34 frame servos. (disable = high, enable = low.)
6	No	I/O #3	Status of I/O line #3 (normally high, low when triggered).
5	No	I/O #2	Status of I/O line #2 (normally high, low when triggered).
4	No	I/O #1	Status of I/O line #1 (normally high, low when triggered).
3	No	Trajectory Generator Active	When the Trajectory Generator is active, the device is calculating motion. (i.e. a move is in progress)
2	No	External Index Found	An index mark on an external encoder has been detected.*
1	No	Internal Index Found	An index mark on an internal encoder has been detected.*
0	No	Index Found	An index mark has been detected on the selected encoder (external or internal).*

\*Only one 120uSec cycle wide.

Note: Latched bits can be cleared using the CIS command.

This word is accessible in two ways: with the Read I/O States (RIO) Immediate command sent from an external host, and through a large number of different Program commands that are available for flow and motion control in programs. When accessed by a host, the IOS allows the host to view information that can supplement the information available from the PSW. As with that word, a host could be programmed to act on changes in the IOS or it could be used for monitoring. The IOS can be accessed from the lower half of register 209. (SilverDust Rev 06) The I/O may also be configured by the host, even while a program is running on the SilverDust by using the Configure IO, Immediate Mode (CII).

### Read I/O States (RIO) Command

The RIO command is similar to the Poll (POL) command. It can be issued by a host to gain access to the I/O Status Word (IOS). The RIO command can be executed at any time and returns the hexadecimal form of the 16-bit binary number that makes up the IOS. None of the bits are latched, so the number returned with the RIO command will represent current information.

### Jump and Motion Commands

Most jump commands used by programs use the IOS, while the most motion commands use the Internal Status Word (ISW) for their stop conditions. The jump commands use the IOS to define the conditions for the jump. The motion commands use the ISW to define basic stop conditions (for example, stop motion if I/O #3 goes HIGH). Using inputs for program flow and motion control is discussed further in Chapter 4.

### Internal Status Word (ISW)

The meaning of each bit in the ISW is explained in the table below. The description for each bit indicates the condition indicated by a “1” in that bit. ISW is the upper word of register 3.

Bit#	Latched?	Definition	Description
15	-	Reserved	Reserved bit.
14	Yes	Low Voltage	Low voltage error has occurred. Defined by LVT command.
13	Yes	Over Voltage	Over voltage error has occurred. Defined by OVT command.
12	Yes	Wait Delay Exhausted	The wait delay timer has expired.
11	Yes	Input Found On Last Move	An input used as a stop condition was found during the last move. Latched but automatically cleared on next move.
10	Yes	Halt Command Sent	The halt command (HLT) was received by the device.
9	Yes	Holding Error	Holding error limit set with the Error Limits (ERL) command has been exceeded with the device in a holding control state.
8	Yes	Moving Error	Moving error limit set with the ERL command has been exceeded with the device in a moving control state.
7	No	Temperature Ok	Internal sensor determines over temperature condition and <u>clears</u> this bit when true. Bit is shared by drive enable lines on 34 frame servos. (disable = high, enable = low.)
6	No	I/O #3	Status of I/O line #3 (normally high, low when triggered).
5	No	I/O #2	Status of I/O line #2 (normally high, low when triggered).
4	No	I/O #1	Status of I/O line #1 (normally high, low when triggered).
3	No	Negative Calculation Result	Result of the last calculation was negative. CLC command.
2	No	Positive Calculation Result	Result of the last calculation was positive. CLC command.
1	No	Zero Calculation Result	Result of the last calculation was zero. CLC command.
0	Yes	Index Found	An index has been detected.

Latched bits can be cleared using the CIS command.

The ISW is available in a data register (upper word of register 3), making it very useful for troubleshooting. An external host can use this status word in several ways. A host can use the Read Internal Status Word (RIS) and Clear Internal Status Word (CIS) commands to work directly with the Internal Status Word (ISW). A host checking the Polling Status Word (PSW) can access the ISW by using

the Check Internal Status (CKS) command. Finally, a program can use the ISW by checking a data register.

### ***Read Internal Status Word (RIS) Command***

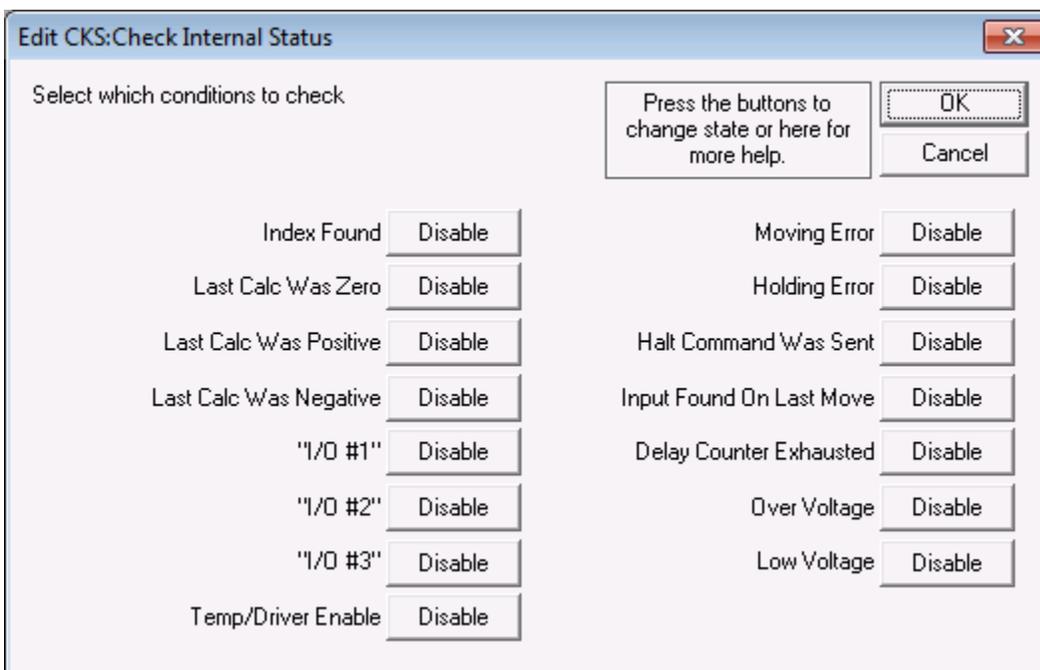
The RIS command is similar to the RIO and POL commands. It is issued by an external host to gain access to the ISW. The RIS command returns the decimal form of the 16-bit binary number that makes up the ISW. Some of the bits are latched while some are not. The bits that are not latched represent the status of the condition associated with them, while the latched bits function as flags to indicate that the associated condition occurred at some time after the last time the flag was cleared.

### ***Clear Internal Status Word (CIS) Command***

The CIS command is used to clear the latched bits of the ISW and IS2. Latched bits must be cleared with this command in order to be reset. This makes the CIS command an important part of the Kill Motor feature. See Technical Document QCI-TD052 Shutdown And Recovery on our website.

### ***Check Internal Status (CKS) Command***

Bit #6 of the Polling Status Word (PSW) shows the CKS status. This allows a host that is polling a device to indirectly use the ISW just by polling. The CKS command sets the conditions that will trigger the Polling Status Word bit. There are 15 conditions for the CKS command to match the 15 bits used in the ISW. The conditions are toggled on and off and then combined with a logical OR. This means that if any of the conditions set with the CKS are true, bit #6 on the Polling Status Word will be set. The purpose of this is to add functionality to the Polling Status Word by allowing one or more conditions of the ISW to be flagged. If a host detects that bit #6 on the Polling Status Word is set, the host could then check the ISW—the host essentially checks the ISW and Polling Status Word using just the Polling Status Word.



## Internal Status Word 2 (IS2) Description

The meaning of each bit in the IS2 is explained in the table below. The description for each bit indicates the condition indicated by a “1” in that bit. IS2 is the upper word of register 236.

Bit#	Latched?	SD	Definition	Description
15	No	6	I/O #7	I/O #7 bit
14	No	6	I/O #6	I/O #6 bit
13	No	6	I/O #5	I/O #5 bit
12	No	6	I/O #4	I/O #4 bit
11	No	25	Thread 2 Running	Thread 2 Running
10	No	25	CAN Comm Error	CAN Communication Error See 2000h Critical Error Mask in CANOpen® User Manual for details.
9	Yes	6	Extended IO Fault	Check data passed through the isolation barrier to the extended IO and back to the processor was corrupted. Indicates loss of 24v IO power and or ground.
8	Yes	6	Encoder Fault	Encoder signals analog levels out of specification, indicating the connector is loose or not connected, or signals are open, shorted, misconnected, or having excessive interference.
7	Yes	6	Drv Temp Fault-Digital	Driver internal sensor indicates over temperature. SN N3: Over current and over temp
6	Yes	11	H/W Drv Disabled	HIGH = Driver Enable Line LOW=Driver Disabled LOW = Driver Enable Line HIGH=Driver Enabled
5	-	11	Drv Temp Fault-Analog	SN N3: Driver analog sensor indicates over temperature.
4	Yes	11	Motor Temp Fault	Temperature sensor in motor (I-Grade, IP-65) indicates over temperature.
3	No*	11	Fac Blk Drv Disabled	Driver disabled during Factory Block execution (startup) due to one of the following conditions: <ul style="list-style-type: none"> <li>Motor miss-match: Attached motor type does not match what controller was initialized for.</li> <li>Bad read/bad data in motor memory.</li> </ul>
2	Yes	6	Encoder Re-phased	Excessive lost encoder pulses detected between index pulses. Encoder-motor phasing redone. May be excessive noise, encoder problems, or wiring problems.
1	Yes	11	Motion Limit Fault	HIGH indicates that motion is being limited by one of the following: <ul style="list-style-type: none"> <li>Soft Stop Limits (SSL)</li> <li>Velocity Limits (VLL)</li> </ul>
0	No	8	Count Down Timer Active	HIGH indicates Millisecond countdown register (Register 245) is not zero.

Latched bits can be cleared using the CIS command.

\*Bit reflects Factory Driver disable bit and is not cleared by a Clear Status Word command.

### *Internal Status Word 3 (IS3) Description*

The meaning of each bit in the IS3 is explained in the table below. The description for each bit indicates the condition indicated by a “1” in that bit. IS3 is the lower word of register 203.

Bit#	Latched?	Definition	Description
15	No	Reserved	
14	No	Reserved	
13	No	Multi-Tasking Active	1=active
12	No	Driver Enabled at KMR Entry	1 allows driver to remain on while in Kill Motor Recovery program if multitasking is active when fault occurs.
11	No	Driver Disabled-DMD	1=driver disabled via enable/disable command or because of fault.
10	No	Reserved	
9	No	Reserved	
8	No	Reserved	
7	No	Reserved	
6	No	Anti-Hunt Mode	0=Standard, 1=Allow Anti-Hunt while motion active
5	No	Servo Loop Mode	Mode set by Closed Loop Mode (CLM) command 0=Position, 1=Velocity
4	No	Index Phase Alignment	Index Phase Alignment was enabled during initialization
3	No	Reserved	
2	No	Index Phase Alignment Fault	0=Normal, 1=realignment from index has occurred (normally due to encoder counts lost)
1	No	Reserved	
0	No	Reserved	

### **Internal Status Word 4 (IS4) Description (SD 05)**

The meaning of each bit in the IS4 is explained in the table below. The description for each bit indicates the condition indicated by a “1” in that bit. IS4 is the lower word of register 236.

Bit#	Latched?	Definition	Description
15	No	Reserved	
14	No	Reserved	
13	No	Reserved	
12	No	Reserved	
11	No	CAN HW Present	CAN only enabled if CAN hardware is present
10	No	Extended IO HW Enabled	8 or 16 bits of extended IO
9	No	Reserved	
8	No	DC Motor Mode Enabled	Set by DC motor mode command
7	No	CAN Rotary Switch Present	1=hardware present
6	No	Thread 2 Running	1=Thread 2 running
5	No	Reserved	
4	No	PVIA Vel & Acc	0=Filters,1=Observer
3	No	Driver Disabled by Factory Block Fault	1=disabled. Motor type indicated from motor memory does not match motor type configured in SilverDust.
2	No	Breakpoints Enabled	Allows breakpoints flagged in code to trigger break
1	No	Encoder Runout Adjustment	Encoder runout has been calibrated, and is in operation.
0	No	Index Phase Alignment -	At least one index pulse has been detected since closed loop,

### **Extended IO Word (XIO) Description (SD)**

The SilverDust D2-IGB provides 16 additional I/O, and the D2-IG-8 provides 8 additional IO. The input and output state of these IO are available in Register 238. The upper word provides the debounced (See DIF command) levels read back from the extended I/O, while the lower word provides the commanded levels sent to the I/O. The upper word reports a voltage at the I/O pin of greater than 1.5v as a 1 and below 1.5v as a 0. The lower word reports the requested drive level of the extended IO: a 0 indicates that the output is off (floating), while a 1 indicates that the output is active (pulling the IO to ground). This register may be read, written, or modified by the host using standard register commands. IO may also be individually set using the various IO commands. Note that the lower word (output) will retain the written data, while the upper word will return to the actual detected levels the following cycle.

## Error Limits and Drag Mode

The ERL command is used to change four settings. It sets the moving and holding error limit conditions, sets the delay time for switching from moving torque to holding torque, and enables or disables a special operating mode called Drag Mode. Drag Mode helps eliminate servo wind-up and allows the device to emulate a mechanical slip clutch.

### Error Limits Command Parameters

The Moving Error Limit, Holding Error Limit, and Delay to Holding time are all set by the ERL command. The three parameters determine the conditions under which the device reports a moving or holding error. A SilverLode servo reports a holding or moving error by setting a bit in the Internal Status Word, the I/O Status Word, and the Polling Status Word. There is a separate bit in each status word for moving and holding error. The Delay to Holding Time is intended to set a delay following a motion for the system to settle to a smaller error limit following the motion.

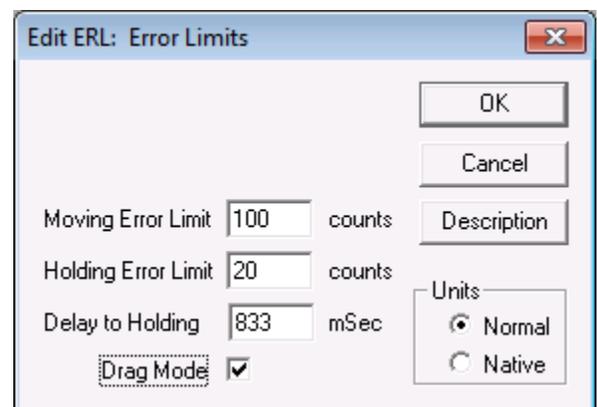
### Error Limits Operation

A SilverLode servo applies moving torque whenever the Trajectory Generator is active. While the Trajectory Generator is active, the servo compares the position error with the moving error limit. Position error is calculated by subtracting the target position (set by the Trajectory Generator), from the actual position (read from the encoder or external feedback device). If the magnitude of the position error exceeds the moving error limit, the moving error bit in the three status words is set. The bit in the Internal Status Word (ISW) may be used to automatically shutdown the servo (see Technical Document QCI-TD052 Shutdown And Recovery on our website), while the bit in the Polling Status Word (PSW) and I/O status Word (IOS) can alert an external host to the error condition if the device is being polled.

At the end of a move, the Trajectory Generator goes inactive and stops changing the target position. When this happens, the device starts the Delay to Holding timer set up with the ERL command. Until this timer expires, the device continues using the moving torque limits and checking the moving error limit. After it expires, the device applies holding torque and uses the Holding Error Limit to check for a position error while holding position. The holding error is reported using a holding error bit in the three status words.

### Drag Mode

Servo systems can suffer from a problem called “position wind-up”. This condition occurs when a servo motor cannot keep up with a move or is stalled by a mechanical jam. When the jam is released and the shaft is free to move again, the position error can be huge, and the servo might spin at a very high speed to catch up. This high-speed reaction occurs because of the basic torque calculation that virtually all servo motors (including SilverLode servos) perform as part of their control equation. If position error is very large, then so is the torque response. The simplest way to prevent this response is to shut down the servo motor. This may be fine if the shutdown is due to a jam, but there are cases where a shutdown is not acceptable.



An alternative to shutdown would be to limit this position error so it never becomes large enough to cause a problem. Drag Mode allows the control algorithm to forgive excessive position error and prevents extreme motion. Setting smaller error limits with drag mode enabled allows the unit to “slip” when the error is exceeded emulating a slip clutch.

Drag Mode is enabled with the ERL command. In Drag Mode, when an error limit is reached (moving or holding), the device adjusts the target position so that it differs from the actual position by no more than the error limit. However, this can cause the final position to fall far short of the original target position if one of the basic move commands is used (MRV, MRT, MAV, or MAT). Profile Move Commands, eliminate this problem and the original target is achieved even with Drag Mode enabled. This compatibility with Drag Mode is one of the benefits of using Profile Move commands.

Note, the Holding and Moving status bits are still set in the Internal Status Word (ISW) and will cause a shutdown if they are used in the kill motor condition commands KMC and/or KMX. Profile Move Operation

The Profile Move commands add a new dimension to the servo by allowing the move parameters to be changed while a motion is in process. This gives the servo the ability to create custom motions as required.

Profile Move commands can perform very complex motion profiles by allowing the move parameters to be changed dynamically. Move parameters (stored in Data Registers) can be changed by an external Host controller or by an internal program.

There are two Profile Move commands:

1. For a single move; use the Profile Move (PMV) command to execute a single move where the command ends when the target position is reached.
2. For a continuous move; use the Profile Move Continuous (PMC) command to execute a move that does not stop when the target position is reached. Once in position, this operation will wait until the position parameter is changed so there is no need to reissue a move command. Multi-Tasking must be enabled for PMC to function properly. This continuous move can be terminated with a Stop on Input condition, or the stop techniques mentioned previously.

Both Profile Move commands use Data Registers #20 to #24 for parameter storage. The Profile Move commands use linear acceleration and deceleration parameters, where a separate deceleration parameter is provided for different acceleration and deceleration profiles. The S-Curve Factor (SCF) command does not work with the Profile Move command. Profile Move commands also use an offset parameter, which causes the servo to move an offset distance after a Stop on Input condition is met.

Register Number	Description	Comment
20	Position	Absolute destination value.
21	Acceleration	Sets the acceleration rate that is used when increasing the move speed.
22	Velocity	The maximum speed that is allowed during a move.
23	Deceleration	Sets the deceleration rate that is used when decreasing the move speed.
24	Offset	When a stop condition is met, this value is added to the current position and copied to Register 20 for a profiled stop. If set to zero, only the deceleration values is used to ramp down to a stop.

Using both the Enable Multi-Tasking (EMT) and Calculation (CLC) commands with the PMV and PMC commands allows the functionality to create custom motion profiles similar to the one shown. In the example program shown to the right, a PMV command is automatically updated. The velocity register is being incremented by the CLC command, which increases the velocity every 2 seconds.

Line# Oper	Label	Command
1:EMT		Enable Multi-Tasking
2:WRP		Write 200 rev to User I Profile Mo 0] Register
3:WRP		Write 0.5 rps/s to User I Profile Move Acc[21] Register
4:WRP		Write 1 rps to User I Profile Move Vel[22] Register
5:WRP		Write 40 rps/s to User I Profile Move Dec[23] Register
6:WRP		Write 0 rev to User I Profile Move Offset[24] Register
7:PMV		Profile Move:
8:DLY	LOOP	Delay for 2000 mSec
9:CLC		Increment User I Profile Move Vel[22]
10:JMP		Jump to "LOOP"

### Related Profile Move Commands

The Profile Move Override (PMO) command overrides any other motion currently in progress and executes a PMV command, using the parameters loaded into the Profile Move registers (#20 to #24). When this command follows a PMC command, the PMC operation will end when the target position is reached, effectively changing the functionality of PMC command to act like the simpler PMV command. Normally, the PMC command will not end unless explicitly stopped by a stop condition. Using the PMO command after a standard PMV command will have no effect, other than using any new stopping conditions contained in the PMO command. PMO will also override other modes if Multi-Tasking is enabled (e.g. Step and Direction).

The Profile Move Exit (PMX) command will stop any profile move currently executing, bringing the servo to a halt using the Profile Move deceleration register (#23).

### Profile Velocity Continuous (PVC)

PVC accelerates to the velocity given in the second register of the register pair using the acceleration from the first register of the register pair. During the move, either parameter may be updated, and the motion will be modified accordingly. The registers may be modified either by a Host controller using the serial interface or by an internal program if Multi-Task (EMT) is enabled. If Multi-Task is not enabled, the program execution will hold on the PVC command. See PVC in the Command Reference for more details.

Note, since there is no position parameter, the servo is able to maintain the desired velocity indefinitely.



## ERL - Error Limits and Drag Mode Operation with MAV and PMV

This exercise demonstrates the effects Drag Mode has on motion from the Move Absolute Velocity Based (MAV) and Profile Move (PMV) commands. Drag Mode adjusts the target position to keep it within a certain number of counts of the actual position. The number of counts is determined by the Moving Error Limit parameter of the ERL command. For example, if the Moving Error Limit is 500 and the servo jams at position 2000 while moving in the positive direction, the target position will not exceed 2500, regardless of what the original target was. The entire motion profile of the MAV command is pre-calculated by the device. It only moves the calculated time of the motion profile so the device will stop short of the target position. In contrast, PMV will not stop short because it recalculates the motion every servo cycle.

**Note:** This exercise requires the user to manually stop the motion of the shaft. A small flywheel attached to the shaft makes this task easier. The torque limits must be set low, so it will not be difficult to stop the shaft. Do not use higher torque motors (34 frame, for example), as even at a 30% setting the motor torque can exceed finger/wrist strength! Leather work gloves are suggested, as is the avoidance of loose clothing that could catch in the system! Please use caution.

1. Power up the device and start QuickControl. Start polling the motor and verify that the system is operating properly.
2. Select **File** → **Open**. Navigate to "...\\QCI Examples\\Profile Moves\\" and select the file, **Absolute and Profile Move with Drag Mode.qcp**.
3. Notice the error limits set with the ERL command in line 3. Also note that Drag Mode is enabled with this command.
4. Edit the TQL command in Line 7. Set the Closed Loop Moving and Open Loop Moving torque limits to 30%. Press **OK** when done.
5. Select **Tools** → **Register Watch**. Press **Add Register**; select **Target Position[0]** and **Position** data format. Press **OK** when done.
6. Repeat Step 5 but add **Actual Position[1]** and **User | Profile Move Position[20]**. Choose **Position** for the data format for both.
7. Edit the Absolute Location data of Line 10 (MAV) to read +10,000 counts. Press **OK** when done.
8. Edit the Data value in Line 14 (WRP) to read -10,000 counts. Press **OK** when done.

Some examples of Drag Mode are:

- A SilverLode servo can be configured to emulate a mechanical slip clutch. For this application, the ERL command is used to set the error limits and enable Drag Mode, and the Control Constants (CTC) command is used to tune the response. When configured for this operation, the servo will resist any external force on the shaft with a tunable amount of torque and will not fly back to its original position when the external force is removed.
- If the shaft were jammed, the actual position would not change since the shaft would be stuck. Drag Mode would freeze the target position until the jam was cleared, preventing the target position from running away. The target position would freeze a number of counts away from the actual position equal to the holding error limit.
- If the Trajectory Generator were commanding a move that was too fast for the motor to physically keep up with (due to excessive load or insufficient torque), Drag Mode would limit the target position runaway to a number of counts equal to the moving error limit.
- If the device were holding a position and the shaft was moved outside of the holding error limit, Drag Mode would "drag" the holding target position along as the shaft was moved, preventing a rapid move back to the original position when the shaft is released. A SilverLode servo would only move the shaft back a number of counts equal to the holding error limit.

## Anti-Hunt™ Feature

Digital servo systems share a common characteristic called “dithering” or “hunting” when holding a position. This occurs as any error causes the torque to build up until the motor overcomes friction and stiction and the motor moves to correct; often the motor may overshoot by a small distance and the process repeats. Dithering typically occurs when the holding position is near the count transition point of the digital feedback device (i.e. optical encoder or resolver), causing the shaft to oscillate between counts. For some applications, dither is desirable and may even be intentionally created. In other applications the hysteresis required for the Anti-Hunt algorithms is not acceptable. For these applications, disable the Anti-Hunt. In other applications, however, any dither is unacceptable and in severe cases, can cause significant oscillations in the whole system structure or audible noise when no motion is present. These applications benefit from Anti-Hunt.

SilverLode servos have a unique operation called Anti-Hunt that is designed to eliminate dither by switching from closed loop control to open loop control. When using open loop control, the device ignores any error feedback so small errors in position do not result in constant corrections. These constant small corrections are what cause the servo dither during closed loop operation. In addition to eliminating dither, the device can use Anti-Hunt to help smooth out some kinds of motion profiles by using open loop control during some parts of the move. This section explains the operation of Anti-Hunt, describes how to use it, and covers the commands used to set it up.

### Using Anti-Hunt™

The servo can be set to enter Anti-Hunt only when holding a position, or it can be set to enter Anti-Hunt any time position error is low enough. When using open loop control to hold position, the device ignores small position errors and lets the shaft sit at the position it held when it stopped. When using open loop control during a move, the servo ignores small errors and commutates the motor based only on the motion profile from the Trajectory Generator. Both settings for Anti-Hunt can make the servo operate more smoothly, but they also limit the final position accuracy.

Anti-Hunt is mainly used in two situations:

- 1) Holding position firm and steady without servo dithering.
- 2) A particular operation requires using open loop control for part of a move or hold.

If the accuracy of final position in an application requires strict compliance, Anti-Hunt may need to be turned off because it creates a feedback deadband around the final position. When the shaft is anywhere inside this deadband, the device ignores position feedback and it will not correct the final position error. In general, open loop control should not be used during motion unless there is a specific design reason (e.g. it eliminates dither in slow, high-inertia moves).

### Anti-Hunt™ Commands

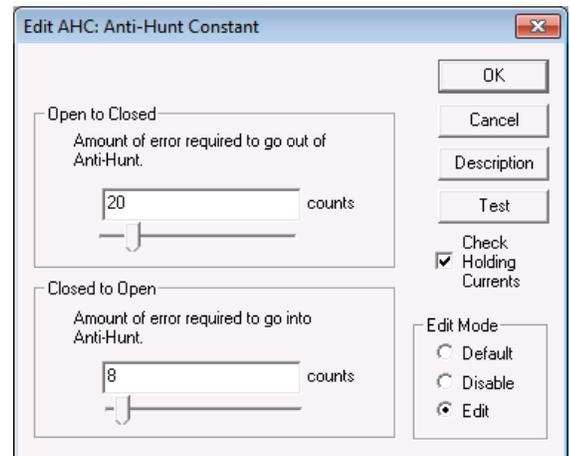
Five different commands are important to Anti-Hunt operation: Anti-Hunt Constants (AHC), Anti-Hunt Delay (AHD), Anti-Hunt Mode (AHM), Torque Limits (TQL), and Error Limits (ERL). The AHC and AHD commands set the conditions under which the device enters and exits Anti-Hunt, while the AHM command sets the type of anti-hunt operation the device uses. The AHC, AHD, and AHM commands are covered below.

### *Anti-Hunt Constants (AHC) Command*

The AHC command sets the conditions for the device to enter and exit Anti-Hunt. The AHC command uses three parameters: Closed to Open error, Open to Closed error, and the Check Holding Currents flag.

The Closed to Open error parameter sets the point at which the servo can enter Anti-Hunt and transition to open loop control. A SilverLode servo can enter Anti-Hunt once position error is equal to or less than this value. Position error is the difference between the target shaft position and the actual shaft position, measured in encoder counts.

The Open to Closed error parameter establishes the point at which the servo exits Anti-Hunt and returns to closed loop control. If position error ever equals or exceeds this value when the servo is in Anti-Hunt, the device immediately exits Anti-Hunt, switching to closed loop control to correct the position error. Setting the Closed to Open and the Open to Closed parameters to zero disables Anti-Hunt.



When the **Check Holding Currents** box is selected, as shown in the QuickControl screenshot, the servo checks another condition besides position error before entering Anti-Hunt. The servo compares the actual torque required by closed loop control (as indicated by the current flow through the motor windings) to the open loop torque limit. If the closed loop torque required exceeds the open loop torque limit, the servo will not enter Anti-Hunt. The servo does not compare the closed loop torque used to the open loop torque limit if the Check Holding Currents condition is disabled.

The **Use Default for Device** checkbox is checked by default. This causes QuickControl to set AHC at download time dependent on the device's encoder CPR.

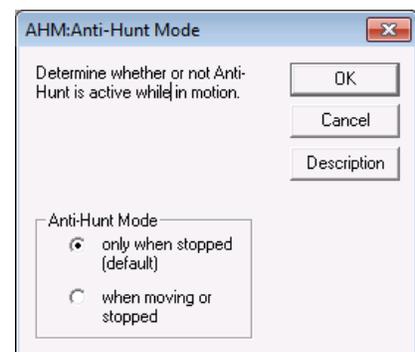
See the Command Reference for more details.

### *Anti-Hunt Delay (AHD) Command*

The AHD command sets the time delay the device uses before entering Anti-Hunt. This delay is useful for allowing a system to settle (stop ringing) before switching to open loop control, since position error and torque must remain within the limits set with the AHC command for the duration of the delay.

### *Anti-Hunt Mode (AHM) Command*

The AHM command controls the operation of Anti-Hunt. Anti-Hunt operates in one of two states, as shown in the QuickControl screenshot. In its default state, the servo will only enter Anti-Hunt when the servo is holding a position (meaning that the Trajectory Generator is inactive). If this default setting is changed, the servo can enter Anti-Hunt when moving or when holding position. This means that the servo can switch to open loop control whenever the position error is low enough, so an entire move could be executed using open loop control.



## *Error Limits (ERL) and Torque Limits (TQL) Commands*

These commands are covered in their own sections, but are important for Anti-Hunt. When the target velocity reaches zero, the ERL delay timer starts counting down. Until the timer expires, the control loop limits torque according to the “Moving” limits, Closed Loop Moving, if in closed loop, and Open Loop Moving if in open loop (including at position in Anti-Hunt. After the timer counts down to zero, the “Holding” limits are used – Closed Loop Holding or Open Loop Holding, as set by the TQL command. If the optional closed loop torque condition has been enabled with the AHC command (the Check Holding Currents box in QuickControl), the servo will compare the closed loop torque being used to the relevant open loop torque limit.

## **Anti-Hunt Operation**

Anti-Hunt essentially sets up a open-loop deadband around the target position. Inside this deadband, the servo uses open loop control and therefore ignores error feedback. The width of the deadband is defined by the Anti-Hunt Constants (AHC) command. When the position error becomes low enough to enter the deadband, the device performs some other checks (explained below) and then enters Anti-Hunt. When the position error becomes too large, the device immediately switches to closed loop control and exits Anti-Hunt. The Anti-Hunt deadband should be a different width depending on whether the device is entering Anti-Hunt or exiting it. The exit deadband is normally larger than the entering one.

The servo checks several conditions before entering Anti-Hunt. The conditions checked before entering Anti-Hunt are:

- **Position error.** The servo compares position error to the “Closed to Open” value set with the AHC command. If the absolute value of the position error is less than or equal to the AHC value, this condition is met and the servo can enter Anti-Hunt.
- **Anti-Hunt delay timer.** Once the position error is low enough, the servo starts counting down the Anti-Hunt delay timer, set by the Anti-Hunt Delay (AHD) command. The servo does not enter Anti-Hunt until the timer expires. This allows the servo to settle in before going to open loop. The timer restarts if position error exceeds the closed to open limit set with the AHC command while still in closed loop mode. If the closed loop torque condition (see below) is enabled, the transition to closed loop can not happen until the closed loop torque being used drops below the open loop torque limit.
- **Moving status (optional).** When the Anti-Hunt Mode (AHM) command is set to its default value, the servo only enters Anti-Hunt after the end of a move when it is holding a position. The end of a move occurs when the Trajectory Generator goes inactive. In this default setting, the servo uses closed loop control throughout the move. When the AHM command is set to the non-default value, the servo can enter Anti-Hunt at any time, moving or stopped. This allows the servo to use open loop control during a move if the other Anti-Hunt conditions are met.  
**Closed loop torque (optional).** If the AHC command is set to its default state, the servo compares the torque being used during closed loop operation with the open loop torque limit (the open loop moving limit during a move, the open loop holding limit while holding position). If the closed loop torque exceeds the open loop torque limit, the device will not enter Anti-Hunt. This extra condition can be disabled with the AHC command. This would be used to implement conventional dead band by setting the open loop torque

The servo checks only one condition to exit Anti-Hunt: position error. As soon as position error exceeds the Open to Closed value set with the AHC command, the device switches to closed loop control in order to correct the error. It then operates in closed loop control until conditions to enter Anti-Hunt are met again.

# Multi-Tasking

A SilverLode servo can execute programs with multi-tasking either enabled or disabled. With multi-tasking disabled (default), a program executes one command at a time. If a command starts a move that takes twenty seconds to complete, the program will wait for twenty seconds before executing the next command. With multi-tasking enabled, the move command will execute normally, but the program will continue to the next command while the motor is still in motion. Please See Appendix 1, exercise 3.2 for examples of Multitasking and a step by step sample program to get used to using this feature.

## Multi-Tasking Operation Rules

The purpose of multi-tasking is to allow the device to continue to execute commands while a motion command is running. The ideal way for this to happen would be for the Trajectory Generator to go active as soon as a motion command is issued and for the next command in the program buffer to be executed on the next servo cycle. In some cases, this is exactly what happens, but some motion commands still stop program flow right after they are issued, just like a delay or wait command. Different types of motion commands also work differently when multi-tasking is enabled. Please see Appendix I, exercise 3.2 for a list of special cases and multi-tasking exceptions.

## Multi-Tasking Operation

Multi-tasking enables the servo to continue executing a program while a motion is in progress. With multi-tasking disabled, when the servo executes a motion command the program pauses execution until the move is complete (i.e. the Trajectory Generator is inactive) or until a stop condition is met. With multi-tasking enabled, the program continues to run during the move, allowing much greater versatility in the program. The Enable Multi-Tasking (EMT) command enables multi-tasking and the Disable Multi-Tasking (DMT) command disables it. Using multi-tasking correctly requires a firm understanding of how the servo controls program timing, the system tasks that the servo executes in the background along with program commands, and the ways the servo responds to different motion commands when multi-tasking is enabled.



### Multi-Tasking for Advanced I/O Control

This example shows multi-tasking use with inputs and outputs during motion. In normal program flow, each command executes in order, and the next command is not allowed to start until the previous command has finished. The EMT - enable Multi Task command allows the commands following a move command to be executed while the motion is still happening. Motion override commands can be used to take over the current motion, while other motion commands will stall the execution until the prior motion has completed. This program executes one of two Register Move Relative, Velocity Based (RRV) commands, depending on the state of I/O #1 and #5. The values stored in registers 25 and 26 are used as parameters for the RRV move commands. The loop is comprised of Jump (JMP) commands in an infinite loop that execute continuously until one of the inputs is triggered low. Once an input is LOW, a move is executed. If I/O #1 is LOW, the I/O #2 will flash (turn ON/OFF) during the motion. If I/O #5 is LOW, the second RRV move will run and the I/O #2 will of HIGH.

1. Power up the servo and start QuickControl. Start polling the servo and verify that it is operating properly. Ensure that I/O #1 is HIGH.
2. Select File → Open. Navigate to “...\QCI Examples\Multi-Tasking\” and select the file, Using Multi-Tasking for Advanced IO Control.qcp.
3. Press the “Download/Restart” button to download and run the program.
4. The servo run the program and wait in the “Forever” loop for an input. Toggle I/O #1 or #5 LOW then HIGH to select the “Move1” or “Move2” routines. If #1 is chosen, the output will flash three times at the beginning of the move.

## Multi-tasking Exceptions and Interactions

- **Delays with Motion Commands.** The time and velocity based motion commands (MRV, MRT, MAV, MAT, RRV, RRT, RAV, RAT, XRV, XRT, XAV, and XAT) are all pre-calculated. This means that the device uses the motion parameters specified with each command and calculates the entire trajectory for the move before sending the calculated trajectory to the Trajectory Generator and actually starting the move. During this calculation period, the device does not execute the next command in the program buffer, even if multi-tasking is enabled. For the SilverNugget, this calculation period is up to two milliseconds for the time-based moves and up to four milliseconds for velocity-based moves. The Pre-Calculate move commands (PCM and PCG) allow some control over this delay period and can be useful for some applications. The Command Reference has more information on these two commands. The times for SilverDust and SilverSterling are significantly shorter.
- **Velocity and Time Based Motion Commands.** When the servo executes a time or velocity based motion command, it checks to see if the Trajectory Generator is busy (active). If another move is still running, the program will pause until the prior motion has completed. This effectively stalls multi-tasking. Once the first move finishes, multi-tasking begins working as expected again. This is one of the most common errors encountered when using multi-tasking. Some other commands can override the first motion command and unblock the program. The velocity and time based motion commands can be overridden by the VMP, HSM, HLT, STP, PMO, or PMX commands. The override commands read the velocity and target position of the motion in process and use those as the starting point for their operation.
- **Velocity Mode Motion Commands.** The velocity mode commands (VMP and VMI) do not pre-calculate their moves, so program flow is not disrupted. They also do not check on the busy status of the Trajectory Generator, so they will override any other move that is still executing. Setting the desired velocity to zero and issuing the VMP or VMI command is the best way to do a controlled stop on a move, because these two commands override the other move commands and because the deceleration can be specified. A VMP or VMI command can be overridden by another VMP or VMI command, a PMO command, or by an HSM, STP, or HLT command.

Note: the VMP command may be placed into a program, while the VMI command is only an immediate command sent via the serial port at execution time.

- **Profile Move Commands.** The profile move commands allow the device to follow a pre-defined motion profile. Like VMP and VMI, the Profile Move (PMV) and Profile Move Continuous (PMC) commands do not pre-calculate their moves. They can be overridden by other PMV and PMC commands, by VMP and VMI commands, or by the PMO, PMX, HSM, STP, and HLT commands. Moves using profile move commands can be dynamically changed by modifying the motion parameters when multi-tasking is enabled or by writing to the registers via serial or CANopen. Interpolated moves using the IMS command work like profile moves with respect to multi-tasking.
- **Step and Direction Commands.** The step and direction commands allow the servo to respond to direct motion commands from a host in a manner similar to a simple stepper motor (although the device can still use closed loop control, unlike any stepper). Like velocity mode and profile move commands, the Registered Step and Direction (RSD) and Scaled Step and Direction (SSD) commands do not pre-calculate their moves. They can be overridden by other RSD and SSD commands, by VMP and VMI commands, or by the PMO, HSM, STP, and HLT commands.
- **Input Mode Commands.** The Velocity Input Mode (VIM), Torque Input Mode (TIM) and Position Input Mode (PIM) commands allow the device to respond directly to the values in registers 12

through 18. Like the last three motion commands, they do not pre-calculate their moves. The input mode commands can be overridden by VMP and VMI commands, or by the PMO, HSM, STP, and HLT commands. Input mode moves can also be changed on the fly by changing the values in register 12 to 18.

- **Hard Stop Move (HSM) Command.** The HSM command immediately stops a move started by any motion command.
- **Stop (STP) Command.** The STP command immediately exits the current program and stops any active move using the specified deceleration (see STP in command reference for details).
- **Halt (HLT) Command.** The HLT command immediately stops any motion, sets the halt bit in the Internal Status Word, and calls the Kill Motor Recovery program. When multi-tasking is enabled, the motor drivers will remain active during the Kill Motor routine if the Kill Enable Driver (KED) command has been issued. With multi-tasking disabled, the motor drivers are always disabled during a Kill Motor routine, regardless of whether or not a KED command was issued. See Technical Document QCI-TD052 Shutdown And Recovery on our website.
- **End Program (END) Command.** If multi-tasking is enabled and an END command is issued, the device will finish the current move if a motion command is active, and then end the program.

### Multi-Tasking Examples

The following examples illustrate how multi-tasking works with the programs.

#### Example 1: I/O Control During a Move

This program starts a 10-revolution MRV move, waits for a 1000 msec delay, then toggles I/O bit #1 for 50 msec while the move is still in progress. The WBS command in line 7 causes the program to wait on that command until the move command is complete. The move completes before bit #2 is cleared in line 8. This program sends an output signal when the move is finished, but also allows other commands to be executed before the WBS command holds the program while the move finishes. Several non-motion commands could be inserted between lines 2 and 7 that would all execute during the MRV move.

Line# Oper	Label	Command
1:EMT		Enable Multi-Tasking
2:MRV		Move 10 REV @ acc=1 Rps/s vel=5 Rps
3:DLY		Delay for 1000 mSec
4:COB		Clear "I/O #1"
5:DLY		Delay for 50 mSec
6:SOB		Set "I/O #1"
7:WBS		Wait On Bit State Until "Trajectory Active" is LOW/FALSE
8:COB		Clear "I/O #2"
9:END		End Program

#### Example 2: Multi-Tasking Motion Command Buffering

Multi-tasking will have no effect in this program. Since time and velocity based motion commands are buffered, when the second MRV command is encountered, the program stalls until the first MRV command is finished executing before starting the second MRV and continuing with the program.

Line# Oper	Label	Command
1:EMT		Enable Multi-Tasking
2:MRV		Move 10 revs @ acc=2 rps/s vel=5 rps
3:MRV		Move 10 revs @ acc=2 rps/s vel=5 rps
4:END		End Program

### Example 3: Multi-Tasking Motion Command Transitioning

In this next program, the servo begins the MRV move and then delays for 3000 msec before continuing with the rest of the program. After the delay time, the device executes the VMP command in line 4. As explained previously, the VMP command overrides all other motion commands, so this command prematurely ends the move from line 2 and accelerates the motor to turn at 20 revolutions per second.

Line# Oper	Label	Command
1:EMT		Enable Multi-Tasking
2:MRV		Move 10 revs @ acc=2 rps/s vel=5 rps
3:DLY		Delay for 3000 mSec
4:VMP		Velocity Mode: acc=4 rps/s, vel=20 rps
5:END		End Program

## Multi-Thread

Multi-Thread allows the user to simultaneously run two programs. Thread 1 is the primary program, which runs after a power on or reset condition, and is capable of all commands except T1F – Thread 1 force (Causes a LRP command to be executed within Thread 1). Thread 2 is invoked from Thread 1; the invoking command specifies how much of the Program Buffer to allocate to Thread 2, and where the new Thread 2 program is stored in the Non-Volatile memory. The allocated memory + 1 location are subtracted from the top of Thread 1 Program Buffer space. Thread 2 executes with the start of its Program Buffer at 0, just like Thread 1. Thread 2 is not able to execute motion commands to prevent the resulting confusion, but it is able to do IO and calculations.

While the second thread is running, Thread 1 gets executed every 240  $\mu$ s and Thread 2 gets executed every 240  $\mu$ s, on alternating 120  $\mu$ s intervals.

Thread 2 is able to read and modify registers, IO, and to use CAN functions. It cannot execute any command which affects the Trajectory Generator (i.e. motion), nor is it able to invoke a new Thread 2 or change its buffer allocation.

To simplify programming, each thread has its own private register 10 (Accumulator) as this register has special uses; Thread 2's register 10 appears as register 248 to Thread 1 and the outside world. Each thread also has its own Zero/Negative/Positive condition flag storage.

A status bit (BIT 11) in IS2 is provided to monitor Thread 2, with a high level indicating that Thread 2 is active. The Kill Recovery Extended (KMX) command may be used to cause Thread 1 to automatically respond to the loss of Thread 2. (Such as shutting down Thread 1 if Thread 2 is performing a safeguard function and dies.)

Thread 2 Kill Conditions (T2K) command specifies which conditions kill Thread 2 and which it will survive. These include Kill Motor (which affects Thread 1), Over Voltage, Under Voltage Driver, Under Voltage Processor, Halt command, and Stop command. The default condition is to halt Thread 2 on any of these conditions unless the user specifically configures Thread 2 to ignore them.

Just as Thread 1 can startup a Thread 2, Thread 2 can force a new program upon Thread 1, acting much like a software invoked Kill Motor Recovery routine. Thread 1 is also capable of stopping Thread 2; Thread 2 also ceases if it encounters an END command. The time slice and the command buffer space are returned to Thread 1 if Thread 2 execution ceases.

If a Thread 1 only command tries to execute in a Thread 2 program, Thread 2 will shut down and produce a Command Error Code #17 (see Appendix C). All other Command Errors Codes for Thread 2 errors are their respective Thread 1 Command Error Code plus 64 (0x40).

Note: CIS (Clear Internal Status) clears the common register, so be careful if this command is used in both threads.

The Delay counter, used in DLY, DLT and WDL, is common to both threads, so use it in only one thread or the other. For timing in the other thread, use the Count Up Timer (reg 244) or the Count Down Timer (reg 245).

### Using QuickControl To Launch Thread 2

Thread 1 and Thread 2 programs are written normally and exist as programs within the same program file (same QCP). The following is an example (QCI Examples\Multi-Thread\Multi-Thread.qcp) contains a Thread 1 and Thread 2 program.

Line# Oper	Label	Command
1:REM		
2:T2S		Start Thread 2 with Program = "Thread 2"
3:REM		Move forward and back
4:MAT	LOOP	Move to 4000 counts @ ramp time=50 mSec total time=200 mSec
5:MAT		Move to 0 counts @ ramp time=50 mSec total time=200 mSec
6:JMP		Jump to "LOOP"

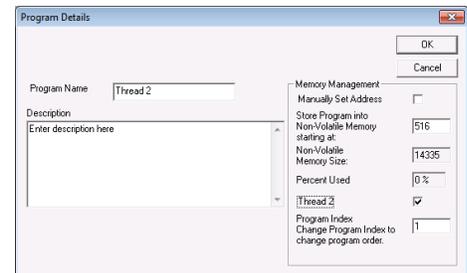
Thread 2

Line# Oper	Label	Command
1:REM		Thread 2 Flash Output
2:REM	LOOP	Main Loop
3:COB		Clear "I/O #102"
4:DLY		Delay for 500 mSec
5:SOB		Set "I/O #102"
6:DLY		Delay for 500 mSec
7:JMP		Jump to "LOOP"
8:REM		End
9:REM		

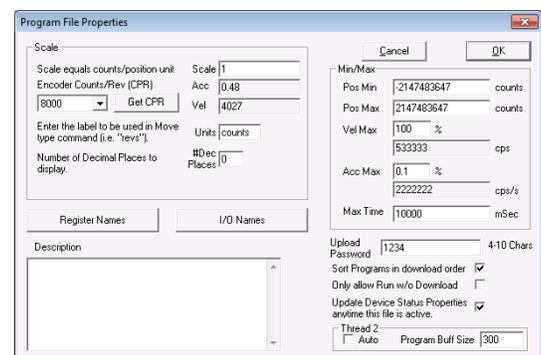
Thread 1

Thread 1 Program is the main program (program 0) and runs at power up. The T2S command allocates Thread 2 Program Buffer space and launches the program "Thread 2". In the example above, Thread 1 Program then executes a loop that moves the servo from 4000 counts to 0. Thread 2 Program simultaneously executes a loop the "flashes" output #101.

In order to allow a program to be run in thread 2, it is necessary to tell QuickControl that it is a Thread 2 program. This is done by going to Program Details from the Programs menu and checking the Thread 2 checkbox.



The amount of Program Buffer allocated to Thread 2 is calculated automatically by QuickControl or can be set manually by the user. If calculated by QuickControl, it is set to the largest Thread 2 program in the file. To manually set the Program Buffer allocation, select Program File Properties from the File menu. In the Program File Properties dialog box, uncheck Thread 2 Auto and enter the amount of words to allocate to Thread 2. The example shown allocates 300 words of the Program Buffer to Thread 2.



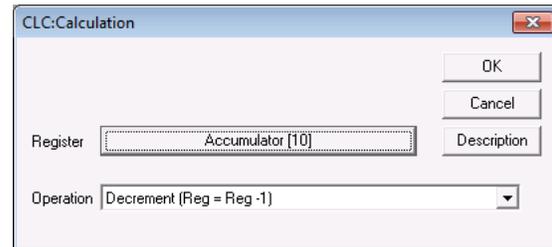
# Specialty Commands

## Calculation (CLC)

The CLC command provides basic math and logic functions.

## Calculation Two Word (CTW)

The Calculation Two Word (CTW) command is identical to CLC except that it requires an extra word of memory that makes it easier to create for a host (see Command Reference for details).



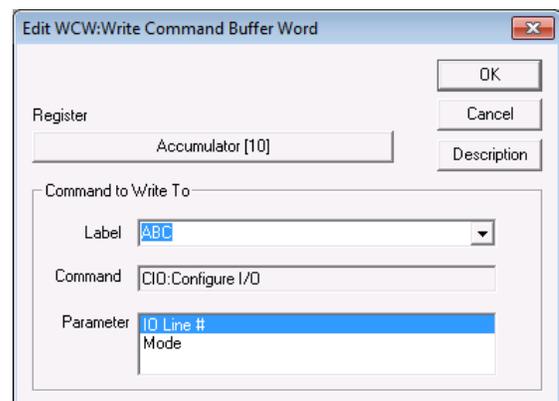
Many programs require operations to modify values stored in data registers, manipulate binary numbers, or aid in programs using loops. Three different kinds of operations are accessible with the CLC command. The first group of operations is basic math functions. These include add, subtract, multiply, divide, absolute value, increment, and decrement functions. The second group contains binary logic and number manipulation functions, which include the bitwise AND, OR, and XOR functions, as well as several bit shift functions. The third group contains data register and accumulator manipulation functions. These operations are used to move data to and from the accumulator register and to load, store, and manipulate data in the data registers. Operations of the CLC command are accessed from a pull-down list. See Command Reference for details.

## Calculation Extended (CLX) and Calculation Extended With Data (CLD)

Three parameter versions of the Calculation (CLC) command that allow for such things as adding two registers and storing the result in a third register. Available on SilverDust and SilverSterling.

## WCW and WCL Commands

The Write Command Word (WCW) and Write Command Word Long (WCL) commands are two of the most powerful commands available. The WCW and WCL commands are extremely powerful commands that should not be used until fully understood, but which can add a new degree of flexibility to programs or host applications. They allow a program to modify itself by overwriting data in the Program Buffer. They turn any command that requires a data parameter into a register-based command. A SilverLode servo does not do



extensive error checking on the data that is changed, potentially making these commands very dangerous. The use of the WCW and WCL commands should normally be limited to use within QuickControl. QuickControl checks for errors by keeping track of which line in the buffer will be modified and by querying for the parameter to be changed, rather than blindly changing data.

Both the WCW and the WCL commands work the same way. The difference between them is that the WCW command is used for 16-bit parameters and the WCL command is used for 32-bit numbers. When either of the commands is issued in a program, the device replaces part of the data in the Program

Buffer with the data in the register the WCW or WCL command specifies. With QuickControl, the Program Buffer data to be replaced is specified symbolically, rather than with an actual memory location, reducing the chance for error. The QuickControl screenshot shows the WCL dialog box used with QuickControl. The parameters for both commands are the same. QuickControl requires the register containing the value that will overwrite the old data, the label of the line containing the command that will be modified, and the command parameter that will actually be modified.

The WCW and WCL commands can turn any command into a register-based command. A simple command to do this with is the Velocity Mode, Program Type (VMP) command. The VMP command requires two parameters: velocity and acceleration. Normally, these parameters are entered with the VMP command and are fixed until a second VMP command is issued. However, if multi-tasking is enabled and a WCL is issued in the program, this can change. One WCL command can link the value in one 32-bit register to either the velocity or the acceleration parameter of the VMP command. This means that two WCL commands could link the values in two user registers (registers 30 and 31, for example) to the VMP parameters. Every time the VMP command is issued, it will use the values copied from the user register specified by the WCL commands. Data in the user registers can be changed by a program, by an external host, or by an analog input.



### Calculation Example

This exercise demonstrates the use of several different functions of the Calculation (CLC) command. It begins with a long move, calculates half the distance moved, and moves back that amount. It is strictly a demonstration of the CLC command. The SilverSterling and SilverDust also have the Calculation Extended (CLX) and Calculation with Data (CLD) commands which allow three parameter calculations and additional operations. These may make calculations easier to write and to understand.

1. Power up the device and start QuickControl. Start polling the motor and verify that the system is operating properly.
2. In QuickControl, select File > Open. Navigate to "...\\QCI Examples\\Data Register\\" and select the file, Register Moves by half with Calculation.qcp.
3. Select Tools > Register Watch to open the Register Watch tool. Press the Add Register button and select User (25). Select Position for the Data Format and press OK. Repeat for Accumulator (10), Actual Position (1), and User (30). Choose Position format for each register.  
The value in register 25 is how small the move position will be allowed to become before restarting. Raising this will make the cycle time shorter. Register 30 contains the beginning position for the move. The accumulator is used with most of the CLC operations in a similar manner to how the accumulator is used in an assembly program. In this exercise, the accumulator's value is not used directly by the motion command. The value in the actual position register should be  $User[30] - User[25]$  at the end of each move.
4. Press the "Download/Restart" button to download and run the program. The motor will begin its first move of 30,000 counts.
5. The calculation process is reasonably simple, using a few math operations to accomplish its task. Any data in the accumulator is first cleared, and then the actual position is subtracted from the accumulator. This results in a negative position value. After this, the binary value in the accumulator is bit-shifted right, which is the binary equivalent of dividing by two. This value is placed in the register used by the move. After this, the absolute value of the accumulator subtracted from the value in register 25. If the result of this calculation is negative, the new move distance will be less than the value in register 25, and the program will reset the position to 30,000 counts.

NOTE: Single Step or Trace through the program to see how each CLC command effects the registers in Register Watch.

NOTE: Using the commands CLX and CLD simplifies this example and should be used on SilverDust and SilverSterling servos.



## Dynamic Speed and Acceleration Adjust - Modifying "Fixed" Parameters

This exercise uses "multi-tasking" with Write Command Buffer Longword (WCL) commands to dynamically adjust the velocity and acceleration parameters of a Velocity Mode, Program (VMP) command. The parameter data is loaded into two registers directly by the user. This program consists primarily of a loop that contains two WCL commands and a VMP command. The WCL commands move data from register 25 and register 26 into the velocity and acceleration parameters of the VMP command, respectively. Using the Register watch tool, both parameters of the VMP command can be modified.

In a real world application, a host could issue Write Register, Immediate Type (WRI) commands via serial communications to change the parameter values. Additionally, the Calculation (CLC) command can be used to modify the register data. XRV is a register based version of VMP that directly uses registers rather than using the WCW or WCL commands.

1. Within QuickControl, select File > Open and navigate to '...\QCI Examples\Applications\' and select the file 'Dynamic speed & accel in VMP.qcp'
2. Press the "Download/Restart" button to download and execute the program. The motor will NOT start moving, since the default Velocity is 0 rps and the default Acceleration is 0 rps/s.
3. Open the Register Watch Tool (Tools > Register Watch). Press the "Add Register" button and select User (25), select Velocity for the Data Format and press OK. Repeat again for User (26) except select Acceleration for the Data Format. Both should have 0 values listed.
4. Click once in the data column of the User (25) entry. Enter a value of 5 rps for the velocity (be aware of the scaling being used) and press enter. Select the data field of User (26) and enter a value of 2 rps/s into the cell. Note: If no "units" are shown in the right column, double click on the units field of the Register Watch Tool and choose the appropriate type (Target Velocity (32 bit) for Reg. 25, rps & Acceleration for Reg. 26, rps/s).
5. Experiment with other values in these registers and note the effect on the motor operation. Try starting with a very slow Acceleration rate (0.5 rps) and a Velocity rate of 30 rps. After motion begins, increase the Velocity rate to 1 rps, then 5 rps, and then 10 rps... Notice the dynamic Acceleration change.

## Input Mode Commands

Three commands, Position Input Mode (PIM), Velocity Input Mode (VIM), and Torque Input Mode (TIM), are used to access three special operating modes. The Input Modes use seven data registers for processing position, velocity, and torque information. They allow the servo to use data from an analog input or an external host to directly control motion. For example, this is a good way to configure the servo to respond to a joystick or analog input. These powerful commands are beyond the scope of this document. See Application Note "QCI-AN047 Input Mode - Joystick".

## Shutdown and Recovery

Every servo cycle (120 microseconds), the SilverLode servo performs an error check based on the settings issued in the Kill Motor Conditions (KMC) (SilverDust Rev 06 or KMX command). If any of these kill conditions are met, the program specified by the Kill Motor Recovery (KMR) command is immediately loaded. The program specified by KMR can then perform any operation, a shutdown, recovery, or other technique. For details on this advanced topic, see Technical Document QCI-TD052 Shutdown and Recovery on our website.

## Advanced Topics

Please refer to the technical documents available on our website for further instruction and information on advanced topics, such as camming, torque control or interpolated motion control. Our Technical Support Staff are also available by phone or email during regular business hours.

# Chapter 4: Input Output Functions

## SMI and SIP interfaces

The SilverNugget servos have I/O voltages level from 0 to +5 Volts for both digital and analog operations. The SilverDust and SilverSterling servos have I/O voltages level from 0 to +3.3 Volts for both digital and analog operation, but are 5v tolerant. All digital I/O used as inputs are digitally filtered, with default filtering time typically set to 10ms in the initialization file. This filter time may be changed, as a group or individually, from 120uS to over 5 seconds.

The SilverNugget and SilverDust controllers provide an SMI input. This 15 pin high-density D-sub connector includes power supply, communications, a 5v/100mA output, and seven Input/Output (I/O) connections, four of which also support analog inputs. Encoder quadrature or step-and-direction inputs may also be applied to these inputs.

The SilverSterling provides an SIP input. This 15 pin high-density D-sub connector includes power supply, communications, a 5v/100mA output, and four Input/Output (I/O) connections, one of which also supports analog input. Three of the connections are dedicated to CAN.

The SilverDust IGB has the SMI interface plus sixteen additional expanded IO lines (IO101-10116). These expanded IO are optically isolated from the rest of the controller, and must be powered from 12v to 24v. The expanded outputs are NPN - that is, they are open collector outputs with a feedback comparator set to approximately 1.5v. An internal pull-up resistor is tied to the isolated 5v supply (derived from the isolated 24v supply). The input thresholds make these inputs compatible with 3.3 through 24v logic, while the outputs are rated to 250mA each, and include clamping to handle inductive loads.

The SilverDust IG8 has 8 expanded IO lines (IO101.-108). These IO function the same as those in the IGB, except that no external 24v supply is needed. The IG8 also has three additional lines, 201, 202, and 203. Setting 201 lights the user status LED, while clearing it turns it off. IO 202 and 203 are attached to the optional Ethernet converter's User IO 2 and 3, respectively. These IO may be configured as inputs to monitor the Ethernet IO or as outputs to drive the Ethernet IO. IO 201-203 do not include I/O filtering functions, as these inputs internal should be noise free.

The SilverDust IG8 also has an optional special Encoder/SSI port. This port may be used to input or output differential encoder signals, as well as being used to interface to Synchronous Serial Interface encoders. Additional signals have been provided to allow interface to SPI style devices for future expansion. This port is configured via the Synchronous Serial Interface (SSI) command. Differential Encoder Input also requires the Select External Encoder (SEE) command.

### I/O Functions

The seven I/O lines can be used for many different functions. The table below lists the I/O functions, their type, their description, and the I/O lines they use. I/O 101+ are only available on the SilverDust-IGB and IG8. IO5-7 are not available on the SIP connector, but are available from the internal 60 pin mezzanine connector.

Function	Type	Description	I/O Lines
General Digital Input and outputs	Digital Input	All I/O lines can be used as general-purpose digital inputs or general purpose outputs. The inputs can be used for a number of uses within a program, including loading new programs and controlling program flow.	1 – 7 101-116.
Motion Control	Digital Input	All I/O lines can be used as input stop conditions for motion commands.	1 – 7 101-116
Kill Motor	Digital Input	Three I/O lines can be used as Kill Motor Conditions, allowing for immediate shutdown based on input. Any IO can be used for SilverDust and SilverSterling via the Kill Motor Extended command (KMX)	1 – 3 SD05: 1-7, 101-116
Modulo Trigger	Digital Input	SilverNugget ONLY: I/O #1 can be used as a special digital input to trigger, enable, or disable the modulo output function.	1
General Analog Input	Analog Input	SilverNugget and SilverDust :Four I/O lines can receive analog signals. These signals can be used within programs. SilverSterling: IO4 can receive analog signals	4 - 7 4
Internal Encoder Output	High-Speed Output	SilverNugget, SilverDust IG8: Three I/O lines can send the raw signal from the internal encoder as an output.(Note: SilverDust-IG/IGB have dedicated lines, SilverSterling has these signals on the 60 pin mezzanine.)	1 – 3
Inter/External Encoder Buffer	High Speed Output	SilverDust IG8: IO 4,5 and 6 may be driven from either the internal or external encoder signals. This function is configured via the SEE, SSI and EMN commands. .	4-6
Modulo Output	High-Speed Output	SilverNugget:Two I/O lines can send a scaled signal from the internal encoder as an output while I/O #1 can toggle the function on and off.	1, 6 - 7
External Encoder Input	High-Speed Input	SilverNugget and SilverDust: Three I/O lines can receive a position signal from an external encoder (or another feedback device like a resolver). This signal can be used for either dual loop closed loop or for an input position command for special applications including camming and electronic gearing.	2, 3, 4 – 6**
PWM output	Digital Output	SilverDust, SilverSterling: allows I/O #2 to function as a 25kHz PWM output, registered controlled, with the PWM width adjusted every 120uS as a function of register values.	2
Done output	Digital Output	I/O #1 can be configured as a “Done” output, indicating that the error is within the configured limit and the sequence has completed. See Enable Done High (EDH) and Enable Done Low (EDL) for more details.	1
Position Compare	Digital Output	I/O #1 can be configured as a Position Compare output, updated every 120 uS. This may be configured for a single compare or for cyclic operation. See Position Compare (PCP).	1

## Curent Sinking and Sourcing capability

Sink or Source Limits			
	SilverNugget	SilverDust	SilverSterling
Output 1	5 mA	4 mA	2 mA
Output 2	5 mA	2 mA	2 mA
Output 3	5 mA	2 mA	2 mA
Output 4	5 mA	4 mA	2 mA
Output 5	5 mA	4 mA	2 mA (internal)
Output 6	5 mA	8 mA	2 mA (internal)
Output 7	5 mA	4 mA	2 mA (internal)

### Analog Inputs

I/O lines 4, 5, 6, and 7 (SilverSterling: IO4 4) can be used as analog inputs. These lines have an effective internal impedance of approximately 200 k $\Omega$  connected to the internal +3.3 (SilverNugget 5v) VDC power supply, giving a slight bias on input. This resistance should be considered in circuit designs that are passive or that have high impedance. I/O lines used as analog inputs must be driven from a low impedance source (10  $\Omega$  or less for SilverNugget and 1K $\Omega$  or less for SilverDust and SilverSterling) or with capacitance across the input. A minimum 0.01 uF capacitor can be used at the input to provide the low impedance source. The internal analog to digital converter (ADC) provides 10-bit resolution (Silver Sterling has 12-bit resolution) of the input signal.

The SilverLode servo implements a 5 msec filter on all analog channels to reduce the effects of noise & transients. This means that analog signals are averaged over 5 msec before being used. The filtered signal is updated every servo cycle (120 usec). (Note: some commands modify the filter constants.)

## Using Digital Inputs

Digital inputs and digital outputs are the most basic uses for the I/O. A digital input could be a switch that closes and opens, sending a signal to the servo, while a digital output might be a solid-state relay or a light connected to a I/O line. Digital inputs and outputs are used by the servo for several purposes. Digital inputs may be used for program flow purposes, for motion control (stop on input) purposes, or for Kill Motor triggers. Digital outputs may be used in user programs for signaling external devices like PLCs or controlling external devices like relays. This section covers the uses of digital inputs and outputs and the commands used with them.

### General Digital Inputs

Four uses of the digital inputs are: general digital inputs, motion control inputs, modulo trigger input, and Kill Motor inputs. By using an I/O line as a digital input, a program can react to an on/off signal from an external source by using a command that controls program flow based on I/O status. There are several commands a program can use to do this. The “wait on bit” commands (WBS and WBE) can stop program execution until a digital input signal changes, while “jump” commands (JOI, JOR, etc.) allow a program to jump to another command within the program based on one or more inputs.

For example, if an application required the servo to start a move if a switch were thrown, the Wait on Bit State (WBS) command could be used. This command can be tied to any I/O line configured as an

input and can be set to wait until the input goes high or goes low. If the command were set to wait until I/O #1 went high, for example, the program would pause at the WBS command for as long as I/O #1 stayed low. As soon as I/O #1 went high, the program would continue. If a motion command immediately followed the WBS command, then the start of the move would be tied to the state of I/O #1.

A looping structure with a jump command could be used for the same application if multitasking were enabled. A loop could be set up using the Jump (JMP) command to repeat the loop and a Jump On Input (JOI) command could be inserted into the loop and tied to the state of I/O #1. If I/O #1 were low, the loop would continue running repeatedly. When I/O #1 went high, the JOI command would cause the program to jump out of the loop. If the JOI command pointed to a motion command, the start of the move would be tied to the state of I/O #1, just like in the previous example. The example programs that come with QuickControl illustrate both of these program flow techniques.

### **Kill Motor on Input**

Digital inputs using I/O lines 1, 2, or 3 can be used as Kill Motor Conditions to immediately stop the motor and end any move. Kill Motor Conditions are covered in detail in Technical Document QCI-TD052 Shutdown And Recovery on our website. As with other digital inputs, the Kill Motor routine can be set up to trigger based on a high or a low state of an I/O line. If an I/O line is to be used as a Kill Motor input, extreme care must be taken to ensure that the I/O line is not used for another function. If an I/O conflict occurred on that line, the Kill Motor routine might not start when it was intended to. Kill Motor Extended (SilverDust, SilverSterling) allow the use of additional IO and conditions to trigger a Kill Motor Recovery.

### **Digital Input Filter (DIF) Command**

The DIF command sets up a filter time for any of the I/O lines used as a digital input. The filter time affects how long a digital input state must be held for the SilverLode servo to see the given state. This filter is useful for noisy systems or for de-bouncing switches because it causes the servo to wait for the specified number of servo cycles (120 usec) before recognizing a change in the state of the input.

## **Using Digital Outputs**

The digital outputs are used as signaling or control outputs. They can be used to indicate the internal status of the servo to an external device like a PLC, or used to control another device like a relay or another the device. The main commands used for this function are Configure I/O (CIO/CII), Set Output Bit (SOB), and Clear Output Bit (COB).

For example, a program could include branching logic that would jump to one of two sections of code based on an input. One section of code would start a rapid move while the other section of code would start a slower move. The section of code for the rapid move could also set I/O #2 high. That output could connect to an input on a PLC. The section of code for the slow move could set I/O #3 high and that output could trigger another input to the PLC. With this setup, the PLC could monitor a critical state of the servo.

With a similar setup, a SilverLode servo with the two move speeds could use its two digital outputs to interface with two digital inputs on a second servo. The second servo could be programmed to respond

in one way if the first servo were moving at the fast speed and in another way if the first servo were moving at the slow speed. Interlocked programs like this can be very useful on multi-axis machines.

## Using Analog Inputs

Analog inputs are another way to use the I/O lines. An analog input could be a potentiometer, a Hall-effect joystick, a temperature sensor, or a pressure transducer. A +5 V power supply is available from one of the pins that can provide up to 100 mA for sensors and other peripherals. The analog inputs can be used for traditional PLC tasks like event triggering or data monitoring, or they can be used for direct motion control (i.e. Joystick Control).

This topic is beyond the scope of this document. See Application Notes “QCI-AN023 Analog Inputs” and “QCI-AN047 InputModeJoystick” for more information.

## Using Encoder Signals with Digital I/O

In addition to the other functions covered in this chapter, the I/O lines can be used for high-speed I/O functions. This section describes the types of signals used by the high-speed I/O functions, their use, and commands used to configure them.

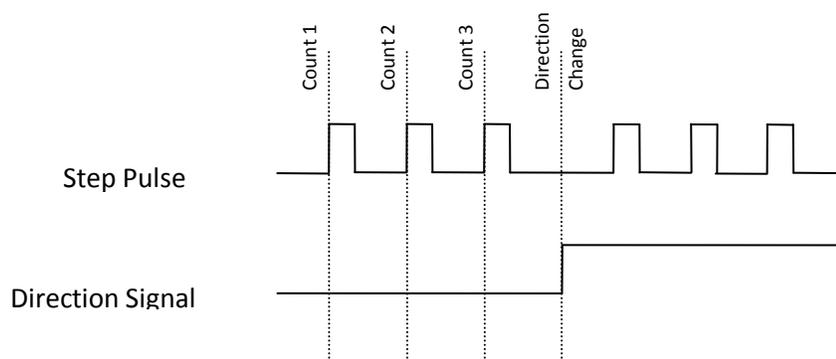
The external encoder inputs, either Quadrature or Step and Direction, are decoded to drive a hardware counter which is sampled every 120usec.

### Encoder Signal Types

SilverNugget and SilverDust: The high-speed digital I/O functions use several types of signal formats: step and direction, A and B quadrature, and, for the SilverNugget, step up/step down.

### Step and Direction Signals

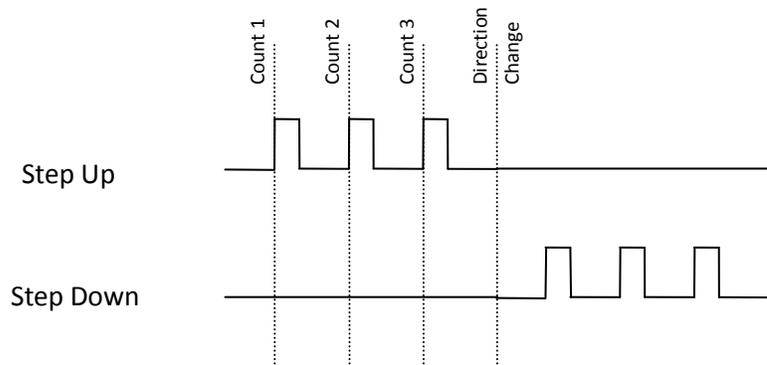
A step and direction signal consists of two parts: a step signal and a direction signal. As the figure shows, every rising edge of the step signal equals one count. The direction signal is high for one direction, and low for the other.



**Step and Direction**

### Step Up/Step Down Signals (SilverNugget Only)

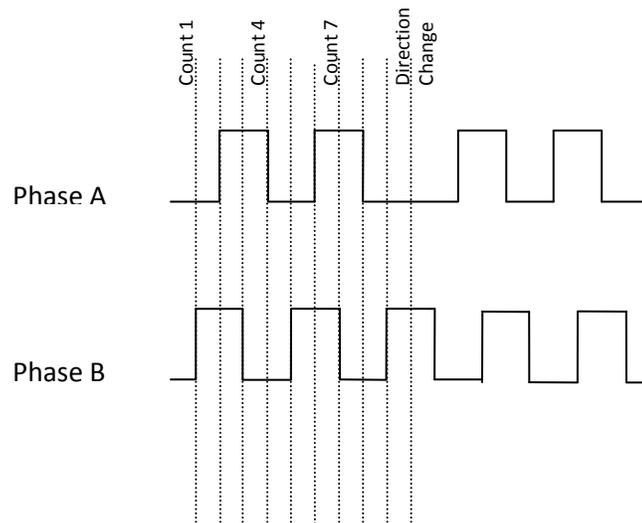
A step up/step down function consists of two step signals. One step signal corresponds to one count of motion in one direction, while the other step signal corresponds to one count in the other direction.



### Step Up/Step Down

### A and B Quadrature Signals

The A and B quadrature format consists of two step-like signals that are 90° out of phase with each other. Every rising or falling edge of each signal corresponds to a count. Direction is determined by which phase is leading and which is lagging.



### A & B Quadrature

The preferred encoder input and output signal is A/B quadrature. The alternative formats of step-up/step-down and step and direction transmit one pulse per encoder count and become subject to the bandwidth limit more rapidly. For example: during a 1000 count per second move, the step formats require 1000 pulses per second on a single line. A/B quadrature uses two signals, and therefore requires only 250 pulses per second on each line to transmit the same information. A/B quadrature signals have a lower frequency than the other two types of encoder signals. This is an advantage when operating in electrically noisy environments. AB quadrature signals also are more resistant to noise, as a noise pulse that is short enough is filtered, and one that is longer usually looks like a one count forward, one count back, causing its effect to be non-cumulative. With the Step/direction inputs, sufficient noise on the step line just causes the counter to continue to count.

See Application Note "QCI-AN019 Electronic Gearing" for more details.

## Dual Loop Control

In addition to simple encoder following control, the external(secondary) encoder input function can be used in a position feedback configuration. This Dual Loop Control operation uses the external (secondary) encoder signal count to replace the position portion of the internal (primary) encoder feedback in the PVIA control algorithm. The internal encoder position signal is still used for motor commutation, velocity estimation, and acceleration estimation. The SilverLode servo positions itself according to the external encoder. This feature is very useful for two applications: high-resolution feedback applications and applications requiring feedback directly from a machine or machine part rather than from the motor shaft. The input signal from the external feedback device can be in any of the three signal formats discussed in this section: step and direction, step up/step down (SilverNugget only), or A and B quadrature. Note that the default tuning gains will need adjusting for different encoder resolutions.

The Select External Encoder (SEE), the Dual Loop Control (DLC), and the Single Loop Control (SLC) commands are used for this feature. The SEE command is used to configure the servo to receive the external encoder input signal, just like when the signal is used for direct motion control. The DLC command configures the servo to use the external encoder signal for position feedback in the PVIA control algorithm. The SLC command puts the servo back in its default state of using the internal encoder for all control purposes. Command details are available in the Command Reference.

There are two applications where dual loop control using an external encoder is very useful:

The Dual-Loop function may also use SSI style encoders on SilverDust IG8 units. This allows dual loop control with absolute position feedback style encoders. The DLC command will use the SSI feedback if the SSI port has been configured as the Dual Loop source via the SSI command prior to the DLC command. Note that this frees up the step and direction controls to allow step and direction control of the units while operating in dual loop mode.

# Chapter 5: Troubleshooting Guide

## Troubleshooting Communications

With the SilverLode product powered up, start QuickControl and the polling routine should automatically find the device. If QuickControl is already running and the device is powered up, press the “Scan Network” button to find the device on the network. If “No Devices Found” appears in the Device Status Monitor, either the device has been initialized with something other than the Factory Defaults (listed in the Hardware Requirements section of this chapter) or QuickControl is not set up to communicate with the device in its present communications state. Some things to check if this happens are:

- Verify the COM port being used to connect to the device.
- Under Setup, select Comm Port / Comm Channels and ensure the baud rate and protocol are set to Factory Defaults (57600 and 8 bit ASCII respectively). Also, confirm the Enable checkbox is checked and the COM Channel enabled is the one device is connected to.
- Verify there are no other programs using the port (only one program can control a COM Port at any given time). These can include other motion control drivers or programs used for communication devices (e.g. Palm Pilot, HyperTerminal etc.). If these programs do not relinquish control of the port, QuickControl will report “Could Not Open select COM Port” and “Access is denied.” errors in the Status Log. Under Setup, select Register Devices and ensure all six devices have the Auto Reg checkbox checked.
- Run Tools ⇒ Unknown Device Wizard (only works with one device connected at a time)
- If networking multiple devices, QuickControl can only register six at a time for viewing.
- If networking multiple devices, each has to be initialized with the Initialization Wizard with a unique ID.
- Under Setup, select Options and make sure the Autoscan ID range encompasses the ID of the units desired.
- Reduce Baud Rate. If you are occasionally losing communications with the device, try reducing the baud rate. Remember to change both the PC's baud rate (Setup->Comm Port) and the device's (Init Wizard->Communications->Baud Rate).

After verifying the integrity of the COM Port and making any necessary changes, try a quick communication test. Stop the QuickControl polling routine (if it is running), and press the red hand icon (STOP) button on the QC toolbar. The red LED on the back of device should flash briefly as it receives and processes the Halt command sent to it each time the STOP button is pushed. This simple test can be done at any time to verify that the device is receiving commands.

It is also good to verify that the Windows PC can communicate through the serial port without QuickControl. Programs like HyperTerminal can accomplish this (see Technical Document QCI-TD024 8 Bit ASCII Protocol Using HyperTerminal). If communication is successful with the third party terminal program, then QuickControl should be able to operate correctly.

### SilverLode Indicator LEDs

There are two indicator light emitting diodes (LEDs) on the back of every SilverLode product, one red and one green. These LEDs provide the user some basic information about the current operational state. When the device is first powered up the green LED should be on (on the SilverNugget, the red LED

flashes on momentarily at power up as well). The red LED is the communication/program done indicator.

If the red LED is on solid (dim glow), then no program is running and the device is not communicating. When the device receives the start of a transmission, the red LED will shine extra bright. When the transmission is processed, the red LED will return to its original state. Sending the device a number of commands in succession (e.g. the QuickControl polling routine) will induce a flickering of the red LED. This flickering can occur in regular intervals or as random blinks depending on the communication scheme.

If the red LED is out completely, then a program is running from an internal program in the Program Buffer (and, in the absence of flashes, no communications is taking place).

The green LED is normally used to indicate motor torque, it starts out a middle brightness, glows brighter for positive torque and darker for negative torque. Its function changes if the Enable Done Low or Enable Done High command has been set; in this state, bright indicates that the motion is complete and the error is within set limits, while off indicates that the above are not true. The green LED is also used to flash error codes in the case of power low, over temperature, etc. according to the user and/or initialization program loaded. This is done by disabling the driver, setting open loop mode, and then setting the “torque” high and low (though with the motor drive disabled, only the LED is affected).

### Green LED Flash Code

Number of Flashes	Fault
1	Startup Recovery Program Kill Motor Condition occurred at startup.
2	Kill Motor Recovery Program Kill Motor Condition occurred after startup as a result of a condition in the KMC or KMX command.
3	Initialization Wizard needs to be run.
4	I-Grade Motor Memory Read Error
5	I-Grade Motor Memory Version is Not Compatible
6	I-Grade Motor Miss-Match. Attached motor does not match with device. Re-run Initialization Wizard
7	I-Grade Motor Memory Checksum Fault

The SilverDust IG and IGB also have a third yellow LED. This yellow LED lights to indicate that the encoder signals are not at valid levels indicating shorts, opens, or otherwise invalid levels. A bit in the Internal Status 2 (IS2) word indicates this state, and may be used to trigger a kill motor via the KMX command. (See IS2 and KMX).

# Troubleshooting Frequently Asked Questions

- **Why do I get a Driver Enable Error/Temperature Fault when I start up?**

Most of our controllers require a jumper between CLAMP+ and DRV ENA. This is done differently on each of our controllers.

IG: Jump pins 17 and 19 on the Auxiliary Connection.

IGB: This is found at the top of the controller.

IGC/IG8: Two jumper are needed. CLMP with ENA+ and ENA- with V-.

SilverNugget N3: Break out the SMI Port and jump pins 1 and 7.

- **How do I know what my 3rd party hardware/software is sending to the controller?**

QuickControl has a function called 'Data Monitor' which shows all of the transmitted and received data in the controller. This can be accessed by clicking on Tools->Data Monitor.

- **How can I check the value of my registers while the program is running?**

Click on the Tools tab in the toolbar and scroll down to Register Watch. This will open a dialog box that will allow you to see any register by clicking on the button Add Register. Another dialog box will appear with the list of all registers on the current device. Select the appropriate data format.

- **How can I see all Internal Status registers and the data it currently contains?**

QuickControl has a function called 'Diagnostics' which shows all of the registers contained within the controller and their corresponding data. This can be accessed by clicking on Tools->Diagnostics.

## Common Issues

### Controller

<b>Controller doesn't power up (No lights)</b>	No power to the controller (+12V to +48V)	Check the following are securely fastened:
		<ul style="list-style-type: none"> <li>• SMI cable &amp; breakout boards</li> </ul> <p>Ensure that the Processor switch is turned on, if equipped.</p>

		<p>Verify that 12 to 48 volts is applied to Input Power and Power Ground</p> <ul style="list-style-type: none"> <li>• SilverLode Multi-Function Interface (SMI) <ul style="list-style-type: none"> <li>V (+): Pins 1 or 7</li> <li>V (-) : Pins 6 or 11</li> </ul> </li> <li>• SilverNugget N3 <ul style="list-style-type: none"> <li>○ Apply 12 to 48 volts to Pin 1 (SMI) to enable the driver</li> <li>○ Apply 12 to 48 volts to Pin 7 (SMI) to power the processor</li> <li>○ Apply 12 to 48 volts to the DB3 connector to power the driver</li> <li>○ Ensure the clamp module's Power LED is on</li> </ul> </li> </ul>
	Unit was hot-plugged	Contact QCI Support.
<p><b>Controller doesn't communicate with PC, HMI, PLC, etc.</b></p> <p><b>(Power and/or Status LEDs flash)</b></p>	Communication wiring is incorrect	<p>Verify that serial cable is a straight through cable (pin to pin) and NOT a crossover cable nor a null modem cable.</p> <ul style="list-style-type: none"> <li>• Pin 2 → Rx</li> <li>• Pin 3 → Tx</li> <li>• Pin 5 → Logic Ground</li> </ul> <p>Try swap communications lines:</p> <ul style="list-style-type: none"> <li>• Rx --&gt; Tx</li> <li>• Tx --&gt; Rx</li> <li>• A --&gt; B</li> <li>• B --&gt; A</li> </ul> <p>RS485 signals are reversed between SilverNugget and SilverDust.</p> <p><b>Remove RS485 termination if communicating in RS-232.</b></p>

	<p>Incorrect COMM port selected in QuickControl</p>	<p>Go to Setup → COMM Port to modify COMM. port number.</p>
	<p>Unknown Device Wizard succeeds but Initialization Wizard does not. "Download Fail"</p>	<p>More than one unit is connected to the communication network. Isolate the controller.</p> <p>QuickControl doesn't have the current motor's parameters. Update devAux.txt file or update QuickControl version.</p> <p>Upgrade QuickControl to latest version.</p>
	<p>Unknown Device Wizard → No Devices Found</p>	<p>Verify that only one unit is connected to the PC.</p> <p>Verify that no other devices (HMI, PLC, etc) are connected to the communication lines.</p> <p>SilverNugget does <b>NOT</b> support Polling/Debugging with QuickControl setup for <b>MODBUS</b>.</p> <ul style="list-style-type: none"> <li>• Change COM port to 8-bit ASCII to communicate with controller → Unknown Device Wiz</li> </ul>

	<p>Communications parameters don't match</p>	<p>Set COMM Port in QuickControl to the following defaults:</p> <p style="text-align: center;">8-Bit ASCII, 2 Stop bits/ No Parity</p> <p>Run the Unknown Device Wizard, Verify COMM light flashes.</p> <p>Run the Initialization Wizard and verify communication parameters between PC and controller match.</p>
	<p>COMM port number out of range</p>	<p><b>QuickControl 5.xx and back, only support single digit COMM ports (1 → 9).</b></p> <p>Refer to Windows documentation to change COMM port number.</p>
	<p>COMM port is blown – Most likely unit was hot-plugged.</p>	<p>Measure impedance between Tx and Logic GND &amp; Rx and Logic GND</p> <p>Both should be within the range of 23KΩ &amp; 26KΩ</p>
	<p>RS-485 under powered</p>	<p>Increase voltage to the RS-485 module, QCI-RS485. Up to 50V max.</p>

	<p>Un-configured, bad or blown COMM port</p>	<p>Restart computer.</p> <p>If available, try a different COMM port and/or computer.</p> <p>If using a USB-to-RS232/RS485, verify the latest manufacture's drivers are properly configured.</p>
	<p>Power supply intermittent</p>	<p>Power supply set for 220VAC</p> <p>Change to 110VAC</p>
<p>Controller not responding as expected.</p>	<p>No Device Found</p> <p>"DIF" command not valid</p> <p>Driver not enabled.</p>	<p>Early firmware revisions don't support newer commands.</p> <p>Downgrade to QuickControl 4.2 and delete any commands not supported.</p>
	<p>Device busy or not responding</p> <p>Product line mismatch.</p>	<p>QuickControl version doesn't support device.</p> <p>Download appropriate version.</p>
<p>Controller starts up with "Driver Disabled"</p>	<p>Underrated power supply.</p>	<p>Connect a higher rated power supply.</p>

## QuickControl Software

Issue	Cause	Action
<p><b>Motor doesn't move to commanded position or at commanded Velocity/Acceleration</b></p>	<p>Motion parameters are initialized with wrong data format.</p> <hr/> <p>Motion parameters request for a slow motion.</p>	<p>Double-click register write commands ((WRP) and select proper data format.</p> <p>If communicating from a Host, verify that the transmitted data is in SilverLode units.</p> <p>Verify units in Register Watch</p> <p>Tools → Diagnostics → IOS tab → Bit #3:Trajectory Generator</p> <ul style="list-style-type: none"> <li>• If Bit #3 is ACTIVE(1), the motor is processing a motion.</li> </ul>
<p><b>“Prog Stop” &amp; “NO COMM” rapidly flashes under the device status monitor.</b></p>	<p>QuickControl is attempting to access a register read from an invalid location.</p>	<p>In the Register Watch window remove any invalid registers.</p> <p>SilverMax/SilverNugget valid registers</p> <ul style="list-style-type: none"> <li>• 0 through 40</li> <li>• 200 through 233</li> </ul> <p>SilverDust/SilverSterling valid registers</p> <ul style="list-style-type: none"> <li>• 0 through 255</li> <li>• 64000 64016</li> </ul> <p>SilverSterling valid registers</p> <ul style="list-style-type: none"> <li>• 0 through 255</li> <li>• 64000 through 64016</li> <li>• 50000 through 50511</li> </ul>
<p><b>QuickControl cannot open COM port</b></p>	<p>Windows 7 compatibility issue</p>	<p>Configure QuickControl to run as Windows XP in compability mode</p> <p>Go to QuickControl root directory (C:Program</p>

		Files/QuickControl).  Right-click on the QuickControl application. Click on the Compatibility tab. Check “Run this program in compatibility mode for” checkbox and select “Windows XP Service Pack 3”.
--	--	--

## Motor

Issue	Cause	Action
<b>Motor doesn't rotate (No holding torque present)</b>	Motor is disconnected.	Check the motor cable is securely fastened.
	Bad/damaged motor cable (s)	Swap motor cables.
	Controller is not initialized via the Initialization Wizard or is initialize with for a different motor.	Run the Initialization Wizard.  Refer to the Green Status LED Flash Code
<b>Erratic behavior (Holding torque present)</b>	Motor Oscillates	May require tuning.  Default tuning parameters are good for 10:1 inertial mismatch.
	Motor faults/kills at the start of a move	<ul style="list-style-type: none"> <li>• Motor underrated. Not enough torque to complete move.</li> <li>• Acceleration parameter too high</li> <li>• <b>Power supply underrated. Verify power supply can provide enough power to the motor(s).</b></li> </ul>
	Index phase is not aligned	On an N3 controller run the Initialization Wizard → Click Options → Check Index Phase Alignment

	Damaged encoder	<p><b>Open-Loop Test</b></p> <p>Open the Initialization Wizard. Click the <i>Open</i> button and select the "Factory Default Initialization - Open Loop.qcp". If necessary, correct the motor cable length and click <i>Download File to Device</i>.</p> <ul style="list-style-type: none"> <li>• The controller will ignore the encoder and all accumulated position error counts. Open the <i>Control Panel</i> and begin to jog the motor. If the motor spins, there's an issue with the encoder:             <ol style="list-style-type: none"> <li>1. A pushed-in shaft can damage the encoder</li> <li>2. Hot-plugging can damage the encoder and/or encoder flex.</li> </ol> </li> </ul>
	Loose coupling	Check couplers are tightly secured.
	Bad driver	Blown driver(s)

# Appendix I: Sample Programs

## Sample Programs

QuickControl comes preprogrammed with many sample programs, located in the QCI Examples Folder. Custom programs can be made by stringing together existing programs. For example, the Initialization program file contains the following programs:

### Main Init

The Main Initialization program contains all of the initialization commands for the device. This program is shown in the next section and is described in detail.

### Startup Recovery

The Startup Recovery is used if a Kill Motor Condition is tripped during execution of the factory default initialization. The default program, disables the driver and flashes the green LED once. See SilverLode Indicator LEDs later in this chapter for flash definitions.

### Kill Motor Recovery

This program is called whenever a condition set in the Kill Motor Conditions (KMC) command is tripped. This may be modified if special processing is required. The default program, disables the driver and flashes the green LED twice. See SilverLode Indicator LEDs later in this chapter for flash definitions.

### Power Low Recovery

This program is called whenever voltage drops below the specified threshold in the Low Voltage Trip (LVT) command and may be modified if special processing is required on a Power Low condition. The default program, disables the driver and ends the program.

### Flash Seq

A utility program that flashes the green LED the number of times specified in register 11. The program is called by the error programs to flash the LED. See SilverLode Indicator LEDs later in this chapter for flash definitions.

### Factory Block Fault

This program is run when a fault has occurred in the Factory Block (see Memory Model for details on Factory Block). This program simply adjusts to flash sequence to correspond with the other faults and runs the Flash Seq program. See SilverLode Indicator LEDs later in this chapter for flash definitions.

# Appendix II Initialization and Set Up

## Initialization File

It is good practice to save a record of the specific power up changes made to each device as a unique Initialization file. This file should contain all the command settings for any particular device in a specific application. The initialization file also contains application parameters such as the Identity, Serial Interface, Baud Rate, Kill Motor Conditions, Error Limits, etc. There should be one initialization file per axis on any multi-axis machine.

When using the Initialization Wizard for the first time, the Factory Default Initialization.qcp file is automatically opened in the Wizard File window. This is a read only file and cannot be overwritten. To save any changes made to the initialization file, press the Save As button and give your file a unique name. In future Initializations, you can then **Open** and select your specialized Initialization file instead of the Factory Default.

## Initialization Wizard Details

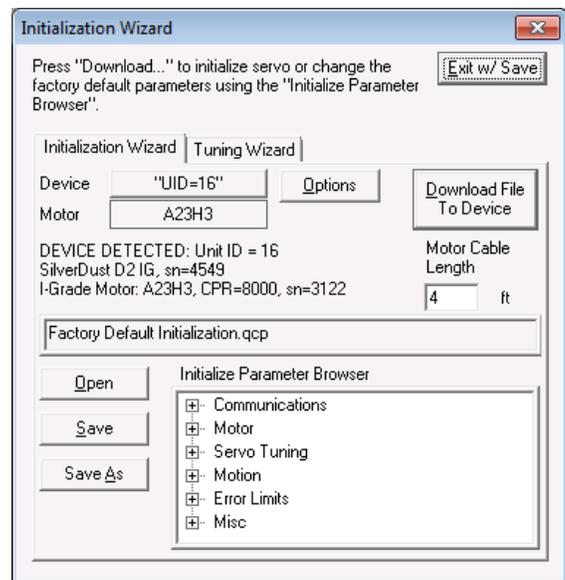
All SilverLode products must go through the Initialization Wizard at least once before operation or anytime a new type of motor/encoder is used.

**NOTE:** If the unit has previously been initialized, you might want to start the Initialization Wizard with the controller powered down to prevent any previously downloaded programs from running. For an un-initialized device (i.e. fresh from the factory), this is not necessary.

To run the wizard from Tools menu:

Tools -> Initialization Wizard

If your unit was powered up, you should see something like the dialog box shown. If using an I-Grade SilverDust with an I-Grade motor, the motor type, encoder type, and motor serial number will be displayed. Other combinations of motors and controllers will not display this information. If the controller is not powered up, then controller and motor type will not be displayed. Please see Appendix I for further information on specific Initialization scenarios.

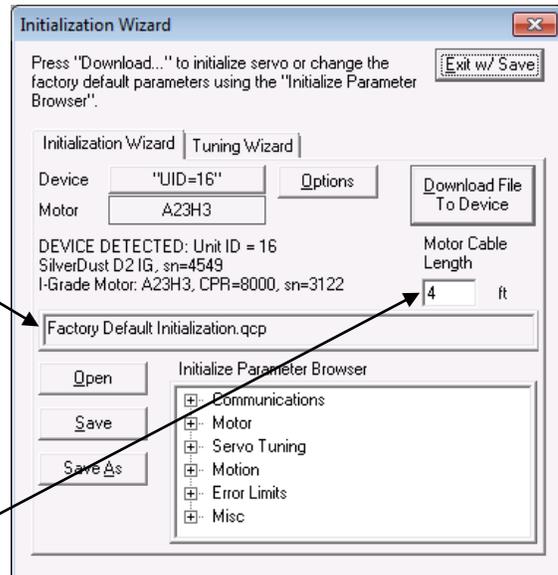




## Basic Default Initialization

This exercise demonstrates how to accomplish a basic initialization of the device using the Factory Default Initialization File.

1. To initialize the device using the wizard, begin by choosing “**Initialization Wizard**” under the “**Tools**” pull down menu.
2. Verify the filename “**Factory Default Initialization.qcp**” is listed in the **File** box.



3. Verify Motor Cable Length is correct.
4. Select the “**Download File To Device**” button to have the default initialization file downloaded to the device.
5. The servo will automatically restart and the parameters will then become active on each successive power up and motor restart cycle.
6. The Program Downloaded screen will appear when downloading is complete. Press the “**OK**” button to clear the message screen.
7. To exit the Initialization Wizard, select the “**Exit w/o Save**” button on the screen.
8. If polling is stopped, press the “**Scan Network**” button to verify proper communications and polling of the device.

# Initialization Scenarios

The following are typical initialization scenarios. Please note, the Initialization Wizard must be re-run anytime the encoder is loosened, or a different motor/encoder pair is connected.

## Scenario 1: I-Grade SilverDust Controller With I-Grade Motor

This is the most common and simplest case because the I-Grade SilverDust reads most of the information it needs from I-Grade motor memory.

1) Press "Download File to Device".

The Initialization Wizard uploads motor ID and encoder information from the I-Grade motor memory and initializes the motor.

2) Finish

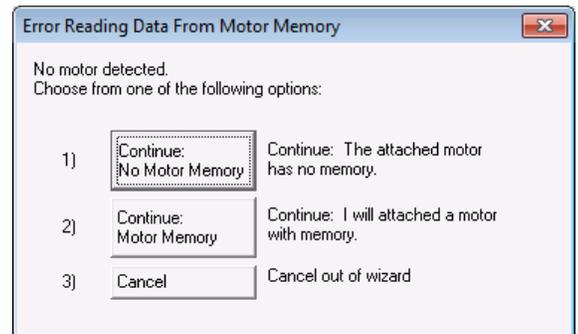
A screen will appear indicating that the selected initialization qcp file has been downloaded. Select OK to exit. The unit should now be initialized and ready for operation. (It may need to be restarted if the No Restart on Download option was selected.) Press Exit to leave the Initialization wizard.

## Scenario 2: I-Grade SilverDust With Non I-Grade Motor

1) Press "Download File to Device".

The Initialization Wizard fails to upload the motor data and displays the "Error Reading Data From Motor Memory" dialog box.

2) Error Reading Data From Motor Memory Dialog Box



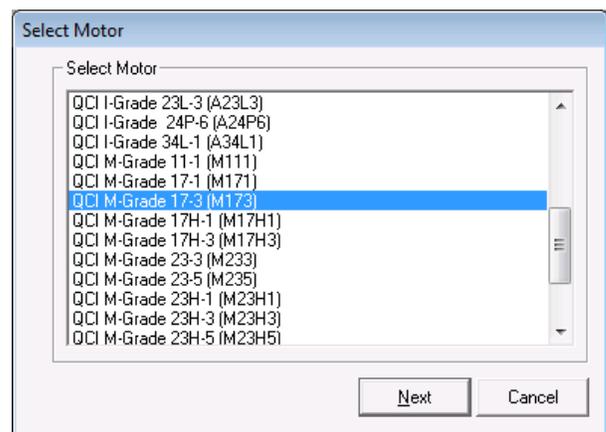
The following choices are presented:

- Continue: No Motor Memory: Select for non-I-Grade motors.
- Continue: Motor Memory: Select if an I-Grade motor will be used, but is not currently attached to the SilverDust.
- Cancel:

Press " Continue: No Motor Memory "

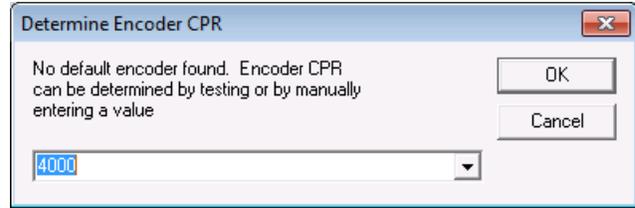
3) Select Motor Dialog Box

Select the desired motor and press Next.



#### 4) Determine Encoder CPR (optional)

If you select a non I-Grade motor, you will get this dialog box after you edit the **Custom Motor** dialog box and Press OK. If you know your encoders Counts Per Revolution (CPR), select it from one of the options. You can also select Test Encoder. Note selecting Test Encoder will require that the wizard rotate the motor.



Press OK

If you selected Test Encoder you will a message warning you about the motor moving, otherwise the wizard will finishes like Scenario 1 above.

### Scenario 3: SilverNugget or SilverDust MG

1) Press "Download File to Device".

2) Select Motor Dialog Box ....

The rest is the same as Scenario 2 above.

### Scenario 4: SilverDust with Custom Motor or 3rd Party Motor

QuickControl can initialize a SilverDust controller to servo most stepper motors as long as they are within the current limitations of the driver (see controller datasheet). QuickControl needs the following information:

1) Press "Download File to Device".

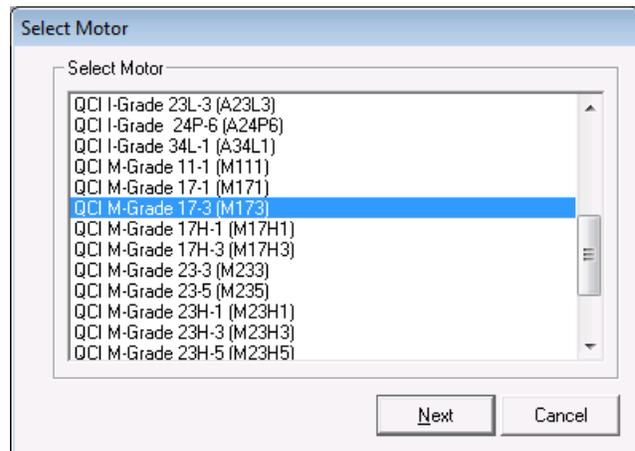
2) Error Reading Data From Motor Memory Dialog Box (I-Grade SilverDust Only)

Press "Continue: No Motor Memory"

3) Select Motor

Select "Custom Motor" for you controller.

Press Next

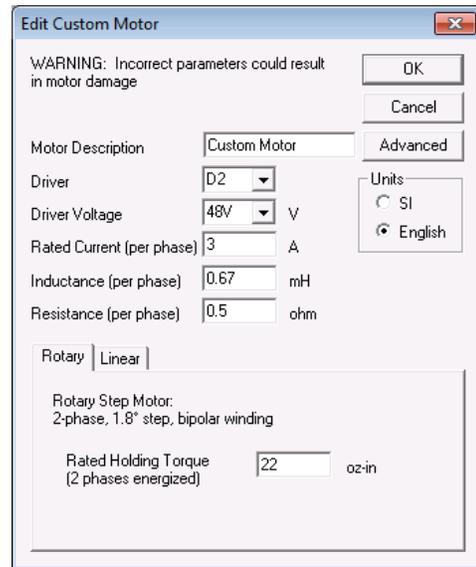


4) Enter this information from the 3rd party motor's datasheet.

Press OK.

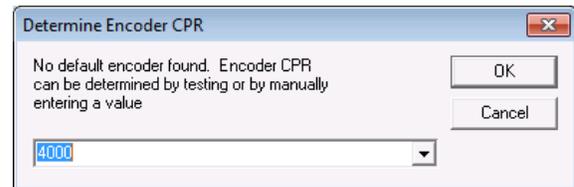
NOTE: Only edit Advanced options with QuickSilver Tech Support's guidance.

NOTE: This information is stored with MCT command in the Initialization Program File (i.e. My Custom Init.qcp).



5) Determine Encoder CPR (optional)

The rest is the same as Scenario 2 above.



### Scenario 5: I-Grade SilverDust Controller With I-Grade Motor To Be Attached in the Field

Since all the pertinent motor and encoder information is contained in the motor's non-volatile memory, it is possible to initialize an I-Grade SilverDust without the I-Grade motor attached. Simply run the Wizard without the motor attached.

1) Press "Download File to Device".

2) Error Reading Data From Motor Memory Dialog Box.

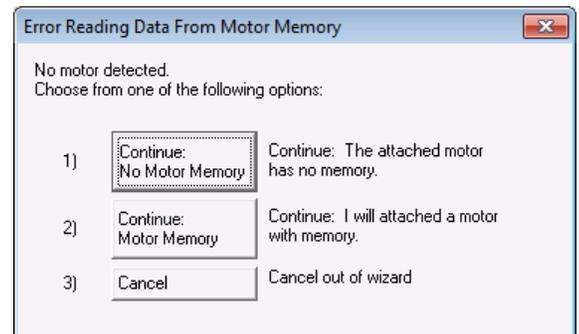
Select Option 2 "Continue: Motor Memory "

3) Select Motor Dialog Box

Select the desired motor and press Next.

3) Finish

The wizard finishes as normal. Power down the controller, attach motor and power up. As long as the selected motor matches the attached motor the controller will work fine.



## Initialization Wizard Options

Selecting the **Options** button will allow you to change the initialization options:

- **Restart on Download** (default checked): This option selects whether the device is to be restarted automatically after each initialization file download. In order for any changes in the initialization file to take affect, the device must be restarted.  
The device can be manually restarted by pressing the Restart button on the Program Info Toolbar or by cycling the device's power. If you are changing the Identity, Baud Rate, Serial Interface, or Protocol, delaying the restart until all user programs have been downloaded may simplify the process.
- **Maximum Velocity Scale** (default 4000RPM): This option allows the user to select the default 4000 RPM or to change the velocity scaling in the system for lower top speeds. With a lower top speed, the scaling for velocity parameters as well as the actual velocity registers is changed so that a full scale value for each of these will correspond to the selected velocity. The velocity and acceleration terms in the CTC and FLC commands may need alteration (to be lowered) in your application if a value other than 4000RPM is selected. Velocity control for systems that do not need higher velocities may be improved by selecting a lower Maximum Velocity.
- **Advanced Line Resistance** (default unchecked): This option causes an addition dialog box to appear anytime a device is being initialized that allows for calculating resistance for user cables using various wire gauges.
- **Encoder Resolution Reduction** (default Divide By 1): This option is only available for SilverDust units. It is intended to allow programs written for older 4000 CPR (count per revolution) systems to be easily adapted to the newer 8000/16000 CPR motors. A divide by 1 provides the full 8000 CPR resolution when using an 8000 CPR encoder. The divide by 2 option "downgrades" the resolution, so the system behaves as if a 4000 CPR encoder is attached.
- **Save On Exit** (default checked): Check this button to save the file (to the PC not the attached device) when the "Exit..." button is pressed. Note, the "Exit..." button's name with change to "Exit w/ Save" when this box is checked.
- **Erase Application Programs** (default checked): When wizard is run, the user's application programs stored in the controller are erased. This option can be unchecked as long as Index Phase Alignment is not used. The wizard must erase the user program(s) in order to make room for the Index Phase Alignment program run within the wizard.
- **Choose Motor/Configure Encoder** (default unchecked): If this option is checked, the user will be prompted to select a motor every time the wizard is run. If this option is not checked, the wizard will only prompt the user to select a motor if it thinks it needs to. Examples of this would include:
  - Un-Initialized Device (fresh from factory)
  - Index Phase Alignment Option Changed

NOTE: If you want to re-initialize a controller for a different type of motor/encoder, check this option at least once. This will force the wizard to prompt for you for a new motor type.

NOTE: This option is ignored when using the SilverDust IG or IGB with an I-Grade Motor, as the motor type is automatically read from the non-volatile memory inside the I-Grade Motor. See SilverDust IG/IGB With I-Grade Motor Memory Initialization for details.

- **Index Phase Alignment** (default checked): If this box is checked, the servo will use the encoder index (z channel) signal to set its phase alignment at power up. If this box is not checked, the servo will rely solely on the initialization program's phase alignment.

### Unknown Device Wizard

The Unknown Device Wizard is helpful if a device is not automatically recognized by the QuickControl software. When the Unknown Device Wizard is run, it prompts the user to cycle power to the device while the wizard is sending out a continuous stream of HLT commands through the Comm port. At power up, the controller listens using each protocol and serial interface while configured for 57600 baud. When the controller “hears” the halt in the given format, the controller will halt, otherwise the normal startup routine will run. The wizard will then prompt for communication configuration of the device. These settings should match how QuickControl is configured. Note that the protocol (8-Bit ASCII or 9-Bit Binary) is determined by the settings under Comm Port in the Setup menu of QuickControl to ensure that the protocol matches QuickControl. Please refer to Technical Document QCI-TD053 Serial Communications on our website for a complete discussion on differences between serial interfaces. Upon completion of the wizard, the device will be successfully communicating.

NOTE: When using a SilverDust with the E-485 Bridge option (Ethernet), the wizard will take a minute or so to complete.

### Download File To Device

Press this button to start initializing your device.

Note if the unit was powered down previous to starting the initialization process, or if the Baud Rate, Protocol, or Serial Interface do not match the connection to the PC, the Unknown Device Wizard is automatically invoked to establish communications with the device.

### Default Configuration

When shipped from the factory, the SilverLode product and QuickControl software are initialized with default values that are used to establish initial communications between device and a PC. Remember, these values can then be edited during the initialization procedure.

Unit ID	<b>16</b>
Supply Voltage	<b>48VDC</b>
Serial Communications Protocol	<b>8 Bit ASCII</b>
Serial Interface	<b>RS-232 (RS- 485 for SilverSterling)</b>
Baud Rate	<b>57600</b>

For SilverDust controllers with the E-485 Bridge (Ethernet) option,

Unit ID	<b>16</b>
Supply Voltage	<b>48VDC</b>
Serial Communications Protocol	<b>8 Bit ASCII</b>
Serial Interface	<b>RS-485</b>
Baud Rate	<b>57600</b>
IP Address	<b>DHCP</b>

## **Typical Configurations**

The following describes typical configurations for SilverLode servo systems. This section is not an all-encompassing list. There are many variations and additions that are possible. However, these examples should provide a good starting point.

For detailed examples on using any particular controller/driver in a system, please refer SilverLode Controller/Driver Datasheets on our website.

### ***Standalone Configuration***

SilverLode servos are capable of operating as a system-level controller without any input from a master controller or user interface. To function in this manner, SilverLode servos are pre-programmed to produce the desired motion and to respond to any sensors or other inputs in the system.

Often in this configuration, the digital I/O is used to initiate the required operations by programming the product to start, end, or select programs using different I/O combinations.

### ***Host Configuration***

Host configuration involves a SilverLode servo that is entirely controlled by a host PC, Programmable Logic Controller (PLC), Human Machine Interface (HMI), or other such device. The host is connected to the device through the RS-232/485 serial port or the E-485 Bridge (Ethernet).

Operation in host configuration mode has the SilverLode configured as a passive slave to the host controller, waiting for each command before performing any operation. After the initialization routine has completed, no additional internal program stays running that would allow the device to perform an operation by itself. This allows for extended operations not available in a SilverLode only network. In cases where many axes of control are required, a host configuration can achieve very complex or highly coordinated multi-axis control. A host controller can also retrieve data from registers.

### ***Hybrid Configuration***

A hybrid configuration utilizes a SilverLode servo operating such that the servo receives command from an external controller, executes internal programs, and uses its internal I/O. This configuration is more versatile than either a pure standalone or pure host-controlled configuration. The SilverLode product can use its standalone abilities to execute internal programs and interact with the system through its I/O and registers. Also, an external host can issue commands or interact with programs.

### ***Multiple SilverLode Servo Configurations***

A SilverLode product can be part of a small RS-232 or a larger RS-485 network. In addition, two or more devices can be interconnected through their I/O lines. The possible configurations for multiple device systems are seemingly endless, although all of the configurations are just combinations of the two basic configurations listed in this section. Many of the advanced applications for SilverLode products use this capability. In CANopen™ operation, controllers may also interact with each other as peers.

# Setup Menu Details

## Comm Port

From the main menu, select:

Setup -> Comm Port

If the Communications Device Properties shows the correct communications port, check the Enable box and press OK.



Press **Modify** to edit the Comm Properties.

## Register Devices

QuickControl will monitor the status of a device when the device is registered into the system and polling is enabled. The polling function continually checks the status of any registered devices and updates the Control Panel, Register Watch and the Device Status Monitor. Registration can be done automatically (recommended) or manually.

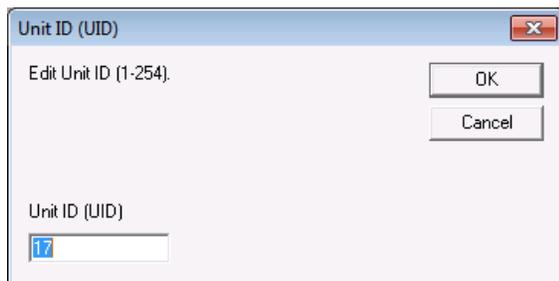
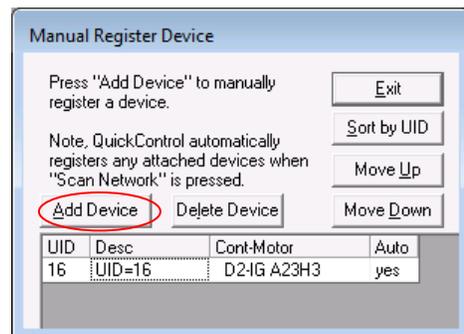
Anytime Polling is started, QuickControl will automatically scan all for any devices within the selected range ( default range is addresses 1-20). If a device is found, the device will be automatically registered and will appear in the Device Status Monitor screen.

To manually register a device, select,

Setup->Register Devices

Press “Add Device” to manually register a device.

Note: Manually registering is best to use if you want QuickControl to communicate with a non-QCI motor.

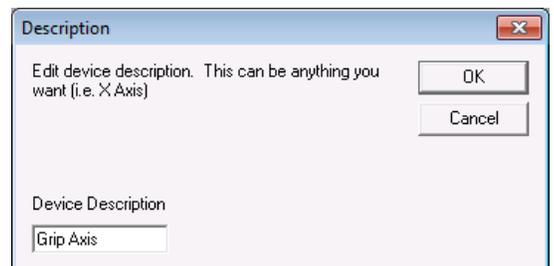


**Unit ID:** Enter the device’s Unit ID (UID). The device’s Unit ID is entered here. Unit ID for devices must be unique. Factory default is 16.

Press OK.

**Description:** Each device should have a description so that the user can have a quick reference for the status of the device. Use only a brief description. This is the description that appears in the Device Status Monitor.

Enter something that describes the device, such as: X Axis, Gripper, Shuttle, etc.

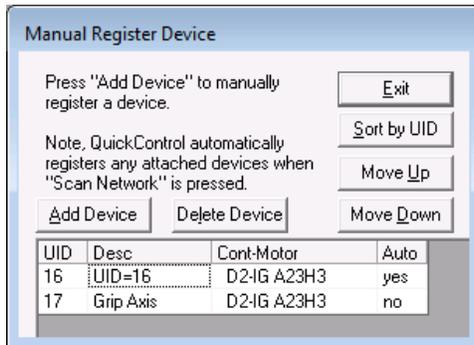
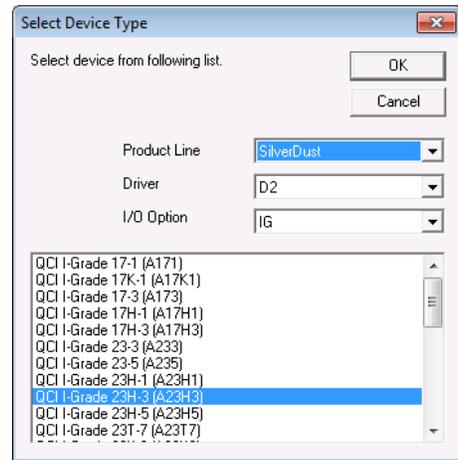


Press OK.

**Select Device Type:** Select the Product Line/Driver, I/O Option, and motor that matches the controller/motor you are registering.

Press OK.

The device will now appear on the list.



Double clicking in the Auto cell will toggle whether the device is Auto registered or not. When Auto registered, all fields except for Description are filled automatically by reading the device information.

You can change the order by pressing Move Up, Move Down or Sort by UID.

## Register Device Options

### Prompt On Delete (default checked)

Checking this box will cause QuickControl to prompt you when you do delete operations.

### Automatically Scan ID Range

This is the Unit ID range that will be scanned for when the Device Status Monitor, Scan Network button is pressed. The default is 1-20 which makes the "scan" a reasonable length of time. NOTE: This also sets the range for the SilverLode Identify (IDT) command.

### Disable Automatic Scan

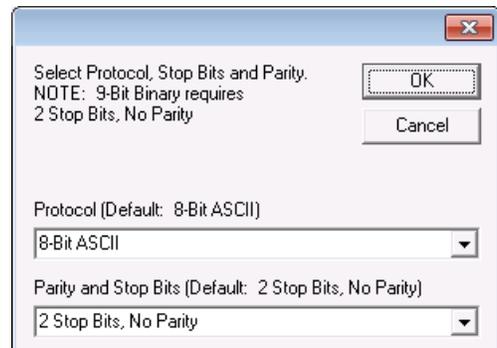
Check this box to skip network scan when Scan Network button is pressed. Only the manually registered devices will be polled.

### Baud Rate

PC baud rate ranges from 9600 to 115K. Default is 57.6K. It is a good idea to leave it here unless you are familiar with serial communication.

## Protocol

Each port needs to know how to communicate with the devices connected to it. The protocol sets up the Comm port so that it can send and receive data in the proper format. The protocols listed are the ones currently supported, including QuickSilver 8-Bit ASCII, QuickSilver 9-Bit Binary, and Modbus™.



NOTE: Changes to Baud Rate and Protocol only affect the PC. They do not affect the attached device. You must use the Initialization Wizard to change the parameters in the device. Therefore, if you want to start talking to the device at 9600 baud, you will first have to change the baud rate for the device, and then change the baud rate for the computer.

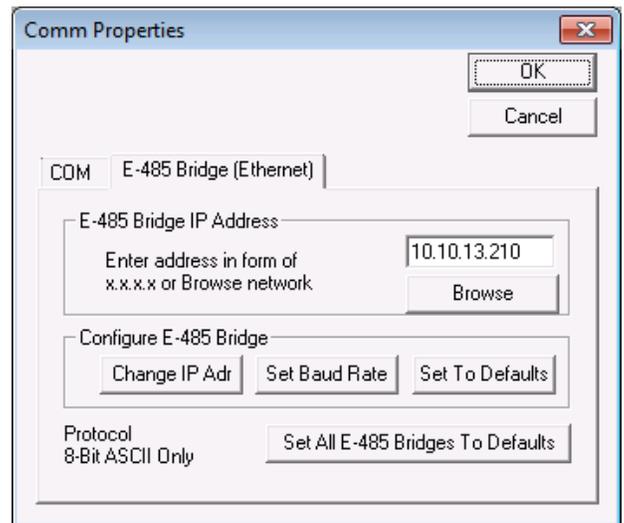
Stop bits and parity need to be matched between the comm port and the controller. Normally 2 stop bits and no parity are use.

### E-485 Bridge (Ethernet)

This tab is for editing the properties of the optional E-485 Bridge. The E-485 Bridge is a module on the controller that accepts standard Ethernet and converts or "bridges" it to the controller's RS-485.

#### E-485 Bridge IP Address

Press **Browse** to have QuickControl browse the local subnet for devices with E-485 Bridges. See "Set All E-485 Bridges to Defaults" below if **Browse** does not find an attached device. Alternatively, enter the IP address of the E-485 Bridge you want QuickControl to connect to.



#### Configure E-485 Bridge

Configure the E-485 Bridge with the above IP address. Note, the following functions configure the E-485 Bridge on the attached controller. The bridge will use the new configuration both now and on future power ups.

- **Change Adr (default DHCP):** Allows the user to change the E-485 Bridge's IP address or configure it to use DHCP.
- **Set Baud Rate (default 57.6K):** Allows the user to change the E-485 Bridge's RS-485 baud rate. This is the baud rate the E-485 Bridge will use to communicate to both the local servo controller and any other controllers attached to the RS-485 network. Note, the controller's baud rate (i.e. BRT command) must match the baud rate set here.
- **Set To Default:** Press this button to download the factory default configuration to the E-485 Bridge. Note, this only configures the E-485 Bridge and does not alter the controller's memory.

#### Set All E-485 Bridges to Defaults

Press this button to download the factory default configuration to all E-485 Bridges on the local subnet (see "Set To Default" above). This is sometimes required before QuickControl can "Browse" a device. For example, if a device was previously configured for IP address 10.10.10.200 and was then moved to subnet 10.10.20, "Browse" would not find it.

For additional information on the E-485 Bridge, see Technical Document "QCI-TD056 E-485 Bridge – Ethernet".

### Phase Alignment

The alignment of the motor and encoder needs to be calibrated to function optimally, so while all I-Grade motors and encoders are calibrated in the factory, the Phase Alignment is necessary for any

systems that include a non-I-Grade motor or encoder. Phase Alignment (Commutation) is accomplished using the position of the encoder with respect to the motor's rotor.

### *Start-Up Phase Alignment*

Every time the servo powers up, the initialization program calculates a Start-Up Phase Alignment by moving the motor back and forth a little. It can then go closed loop and begin to servo. This works well in many applications as long as the motor is allowed to move. If your application has conditions that may stop the motor from freely moving at startup, or needs more tightly controlled torque, additional Phase Alignment is suggested. Some such cases include having a motor brake, a vertical load, if the motor is jammed or if high inertias or higher power motors are involved.

If the motor does not move sufficiently during the Startup Phase Align, it will calculate an incorrect phase alignment. This will cause the motor to have poor performance. It might not even be able to move, or it may move erratically. To allow Phase Alignment for these startup conditions, the alignment of the encoder with respect to the rotor must be determined a different way.

### *Index Phase Alignment*

Index Phase Alignment uses the encoder's index pulse (z-channel) position to calculate phase alignment. For this to work, the index position with respect to the motor's rotor is determined by moving the motor with no load. The alignment is saved to non-volatile memory in the controller. Changing either the motor or controller requires realignment.

If an I-Grade motor and a SilverDust or SilverSterling I-Grade controller are used together, the motor's memory holds the alignment information and is automatically used at power up. The calibration is performed at the factory. The internal alignment will be used whether or not the Index Phase Alignment option is selected. This allows motors of the same type to be swapped without rerunning the alignment.

Regardless of controller, if Index Phase Alignment is used, the servo will only use the Startup Phase Alignment until it sees the index signal at which time it will read the phase alignment information from non-volatile memory. The I-grade motors use a special z-channel with 49 pulses on a 50 pulse spacing, making the motion needed to find a z-channel marker much shorter.

### *Cyclic Phase Alignment*

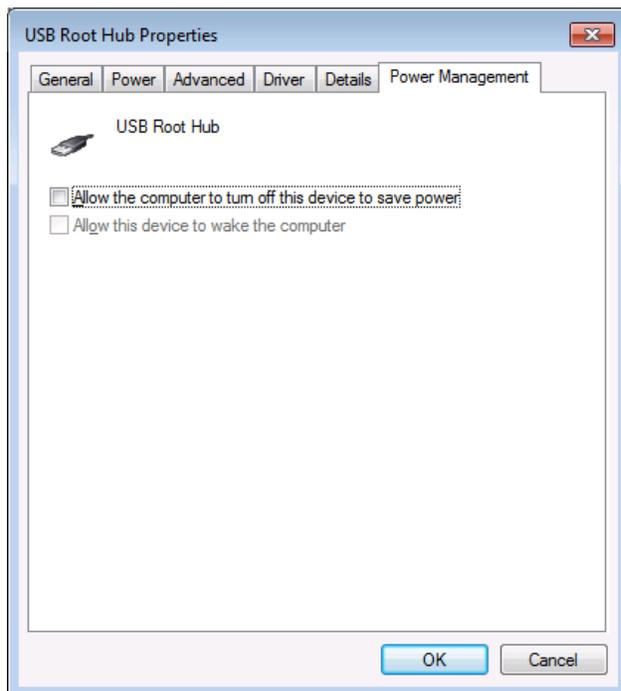
This is a modified Startup Phase Alignment that repeats the start-up move over and over again (cyclic) until a valid Phase Alignment is determined. This is a good option for applications that do not support Automatic Index Phase Alignment, but still need improved Phase Alignment while maintaining the ability to swap motors without re-running the Initialization Wizard.

To use Cyclic Phase Alignment, from Initialization Wizard; open the file Factory Default Initialization - Cyclic.qcp instead of Factory Default Initialization.qcp. Make sure Index Phase Alignment is unchecked under **Options**.

# Appendix III Configuring Your PC

## Auto Power-Down Options

Many recent drivers include a “Power Management” option which allows the PC to power down the selected device when the computer (thinks it) is idle. These devices include the USB hubs and the USB serial ports. Unfortunately, if the keyboard is not in use, the serial port can be powered down even while QuickControl is still using it, causing a fault. To prevent this, it is necessary to uncheck the “Allow the computer to turn off the device to save power” option under the Power Management tab for both the USB serial port you are using, and for the USB hub which services it. If you are not sure, disable the power save (uncheck the box) for all of your USB hubs.



Go to the Device Manager, select each device, right click and select properties and the properties box will appear. Select the Power Management tab, and unclick the save power box.

The Device manager may be found by opening up Windows Explorer, selecting the computer name and right clicking properties. Next select the Device Manager tab.

# Appendix IV QuickControl® Overview

SilverLode products have an extensive command set that allow them to be programmed for a wide variety of complex applications. SilverLode products need to be pre-programmed if they are to be used in a standalone or hybrid configuration.

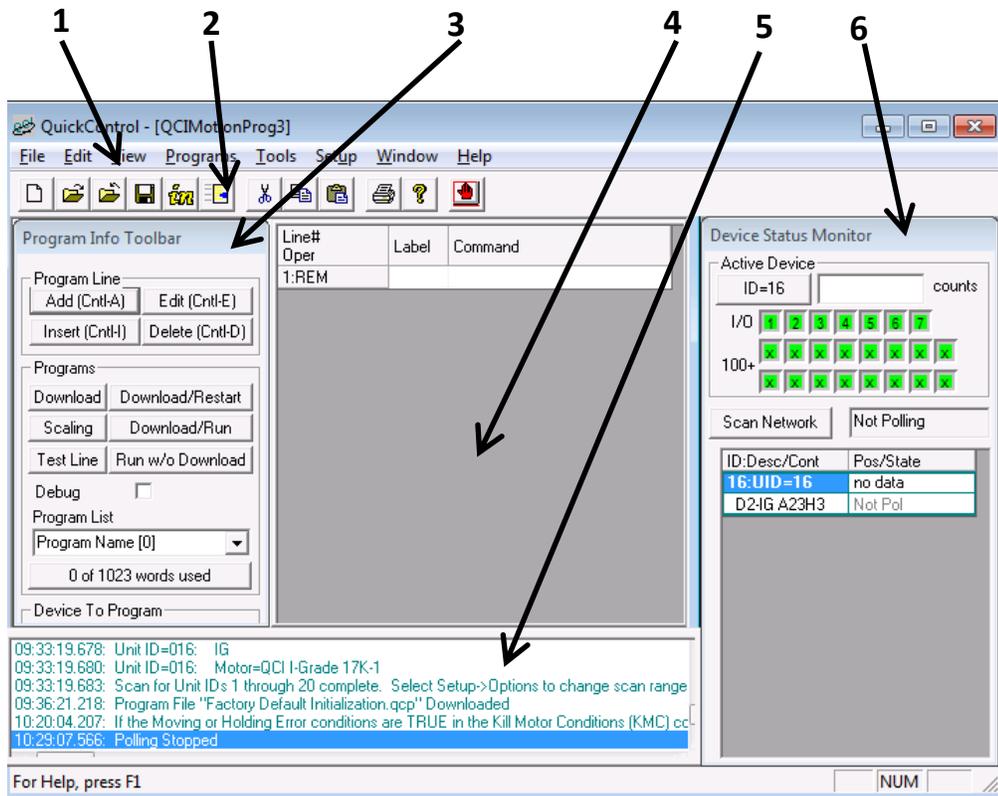
SilverLode products are programmed from a series of commands issued through a serial communications link. The most practical way to program a SilverLode product is to issue these commands from the QuickControl software running on a PC. QuickControl is QCI's Windows-based software interface for the SilverLode products. QuickControl can run on an XP, Vista or Windows 7 based PC connected to a SilverLode product using one of the PC's serial ports. QuickControl is designed to make programming SilverLode products easy and efficient. It is also the only programming interface fully supported by QCI and is required to change the factory default initialization program.

## QuickControl® Interface

The main QuickControl screen offers the user an easy to use programming interface with the ability to monitor the communications status of up to 6 active units connected to the PC Host. An active status log is available for viewing the status information the active device is sending to the host. In addition, any device connected to the host can be selected in order to view current position and the active I/O states.

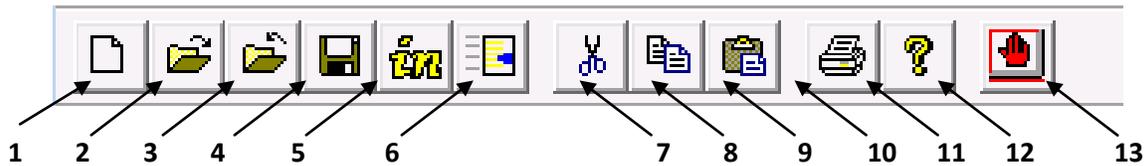
The Main QuickControl Screen is divided into five major sections (see figure below).

- 1) The Menu Bar contains pull down menus with all the functions of QuickControl.
- 2) The Icon Bar contains the most often-used menu items as shortcuts.
- 3) The Program Info Toolbar displays programming and program information.
- 4) The Program Window displays the active program.
- 5) The Status Log displays messages.
- 6) The Device Status Monitor provides information about all connected devices.



## Toolbar

The QuickControl Toolbar provides shortcuts to certain functions that may be used repetitively while using the QuickControl software.



<b>1</b> Create a New Program File (.QCP).	<b>2</b> Open an Existing Program File (.QCP).	<b>3</b> Close the Active Program File.
<b>4</b> Save the Active Program File (.QCP) to PC's Hard Drive.	<b>5</b> Initialization Wizard	<b>6</b> Control Panel
<b>7</b> Cut Selected Lines From Program File	<b>7</b> Copy Selected Lines From Program File	<b>8</b> Paste Previously Cut or Copied Line into Program File
<b>9</b> Print Active Program	<b>11/12</b> Ver/Help	<b>13</b> Stop all motion/executing programs on all devices.

**STOP**  The STOP button stops all command execution and programs running in all devices connected to the PC. It can also be used to verify communications to any device as the Red LED blinks briefly when issued. (Will not affect units running Modbus® protocol.)

## Program Window

Line#	Label	Command
1:REM	START	
2:MRV		Move 1234 revs @ acc=99 rps/s vel=33 rps Stop when "I/O #1" is Falling
3:REM		Jump to label START when I/O #2 = low state
4:JOI		Jump On Input to "START" When "I/O #2" is LOW/FALSE

This area of the QuickControl Screen displays the active program opened. It is where all program lines are created, viewed, and edited. The interface allows the user to single click on the line to highlight that particular line or double click on the line to edit the contents (parameters) of that line.

There are three columns in the Program Window.

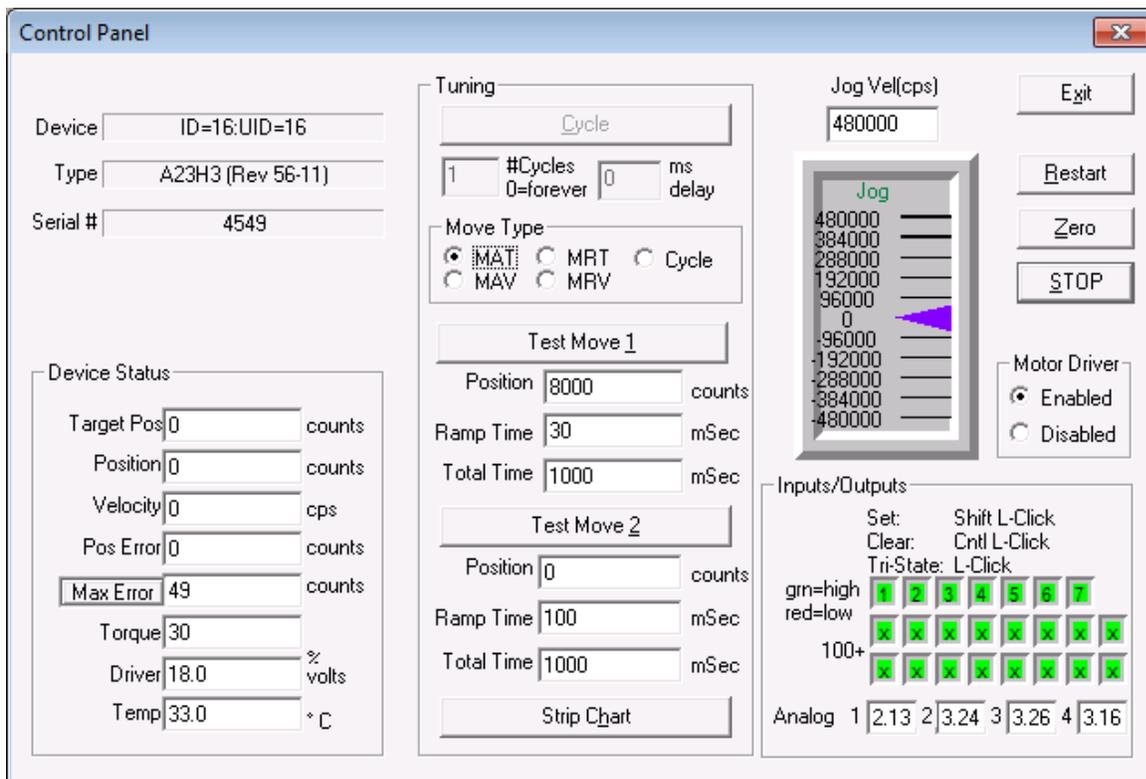
- 1) **Line Number and Operation** – Displays the program line number and the Three Letter Acronym (TLA) of the command.
- 2) **Label** – Allows the user to put in labels for branching operations in programs.
- 3) **Command** – Provides a brief summary of the command on that line and the parameters that are set in that specific operation of the command.

In this example of the Program Window, there are two colors used to differentiate the lines of the program. Blue is used for actual commands and green is used for the remarks (REM).

Remarks are not downloaded with the commands and are only used for documenting a QCP. When a label is placed on a REM line, the label is effectively moved to the next command line. If the label is placed on a REM line that is at the end of a program it will be the next available command line up from that REM line. In the example above, "START" is a label on line 1, a REM line. The program branches to the START label from Line 4 when the I/O condition is met. Since line 1 is a REM line, when this program is downloaded to the device the START label will actually point to line 2, the next command line after the REM line.

### Control Panel

The Control Panel is a Tool in QuickControl that provides access to several important features, such as jogging the motor, toggling inputs/output functions, tuning and testing, as shown below. Additionally, a strip chart can be displayed to show various motion parameters and is useful while tuning.



## *Device Status*

As long as Polling is running, these fields display the device's status in Real-Time. The fields include:

- Target Position: The position the servo is being commanded to.
- Position: The actual position of the servo.
- Velocity: The servo velocity.
- Position Error: The difference between Target Position and Actual Position.
- Max Error: The maximum error that occurred since last time the Zero button was pressed.
- Torque: The servo's torque in percentage of full torque.
- Driver: The driver voltage (which may be independent of the processor voltage on some units)
- Temp: The measured processor temperature.

## *Tuning*

This section, in conjunction with the Initialization Wizard's Tuning Wizard, provides the tools required for tuning the servo loop (see Technical Document QCI-TD054 Servo Tuning for details). Ideally, the servo would be attached to the axis needing tuning with the real world load. Two moves are provided to allow the developer to move between two positions (ie. Extend to pickup a widget then retract back to home). Using an iterative process, the developer would do a move, examine the results on the Strip Chart, adjust the tuning parameters and start again.

## *Cycle*

Press this button to cycle between Move 1 and Move 2. This is just like pressing the Move 1 and Move 2 Test buttons. If multiple cycles are desired, enter the desired number into the #Cycles field. A "0" in this field will make the axis cycle forever. Press the Stop button at anytime to interrupt a cycle. Enter a non-zero value in the "ms delay" field, if some settling time is required between moves.

## *Moves*

The four basic move types are available:

- MAT: Move Absolute, Time Based: Move to an absolute location in a specific time.
- MAV: Move Absolute, Velocity Based: Move to an absolute location at a specific velocity.
- MRT: Move Relative, Time Based: Move: Move some distance relative to your current position in a specific time.
- MRV: Move Relative, Velocity Based Move some distance relative to your current position at a specific velocity.
- Cycle: Select to enable Cycle (see above).

See the Command Reference Manual or Chapter 2 of this manual, Basic Motion Commands for more details on these commands

Once your Move Type is selected, enter the move parameters and press one of the Test buttons or Cycle to execute the move.

## *Jog Slider*

The motor velocity can be set in real time by sliding the pointer with the mouse. The Jog Vel field is used to set the full scale velocities at each end of the slider. Using the mouse, run "Jog Vel" slider up and down to jog the servo.

NOTE: If the Shift key is pressed when the mouse is released, the servo will continue at the same velocity (it will not stop). This is useful if you want to test constant velocity application.

### Zero

Press this button to zero the motor target and position. It also zeros out the Max Error latched value.

### Motor Driver

This radio button allows the motor driver to be enabled or disabled.

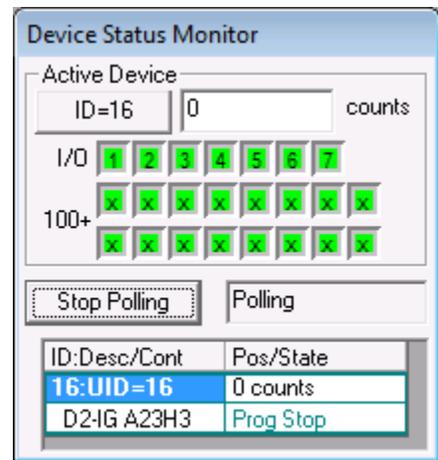
### Inputs/Outputs

This displays the states of all 7 I/O Channels. A “Red” I/O Channel indicator denotes a logic low and a “Green” indicator denotes a logic high. If the SilverDust IGB is connected, the additional 16 I/O are also shown.

Please See Chapter 4: Input Output Functions for more information.

### Device Status Monitor

The Device Status Monitor occupies the right hand portion of the main QuickControl window. It provides status information on QuickControl and the attached devices.

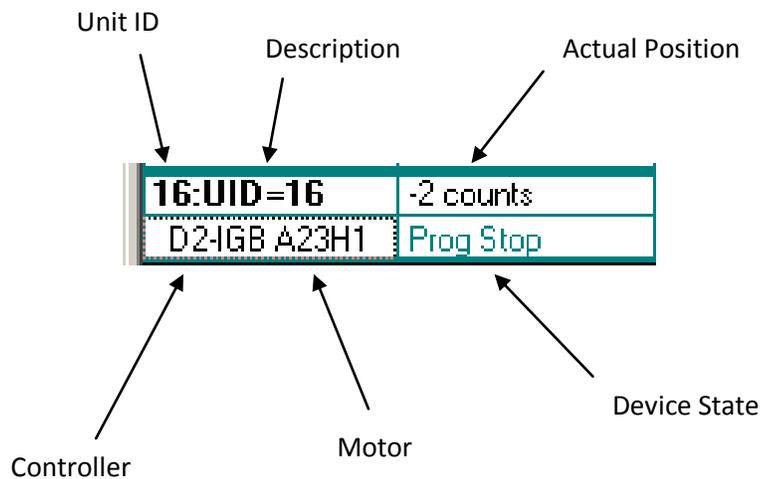


### Active Device

The status of the active device (selected by pressing any cell in the Device Status List) is displayed anytime polling is running. This includes the current position of device and the I/O Status. If the device ID button is pressed (i.e. ID=16), the Register Watch tool will come up.

### Device Status List

This portion of the display is used as a quick reference to the Registered Devices. QuickControl uses a polling routine to check the status of these devices. Each device has four (4) cells which contain the following data:



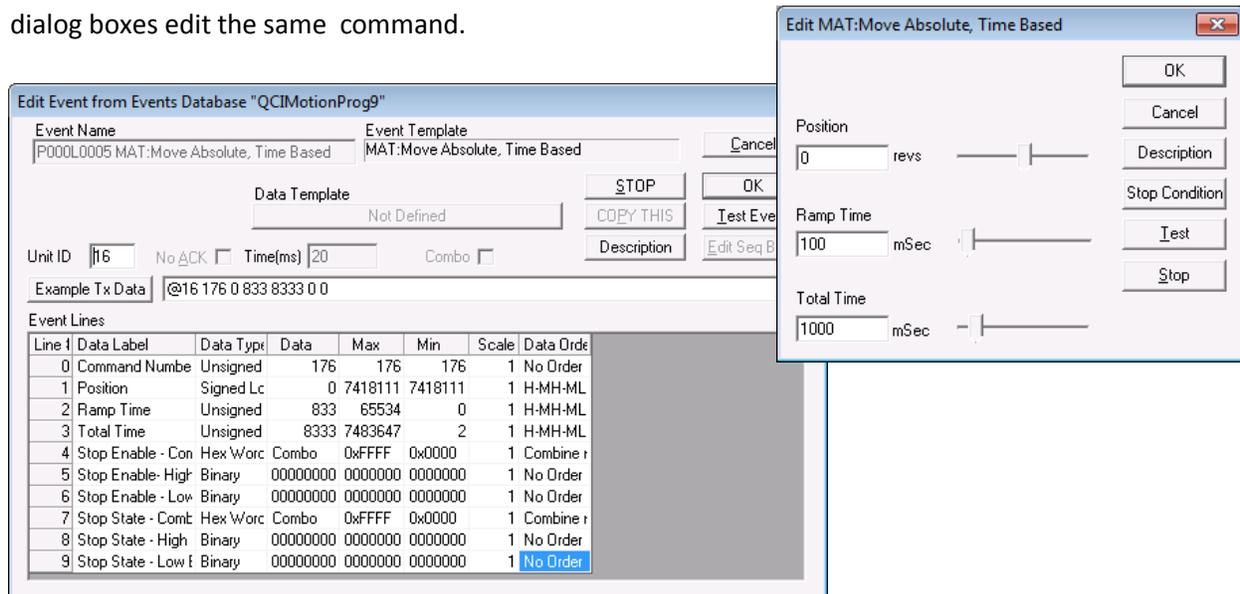
## Device State

The device can have the following states:

- ✓ Not Pol: Not polling (press Scan Network to start polling)
- ✓ Prog Stop: Polling active. No program running in device.
- ✓ Prog Stop-Kill: Polling active. No program running in device. Device had an error and is in a Kill Motor state (shutdown). See Technical Document “QCI-TD052 Shutdown and Recovery” for more details.
- ✓ Prog Stop-Soft Limit: Polling active. No program running in device. Device is limiting motion in response to the parameters set in an SSL or VLL command (see Command Reference for more details).
- ✓ Prog Run: Polling active: Program running in device.
- ✓ Prog Run-Kill: Polling active: Program running in device. Device had an error and is in a Kill Motor state (shutdown). See Technical Document “QCI-TD052 Shutdown and Recovery” for more details.
- ✓ Prog Run-Soft Limit: Polling active: Program running in device. Device is limiting motion in response to the parameters set in an SSL or VLL command (see Command Reference for more details).

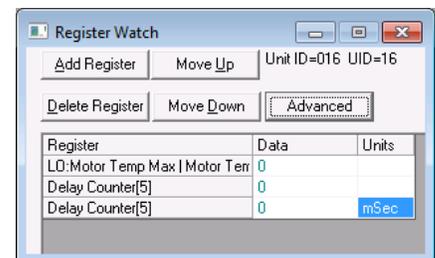
## Edit Details

Normally the detailed string used to send the command is hidden from the user. The Edit Details option displays a detailed edit dialog box for the selected command, showing the full string. The following two dialog boxes edit the same command.



## Register Watch

The Register Watch Tool is a powerful tool within QuickControl for monitoring and adjusting the contents of registers. This tool allows QuickControl to simulate a host, allowing an application developer to adjust register values while a program is running within the servo.

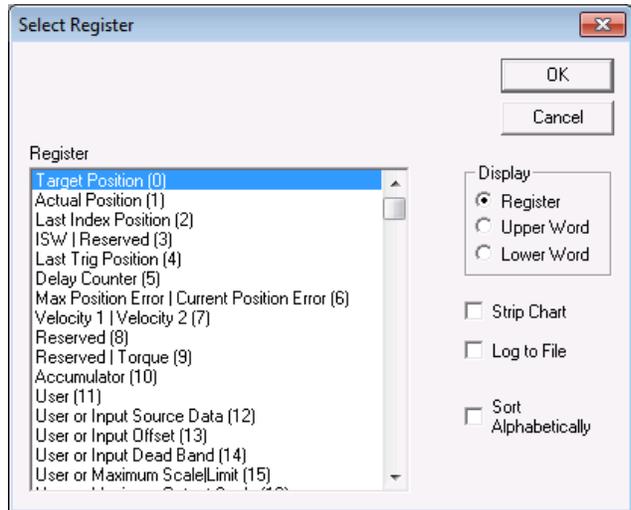


Open the Register Watch window from:

Tools->Register Watch

NOTE: Register Watch can also be launched by:

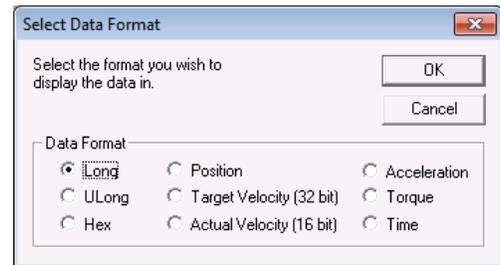
- Press the Device button on the Device Status Monitor
- Right Click on the device in the Device Status List.
- Double click on the device's 2<sup>nd</sup> row in the Device Status List.



Press Add Register to add a register to the list of “watches”. The Select Register dialog box allows you to select watching the whole register or only the upper or lower half.

Check Strip Chart to add a real-time strip chart and/or check Log to File to begin logging the data to a log file (log files found in Logs folder).

After selecting the channel, QuickControl will allow you to select the data format which enables you to watch the data in your engineering units.



Register Watch will watch the registers of the active device as selected in the Device Status Monitor. In Register Watch you are able to:

- ✓ Change the register being watched by double clicking in the register cell.
- ✓ Change the data by clicking in the data cell and entering new data. Note, although the data will be changed in the indicated register, it may get overwritten if the program is also modifying the same register. If the register selectable is not writable, the value will not stay changed on the display (and will not be changed in the attached unit), and a message will appear in the Status Log portion of the screen.
- ✓ Change the data format by double clicking on the units.

Right click on the register (row) to display a pop-up menu with many of edit features including:

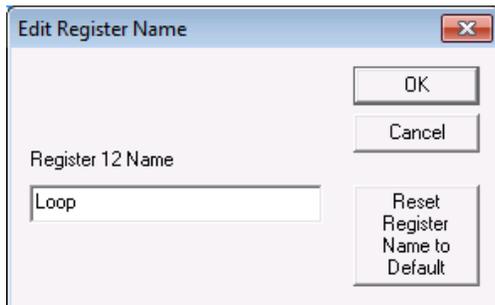
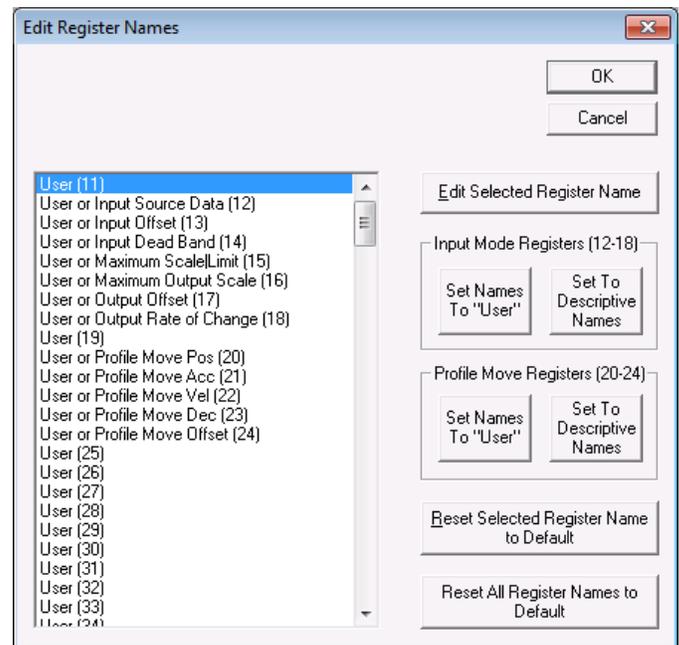
- ✓ Column Width: Manually set width of each column
- ✓ High Priority: When selected for a given register watch, QuickControl disables all other background communication including all other register watches and only polls the selected register. This allows for very fast updates on one register.

NOTE: The more registers you add to the list, the less frequent any single register will be updated.

## Register Names

QuickControl allows for editing the names of registers 11 through 199 (user registers). Any user registers used in a QCP file can be re-named to add more meaning to the program code. For example, registers being used for a specific function, like a loop counter can be named as such, just like the Input Mode registers and the Profile Move registers. If the QCP file does not use Profile Move or Input Mode functionality, these registers can be re-named.

Access the Register Names dialog window from the Programs pull down menu or from within the Scaling dialog window found in the Program Info Toolbar.



### Edit Selected Register Name

Select a register from the list box on the left and press Edit Selected Register Name button to re-name it.

Press Reset Register Name to Default to reset the name back to factory default.

### Input Mode Registers (12-18)

By default, these registers already have descriptive names because they are used in the PIM, VIM and TIM commands. If these commands are not being used, registers 12-18 can be used as general purpose user registers.

Press Set Name to User to set registers 12-18 to the name "User".

Press Set To Descriptive Names to set registers 12-18 back to their factory default names.

### Profile Move Registers (20-24)

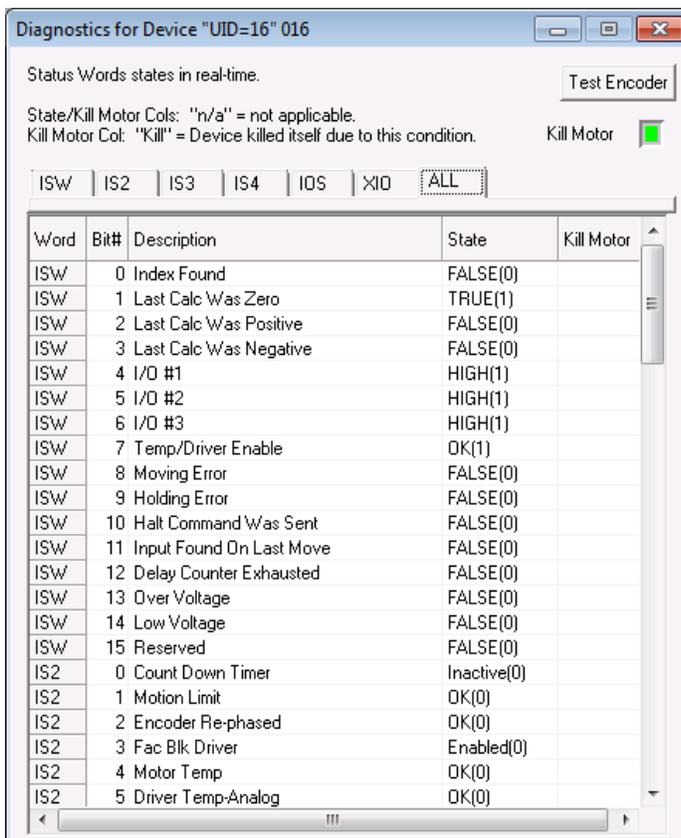
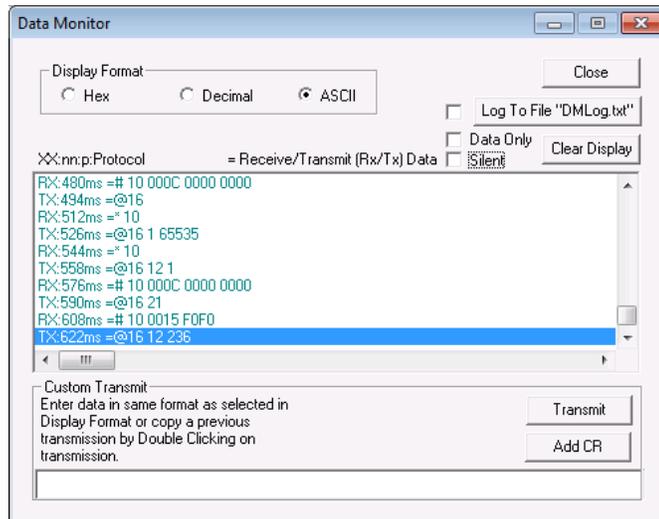
By default, these registers already have descriptive names because they are used in Profile Move commands (i.e. PMV and PMC). If these commands are not being used, registers 20-24 can be used as general purpose user registers.

Press Set Name to User to set registers 20-24 to the name "User".

Press Set To Descriptive Names to set registers 20-24 back to their factory default names.

## Data Monitor

The Data Monitor is a diagnostic tool that enables the user to view all transmitted and received data from all enabled ports as well as send custom packets out any single active Communication Port. The Data Monitor can be thought of as a serial or network analyzer specially designed for the QCI products. It provides the tools necessary to test your master controller's functionality. The fields are defined as follows:



## Diagnostics

The Diagnostics dialog box displays the real-time state of the current QuickControl program and the system it is controlling. Further information on all the commonly used status words can be found in chapter 3.

For those status words that are part of the KMX or KMC command, the Kill motor column will display "Kill" if the corresponding bit was the cause of the Kill Motor. The Kill Motor "LED" (top right of dialog box) will be red if a Kill Motor is in effect and green otherwise.

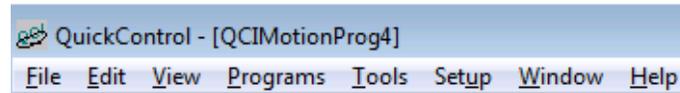
## Firmware Download Wizard

From time to time QCI adds new features to their products, which requires a new version of firmware to be downloaded. The latest firmware and instructions for downloading can be obtained from QCI Product Support. Please make sure to shut down other background programs that do file saves when downloading firmware. A failed firmware download will render the device repairable by the factory only.

# Appendix V: Shortcut Key

## QuickControl Menu Bar

The QuickControl Menu Bar provides access to all functions of the software. These “pull down” type menus offer many selections that can bring up other menus to accomplish certain objectives. Above the Menu Bar, next to the QCI logo & QuickControl name is the active filename being displayed in the QuickControl Program Window. See Menu Details below for more information.



### File Menu

- **New Program File** - Opens blank program template
- **New Sequence File** – Advanced Feature (see Technical Document QCI-TD025: Events and Sequences).
- **Open** – Opens a saved file
- **Close** – Closes the active file in QuickControl
- **Save** – Saves the current file
- **Save As** – Saves the current file with user set parameters
- **Program File Properties** – Contains scaling, password protection, and user defined names. This dialog box can also be accessed by pressing Scaling on the Program Info Toolbar. See File Menu Details below for more information.
- **Upload Program File** – Retrieves the current program residing in non-volatile memory. See File Menu Details below for more information.
- **Print** – Prints the active QuickControl Program file to the default printer
- **Print Setup** – Allows the user to change printer and paper settings
- **All Halt** – Sends the All Halt command
- **Recent Files** – Displays a list of files recently open in QuickControl
- **Exit** – Closes the current files opened and exits QuickControl

### Edit Menu

- **Cut** – Removes highlighted section of text or program line
- **Copy** – Copies highlighted section of text or program line
- **Paste** – Inserts a copied or cut section of text or program line before selected line
- **Select All** – Highlights an entire program
- **Events** – Advanced Feature (see Technical Document QCI-TD025: Events and Sequences).
- **Find** – Find (search for) text in program file.

### View Menu

- **Toolbar** – When checked, Toolbar will be displayed
- **Status Bar** – When checked, Status Bar will be displayed
- **Device Status** – When checked, Device Status will be displayed
- **Column Width** – Set the program's column widths

## Program Menu

- **Add Line** – Adds a new line to a program
- **Insert Line** – Inserts a line above the selected line in a program
- **Edit Line** – Edits the selected program line
- **Edit Line Details** - Edits the selected program line with all the command details.
- **Delete Line** – Deletes the selected line in a program
- **Disable Line** – Disable selected line in a program. The program line is "grayed out" and is not downloaded.
- **Enable Line** – Enabled a disabled line.
- **New Program** – Creates a new blank program
- **Delete Program** – Deletes selected program
- **Program Details** – Allows a user to name, describe, and modify the stored location of a program.
- **Scaling** – Allows a user to adjust scaling parameters and max/min ranges
- **Register Files** – Links register files or file arrays (.txt based) to active QuickControl program (see Register Files for more details).
- **Register Names** – Assigns user defined names to registers. See Programs Menu Details below for more information.
- **I/O Names** – Assigns user defined names to I/O lines. See I/O Names below for detail. See Programs Menu Details below for more information.
- **Download** – Download program file to device.
- **Download/Restart** - Download program file to device then restart device.
- **Download/Run** - Download program file to device then run first program without restarting.
- **Run w/o Download** - Download program to RAM then run.
- **Download and Chart** - Download program file to device then bring up chart. See Download and Chart below for more information. See Programs Menu Details below for more information.
- **Erase Application in Device** – Erases current program in non-volatile memory
- **Toggle Breakpoints** – Toggles Breakpoint at current line
- **Clear all Breakpoints** – Clears all Breakpoints in all programs
- **Single Step** - Single step program line. See Program Debugging for details.
- **Real-Time Trace** - Trace Program in Real-Time. See Program Debugging for details.
- **Jump To Line** – Jump program execution to selected line.

## Tools Menu

- **Initialization Wizard** – Sets up and initializes devices. See Tools Menu Details below for more information.
- **Unknown Device Wizard** – Establishes communications with a device of unknown parameters. See Tools Menu Details below for more information.
- **Control Panel** – Tool for Jogging, Tuning, and Monitoring SilverLode products. See Tools Menu Details below for more information.
- **Register Watch** – Change and/or view data in the SilverLode products data registers with this utility. See Tools Menu Details below for more information.
- **Data Monitor** – Monitor all data sent and received by a device and the PC. See Tools Menu Details below for more information.
- **Diagnostics** - Displays the real-time states of all the commonly used status words.
- **Firmware Download Wizard** – Downloads firmware to device. See Tools Menu Details below for more information.

## Setup Menu

- **Comm Port** – Selects Baud Rate, Communications port, and Protocol for QuickControl. See Setup Menu Details below for more information.
- **Register Devices** – Allows user to manually register devices into QuickControl. See Setup Menu Details below. See Setup Menu Details below for more information.
- **Options** – Allows user to edit other setup specifications. See Setup Menu Details below for more information.
- **Polling** – When checked, starts QuickControl polling the network for any connected devices and then displays the state of the device in the Device Status window

## Window Menu

- **New Window** – Creates a new program window
- **Cascade** – Cascades current program windows
- **Tile** – Displays current program windows in a tile arrangement
- **Arrange Icons** – Arranges and aligns minimized program icons
- **Current Program List** – A list of programs currently open in QuickControl

## Help Menu

- **Help Topics** – Opens help menu for tutorials and information on QuickControl
- **About QuickControl** – Displays date and version of QuickControl and product support information

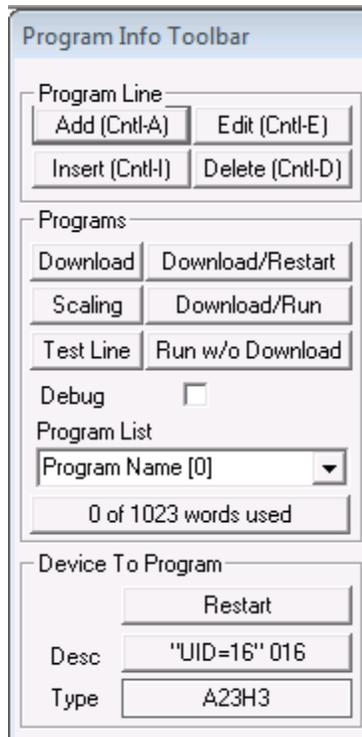
## Program Line Right Click Pop-Up Menu

Right clicking on the a program line will bring up a menu with the following items:

- **Add:** Add a new program line below this line.
- **Insert:** Insert a new program line above this line.
- **Edit:** Edit this program line.
- **Edit Details:** Edits the selected program line with all the command details.
- **Delete:** Delete this line.
- **Disable:** Disable this line in a program. The program line is "grayed out" and is not downloaded.
- **Enable:** Enabled a disabled line.
- **Expand/Collapse:** Expand or Collapse this Combo Command.
- **Test:** Executes this line. For example, if the line was a move command, the attached servo would move.
- **Column Width:** Set the column widths.
- **Toggle Breakpoint:** Add/Delete Breakpoint for this line.
- **Jump To Line:** Jump program execution to selected line.
- **Copy:** Copy this line.
- **Cut:** Cut this line.
- **Paste:** Paste a previously copied or cut line.

## Program Info Toolbar

The Program Info Toolbar is located on the left side of the main QuickControl Screen. This toolbar allows the user to create, edit, download, and debug programs. It also offers information about programs and the current device being programmed.



### Program Line

- **Add (Ctrl-A)** – Adds a line below selected line
- **Edit (Ctrl-E)** – Edits the selected line
- **Insert (Ctrl-I)** – Inserts a line above selected line
- **Delete (Ctrl-D)** – Deletes the selected line

### Programs

- **Download** – Downloads all programs in the active file to the device.
- **Download/Restart** – Downloads all programs in the active file to the device and commands the device to restart.
- **Download/Run** – Downloads all programs in the active file to the device and commands the device to load and run program 0
- **Run w/o Download** – Runs first program in active file (program 0) but does not download anything to non-volatile memory.
- **Scaling** – Shortcut to Program File Properties.
- **Test Line** – Executes selected line.
- **Debug** – Opens Debug Toolbar (see Program Debugging later in this manual for details).

## Program List

The list of selectable programs currently open in the active QuickControl file (QCP)

## No. of words used window

Shows how many words are being used in the currently displayed program

## Device to Program

- **Restart** – Restarts the currently selected device
- **Desc** – Allows user to select the device being programmed and displays the registered label and ID number
- **Type** – Displays type of device currently being programmed



## QuickControl Utilities

This exercise demonstrates two of the most widely used utilities within QuickControl in an effort to show their usefulness in prototyping as well as troubleshooting. Upon completion of this exercise, an understanding of how QuickControl interfaces with the device should be developed. <CR> is used to represent the carriage return in these exercises. Since the carriage return does not have a viewable character it is displayed as a vertical bar “|” when ASCII strings are viewed in QuickControl.

1. Initialize the device and press the **Start Polling** button to ensure functioning communication.
2. Click on **Data Monitor** in the **Tools** pull down menu and view the polling routine that QuickControl uses to report all information about the device. Note that the device never initiates outward communication. Therefore, every piece of information displayed in QuickControl has to be polled out of the device by this routine.
3. Check the **Data Only** box. With this box checked, the Data Monitor only displays the command strings that QuickControl sends to the device and the device response to those commands. With this box unchecked, each data string is preceded by TX/RX indicator (TX for QuickControl transmissions and RX for the device responses) and a recycling clock time. The purpose of the clock is to tell how much time has passed in between a QuickControl TX and the device RX.
  - The TX and RX are redundant because any transmission to the device is indicated with an @ symbol followed by the ID of the unit (e.g. @16).
  - the device responses are preceded with either a “\*”, “#”, or “!” character (See Technical Document QCI-TD053 Serial Communications on our website for a discussion on the individual meanings of each).
4. Check the **Silent** box to stop the polling routine from streaming in the Data Monitor and scroll up in the window to view the routine in detail; or, check the **Log to File** box and press the **Log to File** button to select a specific location and text file to log this routine to.
5. Note the commands used in the basic polling routine (details may be looked up by command number in the Command Reference if desired) and the data registers that are queried.
  - @16 12 1<CR>: This Read Register (RRG) command queries the Actual Position[1] register.
  - Actual Position is displayed in the Device Status Monitor and is updated due to this line of the polling routine.
6. Click on **Register Watch** in the **Tools** menu and **Add Register** when the Register Watch utility appears. Choose Accumulator[10] (or a register that is not being polled by the basic polling routine) and select long format to view values as signed decimal numbers.
  - Uncheck the **Silent** box in the Data Monitor and note that Accumulator[10] is now being queried by the polling routine.
7. Type a “50” into the data box of the added register and press enter. Quickly check the **Silent** box and scroll up in the Data Monitor log to view the transmission.
  - @16 11 10 50<CR>: This Write Register command writes a 50 into the Accumulator[10] register.
  - This string was issued by QuickControl as soon as data was typed into Register Watch and **Enter** was pressed.
8. Move to the Custom Transmit section of the Data Monitor. Type “@16 11 10 100|” in place of “Test Packet.” Add the final carriage return character with the **Add CR** button, and press **Transmit**.
  - Notice that Register Watch is immediately updated by the polling routine.
  - Note how the Data Monitor and Register Watch tools can be used to emulate a PLC, HMI, or other host that sends data serially to the device.

## Appendix VI Techniques for Stopping Motion

When using dynamic systems to control complex motion, provisions for stopping movement and exiting those operations must be addressed. The servo has the functionality to accomplish this through either software or through hardware (not available on all models) via the Drive Enable option.

### Software Stop Options

The commands described below are the software stop options available to stop the servo's motion. All of these options are part of the Command Set and can be used as interrupts to stop motion as described below. Some commands initiate immediate stops while others allow for a predetermined deceleration profile. Immediate stops use maximum acceleration available (and consequentially maximum current available) to stop the servo as quickly as possible; immediate stop options may re-generate a significant amount of power back into the power rail that could damage the driver circuitry or motor windings of the servo. Technical Document QCI-TD0006 contains more information regarding QCI voltage clamp modules that safely dissipate the re-generated power. (Note SilverDust IG and IGB have onboard clamp circuits; only use with load resistor attached!) When designing a stop for the motion, consider the move velocity, load inertia, available back EMF protection, and importance of the stop condition.

The **Stop (STP)** command not only stops the present motion, but also exits the current program and puts the servo into a holding state once stopped. The STP command can be formatted to use a specific deceleration, use the current move's acceleration parameter to decelerate to a stop, or use maximum deceleration to achieve an immediate stop where the target position is set to the present position.



The red stop hand button in the QuickControl Icon Bar issues an all stop command. It sends the **Stop (STP)** command to ID 255 and all the servos connected to the PC will stop.

The **Halt (HLT)** command stops the execution of any command, program, and motion in progress. It also disables the servo drive, which allows the shaft to spin freely and starts the Kill Motor Recovery routine where the recovery method can be programmed (see Technical Document QCI-TD052 Shutdown And Recovery on our website).

The **Hard Stop Move (HSM)** command provides a means to execute a hard stop while multi-tasking. A hard stop immediately disables the Trajectory Generator stopping any movement and exiting any move operation (e.g. Step & Direction). This causes an abrupt stop, which in many cases will cause the servo to overshoot the stop position and oscillate until settled.

More controlled stops can be accomplished by using the **Velocity Mode (VMP or VMI)** command, which allows a predetermined deceleration profile to stop the servo. Use this command to specify a deceleration to zero velocity, stopping the servo.

The **Profile Move Exit (PMX)** works much like the VMP stop and will be discussed in the Profile Move section of this chapter. With this syntax it is unnecessary to specify a velocity. PMX uses the deceleration register for Profiled Moves (Register #23) to slow to a stop.

Soft Stop Limits (SSL) can also be used to stop any type of move. This command defines two data registers as end of travel limits. Once values are stored in these registers, any motion affecting the

target is limited to keep the target position more than the first register value and less than the second. For SilverDust and SilverSterling controllers, the limits are applied to any calculated motion command, producing normal ramping to limit at the target. Other motions, including camming and velocity moves do not produce a ramping to stopped at the limit (nor do SilverNugget controllers). Motion that exceeds a limit (except as already described) is hard stopped at the point that the limit is encountered so no ramping occurs. A status bit is set (allowing Kill Motor to respond if desired), but the active command and/or motion are not affected. This allows motions including VIM,TIM,PIM to be limited without exiting their operation.

### Hardware Stop: Drive Enable feature

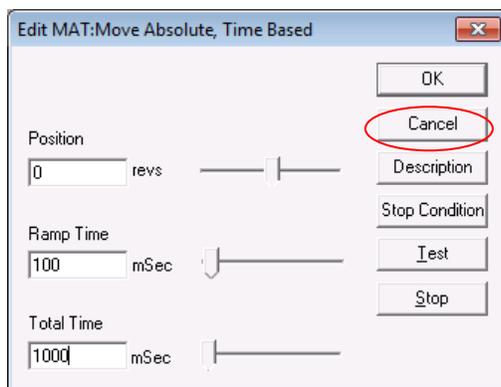
A hardware driver enable/disable feature is available as an additional option for many SilverLode servos, but is standard on those servos with separate motor winding power (i.e. SilverLode products designed for 34 frame motors). It is also standard on the SilverDust IG and IGB. This feature allows the motor driver circuit to be disabled via a solid-state switch. Disabling the motor driver cuts off torque, while the DSP control electronics remain active to track position and respond to commands. With the absence of torque, the load inertia may cause the servo to coast to into a hard stop; deceleration may not be as fast as with a commanded motion.

## Using Inputs to Stop Motion

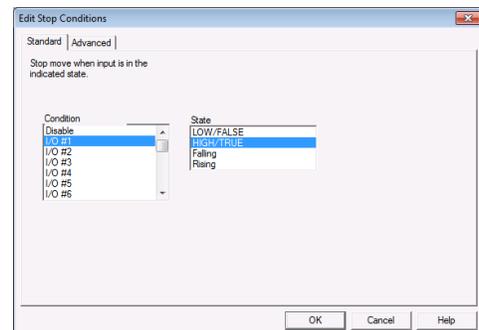
All basic motion commands, as well as the input mode and velocity mode commands, have integrated stop conditions. The stop conditions include the digital inputs, as well as several other internal signals and states, which may be selected within QuickControl. Digital inputs may be wired to sensors, switches, or digital I/O from a PLC or other host. Whatever the hardware connected, the servo's I/O can be used either alone or in conjunction to affect the operation of motion commands.

### Standard Stop Conditions - QuickControl

Most moves have a dialog box that looks something like the Edit MAT dialog box pictured here. Even if the dialog box looks a little different for a particular move command, the Stop conditions can be accessed by pressing the “Stop Condition ” button.



This brings up the standard “Edit Stop Conditions” dialog box, which presents a simple interface displaying the available inputs and conditions.



## Standard Stop Conditions – Serial Communications

The same stop conditions presented within QuickControl are available when sending commands from a host.

### *Stop Enable*

This move parameter is Stop Enable and is the same as the Jump Command Enable Code parameter with the exception of 0 which means "Do Not Check for Input".

### *Stop State*

The stopped state selected can be either level or edge sensitive. This allows stopping on high states, low states, on the transition from a high to low or a transition from low to high. Stop states are setup using the following parameters:

Stop State	Stop on the Following Condition
0	FALSE
1	TRUE
2	FALLING (TRUE to FALSE Transition)
3	RISING (FALSE to TRUE Transition)

For example, to issue a Velocity Mode, Immediate (VMI) command that will stop when I/O #7 is high, the following string would be issued.

**@16 15 200000 100000000 -7 1<CR>**

See Command Reference for details on the VMI command.

## Advanced Stop Conditions

For details on Advanced Stop Conditions, see Technical Document QCI-TD039: Move Command Stop Conditions - Advanced.

## Appendix VII Encoder Outputs

Many of the SilverLode controllers can output their internal encoder signals.. The main reason to use this feature is for an electronic gearing, or following application (see Application Note “QCI-AN019 Electronic Gearing”). This feature is useful for multi-axis systems that must move in unison (the two servos would not be truly synchronized, however, because of the unavoidable processing lag between the lead and following unit, so high-precision systems may need to be coordinated with an external controller like a PLC). The raw internal encoder output function is just that: a buffered copy of the A and B quadrature signal that the servo uses for internal control purposes.

### Internal Encoder Output (SilverNugget Only)

A SilverNugget servo can output its raw internal encoder signal through specific I/O lines. This signal is the same A and B quadrature and index pulse signal that the servo uses for internal control and motor commutation purposes. The A signal is output on I/O line 1, the B signal on I/O line 2, and the index signal on I/O line 3. The command needed to send the raw internal encoder signal to the I/O lines: Enable Encoder Monitor (EEM). The Disable Encoder Monitor (DEM) returns the I/O back to normal usage. I/O lines 1 to 3 must be configured as inputs before the EEM function is used in order to avoid an error.

### Internal Encoder Output (SilverDust-IGB Only)

The encoder signals are available on 3 dedicated terminal blocks: Encoder outputs A, B, Z. These are TTL buffered outputs. Note: the Z-channel of I-Grade motors is a special index channel. The output is a 50% duty cycle spaced at one cycle for 1/50 revolution, with one “tooth” missing. QuickSilver documentation refers to this as a 49/50 index channel.

### Internal Encoder Output (SilverDust-IG8 Only)

The internal encoder signals may be configured to drive either IO4, IO5, and Optionally IO6 with Encoder outputs A, B, Z using the Encoder Monitor (EMN) command. Alternatively, the SSI port may be configured to output differential A, B, Z signals from the internal encoder using the SSI command. Note: the Z-channel of I-Grade motors is a special index channel. The output is a 50% duty cycle spaced at one cycle for 1/50 revolution, with one “tooth” missing. QuickSilver documentation refers to this as a 49/50 index channel.

### Scaled Internal Encoder Output (Modulo Output)(SilverNugget Only)

In addition to the raw encoder signal, the SilverNugget can output a scaled version of its internal encoder signal with the modulo output function. This function can also be used to output an external encoder signal if required. A SilverNugget can only scale the modulo output signal down. The modulo output function is essential for synchronized multi-axis applications since it allows the master to output its encoder signal for the other servos to follow.

The modulo encoder output function can use all three high speed I/O function signal formats: step and direction, step up/step down, and A & B quadrature. The output signal is scaled using the modulo scaling parameter, which can be set to any integer value between 1 and 256. (1 to 32 for SilverNugget N3.) The output signal is different for the three signal formats. For the step and direction and step up/step down signal formats with the scaling parameter set to “1”, the 4000 counts per revolution internal encoder signal is output as a 2000 pulses per revolution signal. For A & B quadrature format

and a “1” scaling parameter, 4000 counts per revolution from the internal encoder is output as a 1000 pulses per revolution. If an A & B signal is sent to another servo, the decoding circuitry in the second servo will turn the 1000 pulses per revolution back into a 4000 counts per revolution signal. A scaling parameter other than “1” will scale the modulo output signal down by the scaling factor: e.g. for an A & B quadrature output with a scaling parameter of “4”, 4000 counts from the internal encoder would be scaled to 250 pulses (which decodes to 1000 counts).

Three commands are used with the modulo internal encoder output function. The Modulo Set (MDS) command enables the modulo output function and starts the signal from I/O lines 6 and 7, the lines that the modulo output function uses. The MDS command sets the modulo divisor (1 to 256), the signal type (step and direction, step up/step down, or A and B quadrature), and the encoder source (internal or external). The Modulo Trigger (MDT) command enables a special modulo function that uses the state of I/O #1 as a trigger to start and stop the modulo internal encoder output signal on I/O lines 6 and 7. The Modulo Clear (MDC) command disables the modulo output function and frees the I/O lines used by the modulo output function. More information on these commands is available in the Command Reference.

## Synchronous Serial Interface (SilverDust IG8 only)

The SSI port is the 10 pin connector on the top of the SilverDust IG8 Package next to the CAN Switches. It provides a total of 4 differential interfaces as well as +5v and ground. The I/O are RS485 signal levels with a 3.3v drive level. These IO may be variously configured to provide different functions via the SSI command.

SSI Mode	Description	1	2	3	4	5	6	7	8	9	10
0	Input External Encoder*	A- (IN)	A+ (IN)	B- (IN)	B+ (IN)	GND	+5v	Z- (IN)	Z+ (IN)	N.C. (IN)	N.C. (IN)
1	Output Internal Encoder	A- (OUT)	A+ (OUT)	B- (OUT)	B+ (OUT)	GND	+5v	Z- (OUT)	Z+ (OUT)	N.C. (IN)	N.C. (IN)
2	SSI position to Reg 253	Data- (OUT)	Data+ (OUT)	Clock- (OUT)	Clock+ (OUT)	GND	+5v	Select- (OUT)	Select+ (OUT)	Data- (IN)	Data+ (IN)
3	SSI position to Reg 253, Dual Loop	Data- (OUT)	Data+ (OUT)	Clock- (OUT)	Clock+ (OUT)	GND	+5v	Select- (OUT)	Select+ (OUT)	Data- (IN)	Data+ (IN)

\* Must also configure SEE command to select External Encoder onto IO 4,5,6

(IN) = 485 differential input, (OUT) = 485 differential output, N.C = No Connect

Mode 0 is the default power up mode. The SSI port is placed with all sections configured as inputs. Inputs A,B, and Z may be directed to IO 4,5, and 6 by means of the Select External Encoder (SEE) command. This provides a means to interface an external differential encoder or Step and Direction source to the Unit. IO 4,5, and 6 may not be used for other purposes if so configured.

Mode 1 configures A,B, and Z as outputs, and outputs the raw internal encoder signals to the SSI port, allowing for differential signaling of these signals to other units.

Mode 2 configures the SSI port to send and receive serial data. Every 120us the chip select is asserted and data may be sent and received, clocked by the serial clock signal, running as a burst of 1.5MHz

pulses. The number of data bits transferred is selected via the SSI command from 8 to 32 bits. (Note: For data > 16 bits, there may be a slight pause between the first 16 bits transferred and the remaining.).

Data OUT (MOSI by SPI specifications) takes data from CAN object 2009h, sending up to 32 LEFT JUSTIFIED bits to the SSI port, as clocked by Clock. Data IN (MISO by SPI specs) takes data from the SSI port and deposits it right justified into register 253. Upper unused data bits are masked to zero. No wrap-around is provided for absolute encoders operated past their limits.

Mode 3 configures the SSI port the same as does Mode 2, except that the SSI port is flagged as the source for dual loop data if the DLC command is activated following this selection. The dual loop source may not be changed if dual loop is active.