

Rugged Servo Drives & Control Systems for Extreme Environments



Controller User's Manual

Document Number: 100266-00K

Date of Doc Rev: 12/2/2020



Notice

This user's manual contains proprietary information belonging to ESI Motion.

The information provided is solely for the purpose of assisting users of ESI Motion's Servo Drives and Modules.

Information supplied in this manual is subject to change without notice.

Revision History

Version	Date	Changed by	Items Changed
A	1/15/2015		Initial Release
B	10/23/2015		Updated to coincide with the Trouble-shooting guide release and to add new Controller features.
C	4/11/2016		Made CAN and Serial configuration variable names common. Updated Manual Feedback with the simplified descriptions. Added section on USB Driver installation. Added Digital In and Digital Out sections. Deleted the Auto Phase section. Added section on verifying current loop tuning Added configurable loop input sections, Updated control loop diagrams in appendixes.
D	9/9/2016		Maintenance updates; added updated test script logic.
E	10/14/2016		Added Digital In/Out examples; simplified loop-input source-selections
F	1/31/2017		Clarified valid secondary-inputs.
G	11/29/18	K. Morgan/T. Ahmad	Updated Style Guide, some drawings and text
H	6/10/2019	J. Jollota	Clarifications from customer feedback, added control-loop configuration examples section.
H	6/10/2019	K. Morgan	Updated image, fixed layout and typos & ToC
J	7/8/2020	J. Jollota	Corrected pole-pairs calculation, added motor-control examples, added back in all common-controller variable descriptions.
K	12/2/2020	J. Jollota	Updated for Atom and controller quadrature/BISS encoder updates.

Important Information:

ESI MOTION makes no warranty, either express or implied, including but not limited to any implied warranties of merchantability and fitness for a particular purpose, regarding any marketing materials and makes such materials available solely on an "as-is" basis. In no event shall ESI MOTION be liable to anyone for special, collateral, incidental, or consequential damages in connection with or arising out of the purchase or use of these materials, and the sole and exclusive liability of ESI MOTION, regardless of the form of action, shall not exceed the purchase price of this product. Moreover, ESI MOTION shall not be liable for any claim of any kind whatsoever against the use of these materials by any other party.

Table of Contents

1	INTRODUCTION	9
1.1	Getting Started	9
2	NAMING CONVENTIONS	9
3	THE HIDS APPLICATION (SUMMARY)	10
4	NEW MOTOR TUNING / CONFIGURATION	10
4.1	Voltage / Current Limits	11
4.2	Motor Parameters	11
4.3	Current-Loop Tuning	11
4.3.1	Common Setup	11
4.3.2	Step Response Procedure	12
4.3.3	Setting the Current-Loop Integral	14
4.4	Feedback Phasing / Calibration	14
4.5	Velocity-Loop Tuning	15
4.5.1	Unconstrained motors	15
4.5.2	Constrained motors	16
4.5.3	Setting the Velocity-Loop Integral	17
4.6	Position-Loop Tuning	17
4.7	Sensorless Configuration	19
4.7.1	Sensorless Velocity-Loop Tuning	19
5	THE HIDS APPLICATION (DETAILS)	19
5.1	Connection to a Controller	20
5.1.1	Connecting with CAN	20
5.1.2	Connecting with USB	20
5.2	The HiDS User Interface	20
5.2.1	Editing Parameter Values	21
5.2.2	Summary Tab	21
5.2.3	Motor Tabs	21
5.2.4	Loop Gains Tab	22
5.2.5	Limits Tab	23
5.2.6	Importing, Exporting and Saving Settings	23
5.2.7	Advanced Tab	24
5.2.8	Firmware Upgrades	24
5.3	Advanced Device Management	25

5.3.1	Page List	25
5.3.2	Data Objects List	25
5.3.3	Data Objects Storage.....	26
5.3.4	Data Objects Access.....	26
5.3.5	Data Objects Filtering	26
5.3.6	Searching for Objects	27
5.3.7	User-Defined Object Page.....	27
5.3.8	Editing Object Values	27
5.3.9	Digital Test Points	28
5.3.10	Analog Test Points.....	28
5.3.11	Digital In	28
5.3.12	Digital Out	28
5.3.13	Serial Test.....	28
5.4	Analog Test Points.....	28
5.4.1	Assigning Data Objects to Analog Test Points.....	28
5.5	Digital In	29
5.5.1	Assigning Data Objects to Digital In.....	29
5.6	Digital Out	30
5.6.1	Assigning Data Objects to Digital Out	30
5.7	Run Panel.....	31
5.7.1	Run Panel Customization	31
5.8	Scope / Digital Test Points.....	32
5.8.1	Assigning Data Objects to Digital Test Points.....	32
5.8.2	Waveform Display.....	33
5.8.3	Channel Selection	33
5.8.4	Trigger Settings	34
5.8.5	Channel Parameters	35
5.8.6	Cursors and Capture Interval	36
5.8.7	Saving Captured Data.....	36
5.9	Settings	36
5.9.1	Savings Values.....	36
5.9.2	Comparing Current Settings with a Text File	36
5.10	About HiDS.....	37
5.11	Simulator	37
5.11.1	Simulator Customization.....	37
5.12	Run Test Script.....	37
6	MOTOR FEEDBACK.....	41

6.1	Resolver	41
6.2	Quadrature Encoder.....	41
6.3	Manual Feedback.....	41
6.4	Sensorless Feedback	42
6.5	Hall Feedback.....	42
6.6	Serial Encoder Feedback.....	42
6.7	Sin/Cos Feedback.....	42
6.8	Endat Feedback	43
6.9	Secondary Feedback	43
7	CONTROL LOOP INPUTS.....	43
7.1	Sine Test Command.....	44
7.2	User Command.....	44
7.3	CAN Command.....	44
7.4	Serial Command.....	45
7.5	Resolver Commands	45
7.6	Analog Commands.....	45
7.7	Loop Filters	46
7.8	Configuration Examples.....	46
7.8.1	HiDS Control.....	46
7.8.2	RS422 Control	47
7.8.3	CAN Control.....	48
8	MOTOR CONTROL EXAMPLES	49
8.1	Analog Velocity Control with a Digital Motor Enable Input	49
8.1.1	Connection / Configuration Verification	50
8.2	Digital Input Control	51
8.3	RS422 Control	51
8.4	CAN Control.....	51
9	THEORY OF OPERATION.....	52
9.1	The Current Loop	52
9.1.1	Clark and Park Transforms.....	52
9.2	Clark and Park Transforms.....	53
9.2.1	IQ Command Determination	54
9.2.2	IQ and ID Error Compensation.....	57

9.2.3	The Inverse Clark and Park	58
9.3	The Velocity Loop	59
9.3.1	RPM Command Determination.....	59
9.3.2	RPM Error Compensation	60
9.4	The Position Loop	60
9.4.1	Radians Command Determination	61
9.5	Manual Feedback	61
10	APPENDIX A: HIDS VARIABLE GLOSSARY	62
10.1	Summary.....	63
10.2	Analog.....	63
10.3	BIT (Built In Test)	66
10.4	CAL	78
10.5	CAN	78
10.6	Compensation	80
10.7	Config.....	85
10.8	Control	85
10.9	Digital IO.....	87
10.10	Encoder	88
10.11	Endat Encoder	91
10.12	Fan	92
10.13	Fault History	93
10.14	Fault Inputs.....	97
10.15	FPGA (Hyperion Only)	105
10.16	Hall	106
10.17	Inrush	107
10.18	Limits.....	108
10.19	LoopInputs.....	109
10.20	Manual Feedback	113
10.21	Motor AHSL, MotorBHSL	115
10.22	MotorParameters	122
10.23	SecondaryFB	123
10.24	Position	124

10.25 Power	127
10.26 Regen	128
10.27 Resolver	129
10.28 Sensorless	132
10.29 System.....	134
10.30 Serial.....	134
10.31 Serial Encoder.....	136
10.32 TestPoint.....	139
10.33 Temperature.....	139
10.34 Utility.....	140
10.35 Velocity Loop.....	143
10.36 Hardware Test	147
10.37 User Defined.....	147
11 APPENDIX B: THE ESI MOTION CURRENT LOOP	148
12 APPENDIX C: THE ESI MOTION VELOCITY LOOP	149
13 APPENDIX D: THE ESI MOTION POSITION LOOP.....	150

1 Introduction

Thank you for choosing ESI Motion's servo drive products. Our control systems offer a complete ruggedized solution which includes ESI's rugged controller and power driver boards, an integrated EMI filter, military-grade submersible case, controller software and user-friendly GUI with built in oscilloscope feature.

This User's Manual covers most of our controllers including the Dragon line which is the core of ESI Motion's integrated plug and play control solution. In addition, the Mite, Scorpion, and Atom Module lines are ideal for military, aviation, automotive or other heavy industrial applications operating in outdoor, high temperature, high vibration, or other extreme environmental conditions. Please refer to ESIMotion.com for further information about these and all other ESI Motion products or reach out to us at sales@esimotion.com.

1.1 Getting Started

This area points to the specific sections of the manual required to assist the user in connecting and running a motor using the HiDS Run Panel.

1. Install the ESI Motion HiDS PC application, which is available from the Downloads section of www.esimotion.com.
2. Connect the motor under test and apply logic power to the Controller; refer to the Controller Installation Manual and datasheet for interface and installation details.
 - a. If a high-voltage (motor-voltage) connection is required, connect the high-voltage source to the Controller, but if possible (if adjustable), reduce the voltage to ~10% of what is required for motor operation.
3. Connect to the Controller via HiDS and open the Run Panel; refer to Connecting to a Controller in section 5.1 and the Run Panel in section 5.7. Note the CAN interface to HiDS is recommended for all motor-control applications.
4. Tune the current-loop; refer to New Motor Tuning / Configuration in section 4.
5. Select your desired feedback and rotate the motor shaft manually. You should see the <feedback>Degrees/Radians value change accordingly; note <feedback> is Resolver, Hall, SerialEncoder, etc.
6. If not provided by the motor manufacturer, calibrate the feedback-position to the motor rotor-position. Refer to the Motor Phasing / Calibration in section 4.4.
7. Attempt to spin the motor using torque(current)-mode and using the desired feedback; refer to Motor Feedback in section 6. Note there is no velocity-control here, so be careful not to over-speed the motor. This verifies all motor-control features except velocity-loop and position-loop tuning, if required.
8. If required, tune the velocity and position loops; refer to Velocity-Loop tuning in section 4.5.

2 Naming Conventions

The descriptions below are applicable to each motor controlled, and because of the number of configuration and run-time variables, there is a readable prefix for common variables:

- There are common configuration and measurement variables, and for dual/multiple-motor controllers, there are configuration and measurement variables for each motor-control subsystem. Each Motor-control subsystem is

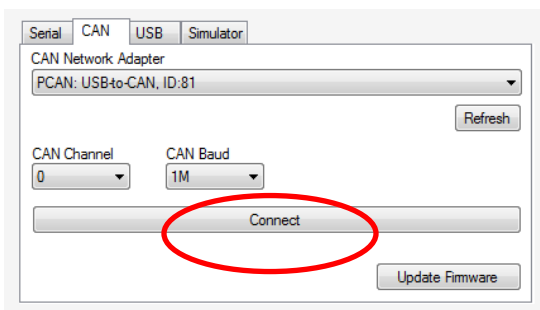
referred to MotorA (primary / single-motor controller), MotorB (for a dual / multiple motor-controller), and in some custom applications MotorC and MotorD (for the third and fourth motor-controller subsystem).

- The prefix for Current loop variables for Motor A, Motor B, and Motor C is Ma, Mb, and Mc, respectively. For example the Motor-A IQ command variable is Ma.IqCmd, and the Motor-B Minimum Current Command is Mb.MinCurrentCommand. Each of these variables is referred to below as Mx.VariableName
- The prefix for Velocity loop variables for Motor A, Motor B, and Motor C is MaVL, MbVL, and McVL, respectively. For example, the Motor-A user velocity command is MaVL.RPMUserCommand, and the Motor-B RPM error to compensate is MbVL.RPMError. Each of these variables is referred to below as MxVL.VariableName.
- The prefix for Motor configuration variables for Motor A, Motor B, and Motor C is MotorA, MotorB, and MotorC, respectively. For example, the Motor-A inductance is MotorA.Inductance, and the Motor-B feedback type is MotorB.FeedbackType. Each of these variables is referred to below as MotorX.VariableName.

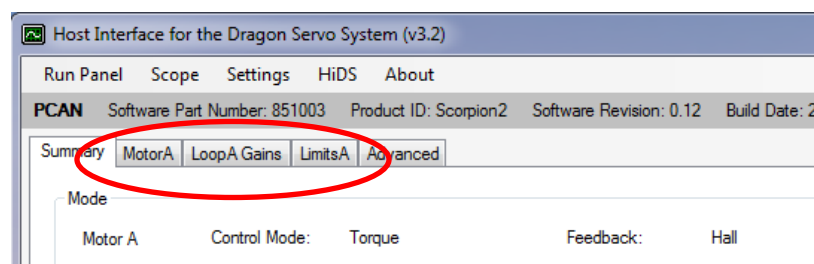
3 The HiDS Application (Summary)

The detailed screen, functions, and variable descriptions are described in sections 5 and 10, but for the basic HiDS connection to a Controller. See section 5.1.1 for the USB-to-CAN devices supported.

1. Execute the HiDS application, select the CAN-tab, and for off-the-shelf (non-customized) controllers, select the default CAN configuration, and click [Connect]



2. The basic configuration items are all located on the MotorA, LoopA Gains, and LimitsA tabs.



4 New Motor Tuning / Configuration

The ESI Line of modular motion control employs an industry standard current-loop, velocity-loop, and in some applications a position-loop. Each of these control loops utilizes proportional, integral, and derivative (PID) error correction to achieve the desired performance.

This section describes the procedure for tuning each control loop to match the intended application. After the tuning is completed, for applications including physical feedback (resolver or encoder) or for sensorless operation, setting the initial motor phase angle is described.

4.1 Voltage / Current Limits

Prior to controlling the motor in any way, the motor limits should be programmed into the ESI Controller.

1. Limit the user-current command (if in Torque mode) and the absolute currents applied to the motor:
 - a) In the HiDS **Limits** tab, set the **Positive Current Command** to the specified motor maximum continuous current rating.
 - b) If the negative current limit is not the same as the positive limit, in the HiDS **Limits** tab, set the **Negative Current Command** to the negative (-1 times) the specified motor maximum continuous current rating.
2. Limit the motor speed; the ESI Controller can be set to disable the motor should the motor RPM reach a maximum level.
 - a) In the HiDS **Limit** tab, set the **Maximum Velocity** to the specified motor maximum speed.
3. Set the Inrush (precharge) voltage, if necessary. For the initial motor configuration, the motor/high-voltage will be set to 10-20% of the nominal operating voltage. Set the **Precharge Voltage** (on the LimitsA tab) to ~10 volts less than the high-voltage setting. Note you may have to adjust the **Maximum Bus Voltage** and the **Minimum Bus Voltage** (all on the LimitsA tab) to be scaled down for the reduced power-supply setting.

4.2 Motor Parameters

The Controller needs to be configured with some fundamental motor properties in order to properly drive the motor. All of these parameters should be available from the motor manufacturer's data sheet.

1. Inductance: In the HiDS **Motor** tab, set the **Line-to-Neutral Inductance** to $\frac{1}{2}$ of the measured or specified motor Line-to-Line inductance.
2. Resistance: In the HiDS **Motor** tab, set the **Line-to-Neutral Resistance** to $\frac{1}{2}$ of the measured or specified motor Line-to-Line resistance.
3. Back-EMF Constant K_e : In the HiDS **Motor** tab, set the **Line-to-Neutral Voltage Constant** to the motor K_e . Note the ESI vector controller uses K_e units of Volts-peak/RPM Line to Neutral, so some unit-conversion may be necessary if the motor-manufacturer specifies different units. To convert the Line-to-Neutral to Line-to-Line, $K_{eL-L} = K_{eL-N} * \sqrt{3}$.
4. Motor Pole Pairs: In the HiDS **Motor** tab, set the **Motor Pole Pairs** to the specified motor pole pairs ($\frac{1}{2}$ times the number of motor poles/magnets).

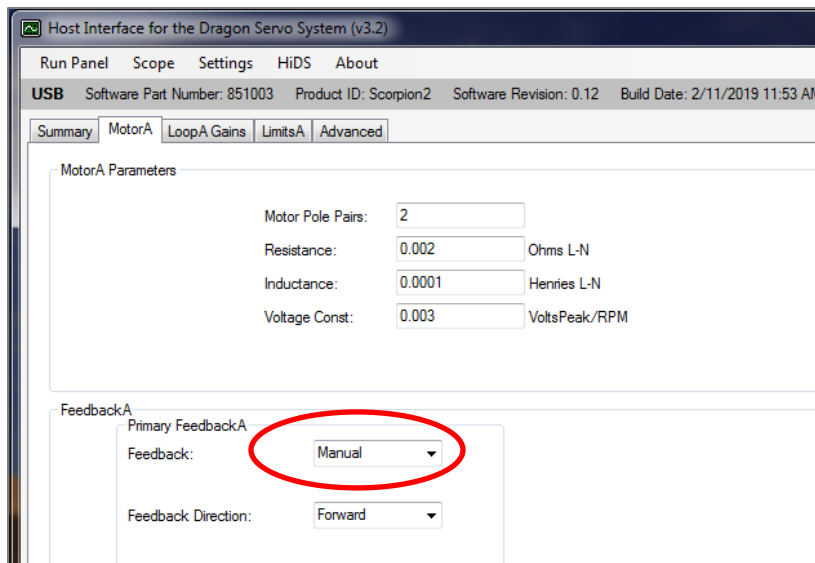
4.3 Current-Loop Tuning

4.3.1 Common Setup

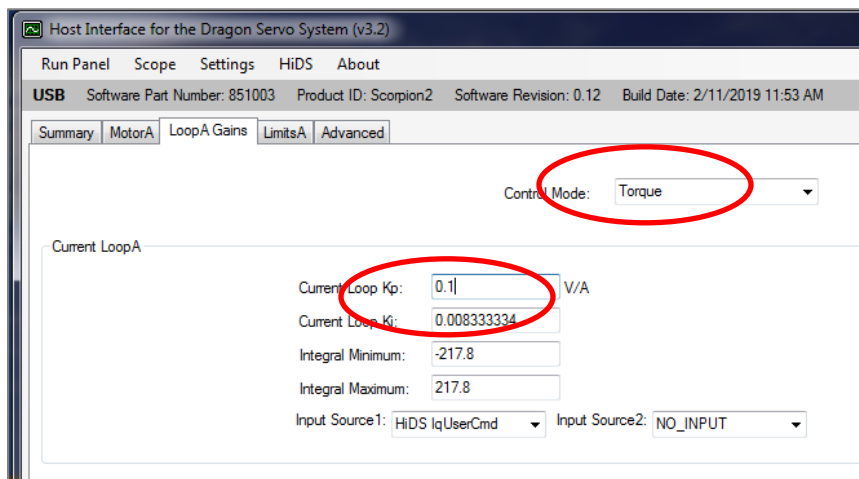
These steps are common to each of the three methods below.

1. Login to the Controller using HiDS. For HiDS login and installation information, refer to section 5.1.

2. Enable the manual response feedback, which disables the physical motor velocity feedback (resolver, encoder, etc), and runs the controller in open-loop mode:
 - i. In HiDS **Motor** tab, select the **Manual** feedback method in the **Feedback** drop-down list.



3. Enable the Controller Torque mode (by disabling velocity mode):
 - i. In HiDS **Loop Gains** tab, select **Torque** method in the **Control Mode** drop-down list.
4. Initialize the current-loop gain to a small value, to avoid possible oscillations from the start:
 - i. In HiDS **Loop Gains** tab, set the **Current Loop Kp** = 0.1.

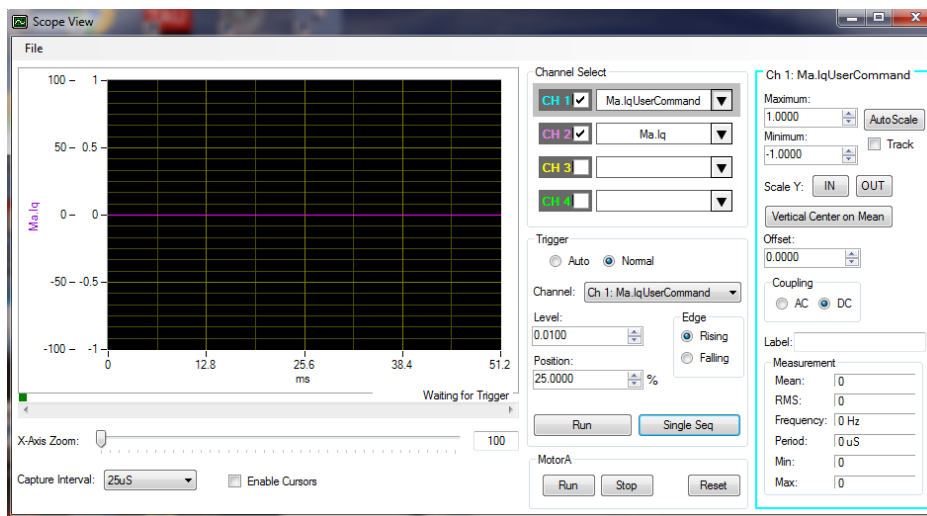


5. Open the HiDS Run Panel, set variables **IqUserCommand** = 0 and **RPMUserCommand** to 0.

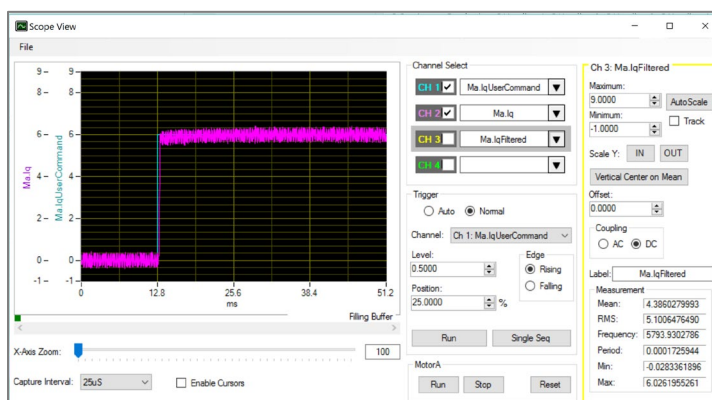
4.3.2 Step Response Procedure

1. Open the HiDS Scope. Set channel 1 to **IqUserCommand** (for trigger), and set channel 2 to **Ma.Iq**. Set the trigger channel to channel 1 (command), trigger-level at 0.01 (non-zero), and trigger-position at 25%, Rising-edge,

Normal-trigger, single-sequence (see section **5.8 Scope/Digital Test Points** for examples of using the HiDS Scope).



2. First test the setup: During this tuning procedure the motor should not turn. In the HiDS **Run Panel**, set variable **IqUserCommand** to approximately 10% of the motor's rated constant-current value, and click on the HiDS Run Panel [Run] button. You may hear some motor hum, but the motor should not turn. The HiDS-scope should have also triggered. Click on the HiDS Run Panel [Stop] button, and re-arm the HiDS-scope by clicking [Single Seq].
3. In the HiDS **Run Panel**, set variable **IqUserCommand** to 0, and click on the HiDS Run Panel [Run] button. Then set variable **IqUserCommand** value to approximately 10% of the motor's rated constant-current value and observe the step response on the oscilloscope. For most applications, the goal is to achieve the classic critically damped response.
4. On the HiDS **Compensation** page, adjust **Current Loop Kp** and re-run the step response (while leaving the motor "running", set variable **IqUserCommand** to zero, and then set variable **IqUserCommand** to 10% of the motor's rated constant-current value) until a critically damped response is achieved.
 - i. Note the gain-values can vary widely depending on the Controller PWM frequency (typically 20Khz) and the motor parameters. **In most cases, Kp will be between 0.1 and 30.**
 - ii. A typical current-loop step response is shown below:



4.3.2.1 Troubleshooting

1. Verify the high-voltage supply is properly read by the Controller; the Vbus value shown on the Run Panel displays the high-voltage measured (may have to click [Run] on the Run Panel, if the Controller has a bus-switch).
2. If the motor inductance is relatively large, the motor power-supply voltage may have to be increased to achieve the characteristic step response.
3. Connect an external current-probe on each of the 3 motor phases, and Manual Feedback can be used to run the motor "open-loop" (without feedback). Refer to section 9.5 for the description of Manual Feedback. Verify current is flowing in each of the 3 phases. Note the motor may or may not turn here because this is the trouble-shooting section, so there must be a problem.

4.3.3 Setting the Current-Loop Integral

Establishing precise integral values can be very application specific, but generally the ESI Controller current-loop integral can be left as the default value.

4.4 Feedback Phasing / Calibration

Motor vector control requires precise knowledge of the actual rotor phase, so the alignment (phase-offset) between the rotor and the physical feedback (resolver, encoder, Hall, etc) must be established. Different motor manufacturers establish the resolver/encoder to rotor alignment differently, so typically the phase angle must be determined during the initial motor operation.

1. Set the appropriate physical motor velocity feedback (resolver, encoder, etc.):
 - i. In HiDS **Motor** tab, select the **Resolver (or Encoder, Hall, etc)** feedback method in the **Feedback** drop-down list.
2. Enable the Controller Torque mode (by disabling velocity mode):
 - i. In HiDS **Loop Gains** tab, select **Torque** method in the **Control Mode** drop-down list.
3. Initialize the motor phase to zero:
 - i. In HiDS **MotorA (or B)** tab, set the **Resolver (or Encoder, Hall) Offset** to 0.
4. In the HiDS **Run Panel**, set the **IqUserCommand** value to approximately 5% of the motor's rated constant-current value.
5. While still in the HiDS **Run Panel**, click on the [Run] button. If the motor spins, go to step 6. If the motor does not spin, try:
 - i. In the HiDS **Motor** tab, Change the **Feedback Direction** from Forward to Reverse (or visa-versa), and re-run.
 - ii. Increase the **IqUserCommand** as high as 20% of the motor's rated constant-current value.
 - iii. In HiDS **MotorA (or B)** tab, set the **Resolver (or Encoder) Offset** to 90.
6. The objective is to change the **Resolver (or Encoder, Hall) Offset** until the motor stops spinning, regardless of the amount of IqUserCommand (torque) which is applied (which indicates the vector-control is 90° out of phase). Adjust the **Resolver (or Encoder, Hall) Offset** value and re-run. Note as you approach the zero-spin value, you may have to increase the **IqUserCommand** slightly to allow an accurate phase value to be determined.

7. When the stopping phase is properly estimated, write it down. Next add 90° to the **Resolver (or Encoder, Hall) Offset**, return to the **Run Panel**, and re-run.
 - i. If the current and RPM sign are the same, then this phase (stop-value plus 90°) is the correct phase to save.
 - ii. If the current and RPM sign are not the same, then subtract 90° from the stopping phase value found in step 6. Verify in the Run Panel that the current and RPM sign are now the same. This phase (stop minus 90°) is the correct phase to save.

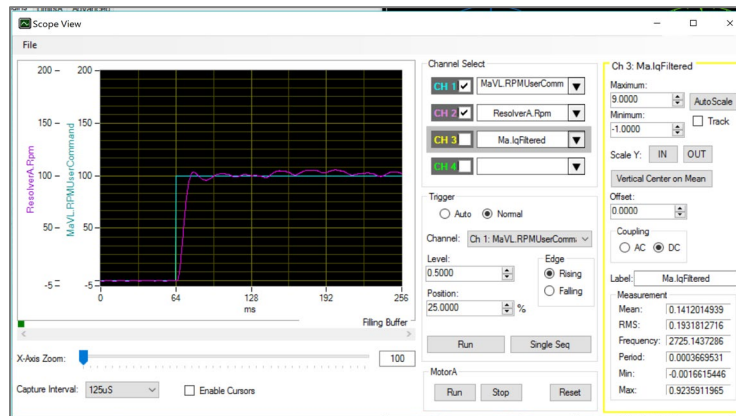
4.5 Velocity-Loop Tuning

This section applies to motors with physical feedbacks (resolver, encoder, etc). For sensorless velocity-loop tuning, see section 4.7 below. There are 2 methods here – the first is for an unconstrained / free-spinning motor (no physical hardstops). The second method is for constrained motors.

4.5.1 Unconstrained motors

1. Set the appropriate physical motor velocity feedback (resolver, encoder, etc):
 - i. In HiDS **Motor** tab, select the **Resolver, Encoder, or Hall** feedback method in the **Feedback** drop-down list.
2. Enable the Controller Velocity mode:
 - i. In HiDS **Loop Gains** tab, select **Velocity** method in the **Control Mode** drop-down list.
4. Set the user-current value to zero, which may be non-zero from the current-loop tuning.
 - i. In HiDS **Run Panel**, set the **IqUserCommand** = 0.
5. Disable velocity ramping (set to an overly large value), which would affect the gain tuning:
 - i. In HiDS **VelocityLoop** page, set variable **MaVL.AccelRPMPerSec** = 100000 (or the Ma.AccelRPM input block on the Run Panel).
6. In the HiDS **Run Panel**, set the **RPMUserCommand** value to approximately 10% of the motor's rated speed value. Note this value must be large enough to allow the selected feedback mechanism to read above its noise floor.
7. Attach the RPM command and the unfiltered velocity value (measured by the selected feedback) to a HiDS scope channel (digital test point).
 - i. In the HiDS **Summary** page, right click on the **MaVL.RPMUserCommand** variable and select Send to Test Point and select DTP1.
 - ii. In the HiDS **Resolver (or Hall or Encoder for encoder)** page, right click on the **RadiansPerSecond** variable and select Send to Test Point and select DTP2.
 - iii. Open the HiDS oscilloscope and configure the oscilloscope to trigger on the rising edge of the DTP1 signal.
8. In the HiDS **Run Panel**, set the **RPMUserCommand** value to 0, and click on the [Run] button; then set **RPMUserCommand** value to approximately 10% of the motor's rated speed value, and observe the step response on the oscilloscope. For most applications, the goal is to achieve a classic critically damped response.
9. On the HiDS **Loop Gains** tab, adjust the **Velocity Loop Kp** value and re-run the step response (while the motor is "running", set the **RPMUserCommand** value to 0, and then set the **RPMUserCommand** value to approximately 10% of the motor's rated constant-current value) until a critically damped response is achieved.
10. Once you are satisfied with the step response, set the **MaVL.AccelRPMPerSec** back to a value reasonable for your motor inertia; a low-inertia motor may be able to accelerate at 10000 RPM-per-second (some even at 100000 RPM-per-second), but a typical value for this is 500 or 1000 RPM-per-second.

A typical velocity-loop step response is shown below



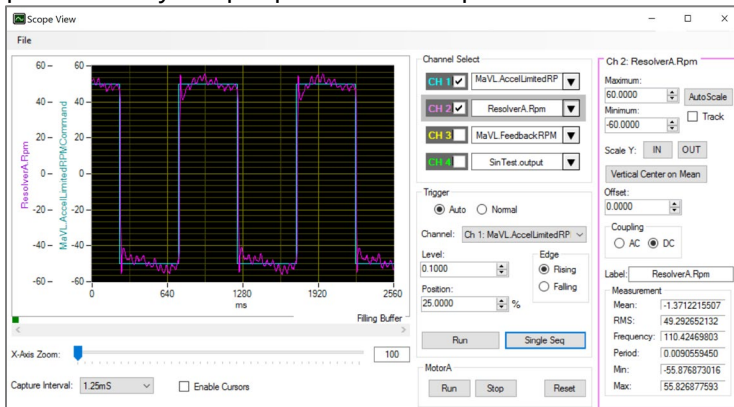
4.5.2 Constrained motors

1. Set the appropriate physical motor velocity feedback (resolver, encoder, etc):
 - a. In HiDS **Motor** tab, select the **Resolver, Encoder, or Hall** feedback method in the **Feedback** drop-down list.
2. Enable the Controller Velocity mode:
 - a. In HiDS **Loop Gains** tab, select **Velocity** method in the **Control Mode** drop-down list.
3. Set the user-current value to zero, which may be non-zero from the current-loop tuning.
 - a. In HiDS **Run Panel**, set the **IqUserCommand** = 0.
4. Disable velocity ramping (set to an overly large value), which would affect the gain tuning.
 - a. In HiDS **VelocityLoop** page, set variable **MaVL.AccelRPMPerSec** = 100000 (or the **Ma.AccelRPM** input block on the Run Panel).
5. Open the Utility page, and set the following variables:
 - a. **SinTest.frequency** = 1 (Hertz).
 - b. **SinTest.OutputIsSquarewave** = 1.
 - c. **SinTest.VelocityCommandAmp** value to approximately 10% of the motor's rated speed value.
Note this value must be large enough to allow the selected feedback mechanism to read above its noise floor.

These settings are applying an oscillating square-wave input to the velocity-loop, which should cause the motor to oscillate back and forth (constrained).
6. Attach the RPM command and the unfiltered velocity value (measured by the selected feedback) to a HiDS scope channel (digital test point).
 - a. In the HiDS **Summary** page, right click on the **MaVL.RPMCommand** variable and select Send to Test Point and select DTP1.
 - b. In the HiDS **Resolver** (or **Hall** or **Encoder** for encoder) page, right click on the **RadiansPerSecond** variable and select Send to Test Point and select DTP2.

- c. Open the HiDS oscilloscope and configure the oscilloscope to trigger on the rising edge of the DTP1 signal.
7. On the HiDS **Loop Gains** tab, adjust the **Velocity Loop Kp** value until a critically damped response is achieved.
8. Once you are satisfied with the step response, set the **MaVL.AccelRPMPerSec** back to a value reasonable for your motor inertia; a low-inertia motor may be able to accelerate at 10000 RPM-per-second (some even at 100000 RPM-per-second), but a typical value for this is 500 or 1000 RPM-per-second.

A typical velocity-loop square-wave response is shown below



4.5.3 Setting the Velocity-Loop Integral

Establishing precise integral values can be very application specific, and often the default value of 0.01 (or 0.005) can be used for most applications. Generally, the ESI Controller velocity-loop integral can be set as follows:

While still observing the velocity step response on the oscilloscope, adjust the **Velocity Loop Ki** until the velocity stabilizes in a time appropriate for the application.

Note there are 2 copies of the Velocity Loop Kp and Ki values on the Loop Gains tab – one pair is for sensorless-feedback, and the other pair is for all other feedback methods. For applications that use both a physical and sensorless feedback on the same motor, this allows the maintenance of 2 velocity tuning settings, which are often different.

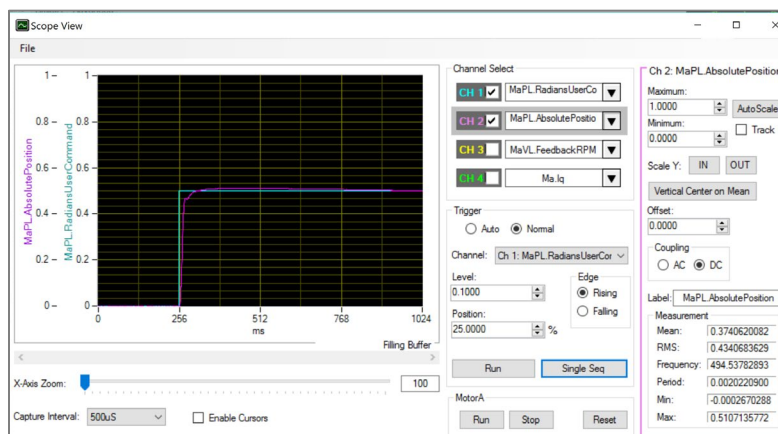
4.6 Position-Loop Tuning

Analogous to the velocity-loop tuning, the objective is to approach a critically-damped step response of the measured position given a command step.

1. Enable the Controller Position mode:
 - a. In HiDS **Loop Gains** tab, select **Position** method in the **Control Mode** drop-down list.
2. Set the user-current value to zero, which may be non-zero from the current-loop tuning.
 - a. In HiDS **Run Panel**, set the **IqUserCommand** = 0 and the **RpmUserCommand** = 0.
3. Disable position-command ramping (set to an overly large value), which would affect the gain tuning:
 - a. In HiDS **PositionLoop** page, set variable **MaPL.AccelRadiansPerSec** = 100000.

4. Open the Position Run Panel (in HiDS, select HiDS->Change Run Panel from... and select the RunPanelSettingsPosition.xml file). In the HiDS **Run Panel**, set the **RPMRadiansCommand** value to approximately 10% of the motor's rated speed value.
 - a. Note if the positional-value is non-zero, the position can be reset by setting the Feedback.RadiansSummed = 0. For example, if resolver is the feedback, open the Resolver page and set ResolverA.RadiansSummed = 0.
5. Attach the Radians command and the measured position to a HiDS scope channel (digital test point).
 - a. In the HiDS **Position** page, right click on the **MaPL.RadiansUserCommand** variable and select Send to Test Point and select DTP1.
 - b. Also in the HiDS **Position** page, right click on the **MaPL.AbsolutePosition** variable and select Send to Test Point and select DTP2.
 - c. In the HiDS **Summary** page, right click on the **MaVL.FeedbackRPM** variable and select Send to Test Point and select DTP3.
 - d. In the HiDS **MotorAHSL** page, right click on the **Ma.Iq** variable and select Send to Test Point and select DTP4.
 - e. Open the HiDS oscilloscope and configure the oscilloscope to trigger on the rising edge of the DTP1 signal.
6. In the HiDS **Run Panel**, set the **RadiansUserCommand** value to 0, and click on the [Run] button; then you may have to experiment with the **RadiansUserCommand** value. With the command-limiter disabled (MaPL.AccelRadiansPerSec = 100000), a large positional step causes a large acceleration, which demands high motor speeds and currents. The **RadiansUserCommand** value should be small enough such that the motor RPM is within the set limits and the motor current (Iq) is also within the set limits.
7. On the HiDS **Loop Gains** tab, adjust the **Velocity Loop Kp** value and re-run the step response (while the motor is "running", set the **RadiansUserCommand** value to 0, and then set the **RadiansUserCommand** value to what was determined in step 6.
8. Once you are satisfied with the step response, set the **MaPL.AccelRadiansPerSec** back to a value reasonable for your motor inertia; a low-inertia motor may be able to accelerate at 100 Radians-per-second, but a typical value for this is 10 Radians-per-second.

A typical position-loop step response is shown below



4.7 Sensorless Configuration

Proper sensorless operation requires accurate motor parameters (Ke, R, L, and pole-pairs) to properly estimate the rotor angle. Once the motor-parameters are configured, there are 8 sensorless configuration variables that control the majority of the sensorless operation. By default, `AutoSetSensorlessXParameters = 1` and these variables are all automatically configured based on the `MaxCurrentCommand` and the `MaOverspeedPos.limit` of the motor. Start with trying to run the motor with these auto-set parameters.

If the motor operation/performance needs further tuning, set `AutoSetSensorlessAParameters = 0`, and these variables can be adjusted:

- `MxSmo.StartupCurrent` -- This is the current required to reliably spin the motor in manual-feedback.
- `MxSmo.MinOpenLoopIq` -- this is the current actually required to run at closed-loop. Must be derived empirically.
- `MxSmo.ClosedLoopRPMThreshold` -- this should be as small as possible to insure reliable startup, but large enough to insure sufficient back-EMF for angle estimation.
- `MxSmo.OpenLoopRPMThreshold` -- this is often 75% of `ClosedLoopRPMThreshold`.
- `MxSmo.OpenLoopRPMPerSec` -- this is the `accelRPMPerSec` used from manual feedback running.
- `MxSmo.RPMToSwitchToMaxAccel` -- usually the same as `ClosedLoopRPMThreshold`.
- `MxSmo.AccelRPMPerSec` -- usually the same as `OpenLoopRPMPerSec`.
- `MxSmo.StartIqRampDelay_ms` – for motors with large inertia, this delay allows the motor rotor to stabilize from the DC-step current input at startup.

4.7.1 Sensorless Velocity-Loop Tuning

While the above procedure fundamentally applies to sensorless, there are some key differences. When in velocity-mode, there are 2 independent regions of sensorless control – open-loop and closed-loop. Sensorless open-loop (below **`MxSmo.ClosedLoopRPMThreshold`**) is essentially the same as manual feedback, so there is no closed-loop velocity-loop control. Hence the **Velocity Loop Kp** and **Ki** values are irrelevant until the closed-loop region is entered.

There are then 2 ways of tuning a sensorless velocity loop:

1. Once the motor has exceeded the **`MxSmo.ClosedLoopRPMThreshold`** RPM, then a velocity step-response can be entered (for example if the closed-loop RPM threshold was 1000RPM, a step from 1200 to 1300 RPM could be used), and the physical-feedback method above can be used as a guide to finish the tuning. This method is problematic though, because the velocity-loop must already be tuned sufficiently in order to exceed the closed-loop RPM threshold.
2. Alternatively, set the **Velocity Loop Ki** = 0, and set the **Velocity Loop Kp** = 0.0001, and try to run the motor into the closed-loop region. With such a low Kp, you may see the motor spin up to the desired RPM and then slowly spin down (because there is insufficient gain to compensate for the RPM error). Start doubling the Velocity Kp until basic control is achieved. You can then optimize the Velocity Kp by attempting to minimize the noise on the measured Iq.

5 The HiDS Application (Details)

First begin by running the application from the Windows Start menu. You will find the application icon named “HiDS” under the “ESI Motion” folder. Once you launch the application, the first screen you will see will be the connection dialog.

5.1 Connection to a Controller

HiDS supports two different methods of connecting to target devices. The type of connection you will choose depends on your particular environment and usage.

The two methods to connect to your controller are (in order of preference):

- CAN
- USB

ESI recommends running over CAN any time power is applied to the motor since USB is less noise tolerant than CAN. USB is convenient for things like firmware updates and parameter loading when motor power is not present.

After you have selected your device and configuration, press **Connect** to begin communicating.

5.1.1 Connecting with CAN

The CAN tab allows you to connect to a device via CAN and requires the use of an **USB-to-CAN compact adapter**. Two USB-to-CAN adapters are supported: the IXXAT part number 1.01.0281.12001 (or equivalent), or the Peak Systems part-number IPEH-002021 (note the Peak Systems device may also be sold rebranded; try Grid Connect). This adapter allows one to communicate with the device using CAN via a USB port on your computer. CAN connections require that you select the CAN channel of your device and the appropriate baud rate. The default CAN channel is 0 with a baud rate of 1M. Note that some customer applications may have unique default CAN settings.

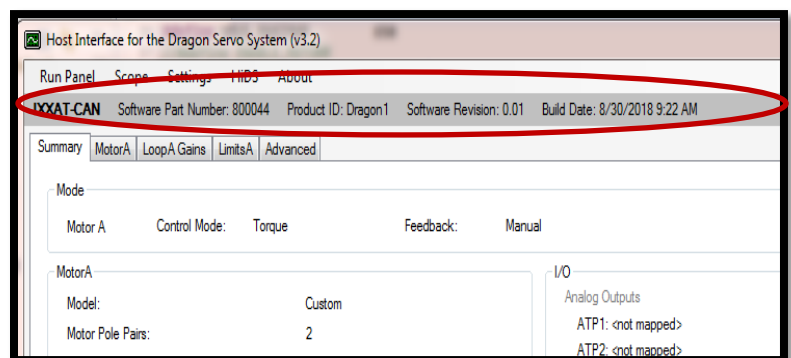
5.1.2 Connecting with USB

To connect via USB, select a target device from the list provided. If you do not see your device in the list or if there are no devices listed, check your connection and ensure that your device is powered on. When viewing the connection dialog, the application will auto-discover your device and refresh the list when you plug in your device. You can also press the **Refresh** button to force a manual refresh of available devices.

Note not all ESI Motion controllers support the USB interface.

5.2 The HiDS User Interface

The main HiDS user interface will be displayed after connection to the device. At the top of the screen you will see a menu bar where you can access various features of the application discussed below. Directly below the menu bar, you will see the device connection display. This display shows the type of connection to the current device, the device software part number, the product Id, the software revision, and the build date. This information can be helpful in identifying the version of firmware currently running on the device.



Example above shows the Dragon:

Below the connection display there is a tabbed information display, defaulting to the Summary tab. This tab contains a summary of critical information about your device. The other tabs provide access to key configuration parameters of the device that you can modify according to your specific requirements. The sections below will discuss key areas of each tab in more detail.

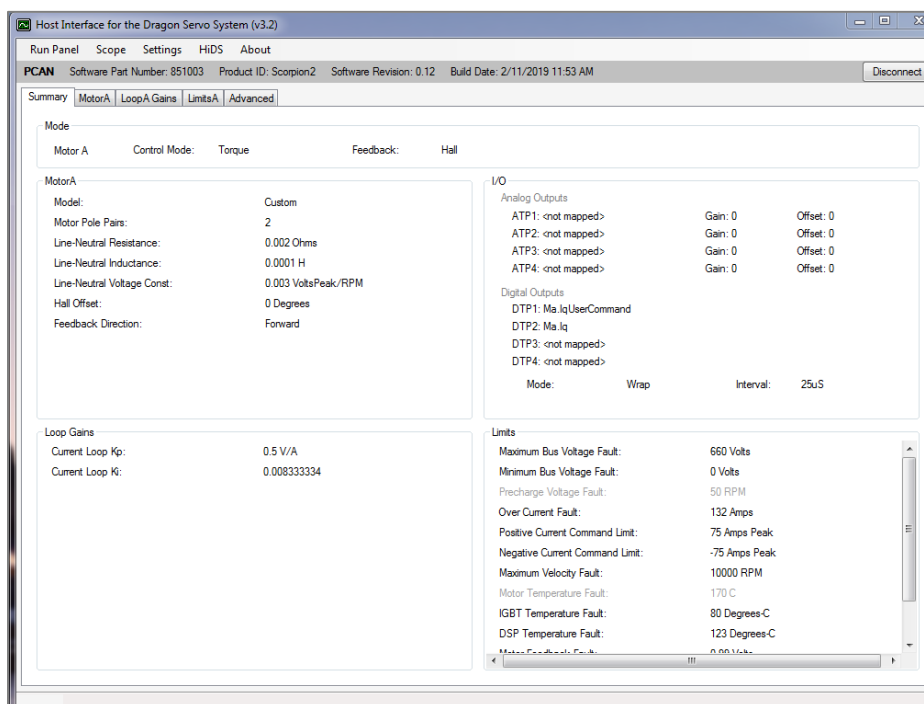
5.2.1 Editing Parameter Values

On the Motor, Loop Gains, and Limits tabs, you will see several text boxes that display the current values of key parameters of your device. If you click in one of the available text boxes and change the value, the value will be written to the device and you will briefly see a confirmation that the value has been changed in the status bar at the bottom of the screen.

If you are entering a value in a field and decide you don't want to send it to the drive, you can press the Esc key on the keyboard before clicking on a different field and the parameter will revert to its previous value. You can also force the value to be updated on the device immediately by pressing the Enter key while the cursor is active in a text box.

5.2.2 Summary Tab

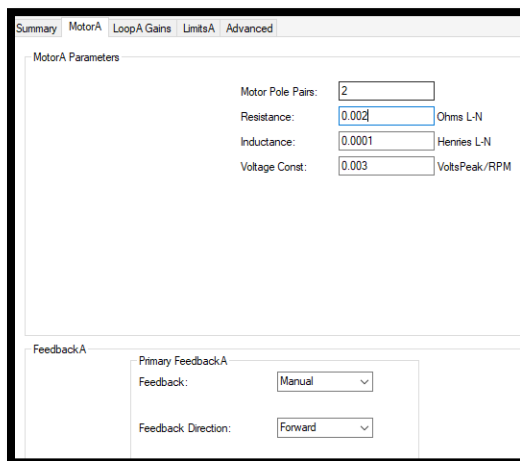
The summary tab displays a read-only view of key device configuration parameters.



5.2.3 Motor Tabs

The top section of the motor page is where you can configure key motor-related parameters for your device. The motor Resistance, Inductance, and Voltage Constant (Ke) are used inside the ESI Motion Controller as a motor equivalent circuit of Line to Neutral since that is how the vector control mathematic is modeled. Resting the mouse over each of these values will display the conversion from Line to Neutral to Line to Line.

The bottom section of the motor page contains feedback related configuration values.



The screenshot shows the 'Motor Parameters' tab for 'MotorA'. It contains the following fields:

- Motor Pole Pairs: 2
- Resistance: 0.002 Ohms L-N
- Inductance: 0.0001 Henries L-N
- Voltage Const: 0.003 VoltsPeak/RPM

Below these fields is the 'FeedbackA' section with two dropdown menus:

- Primary FeedbackA: Manual
- Feedback Direction: Forward

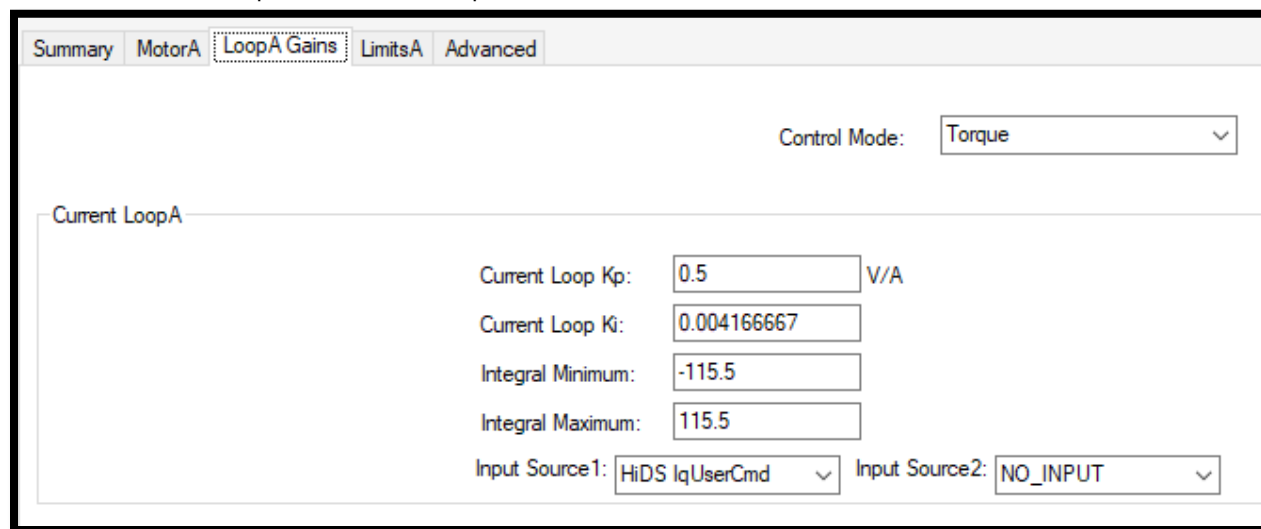
5.2.4 Loop Gains Tab

The loop gains tab is dynamically configured based on the **Control Mode** selection you make at the top of the screen.

In addition, each loop input can be manually configured to be a sum of up to two command sources. The available loop input commands are:

- Sine Test
- HiDS Radian Command
- CAN Command
- RS422 Command
- (Resolver) Cosine Analog Command
- Resolver Sine Analog Command
- Analog Input

Figure below shows the Loop GAIN tab in Torque mode



The screenshot shows the 'LoopA Gains' tab for 'MotorA'. The 'Control Mode' is set to 'Torque'. The 'Current LoopA' section contains the following fields:

- Current Loop Kp: 0.5 V/A
- Current Loop Ki: 0.004166667
- Integral Minimum: -115.5
- Integral Maximum: 115.5
- Input Source1: HiDS IqUserCmd
- Input Source2: NO_INPUT

5.2.5 Limits Tab

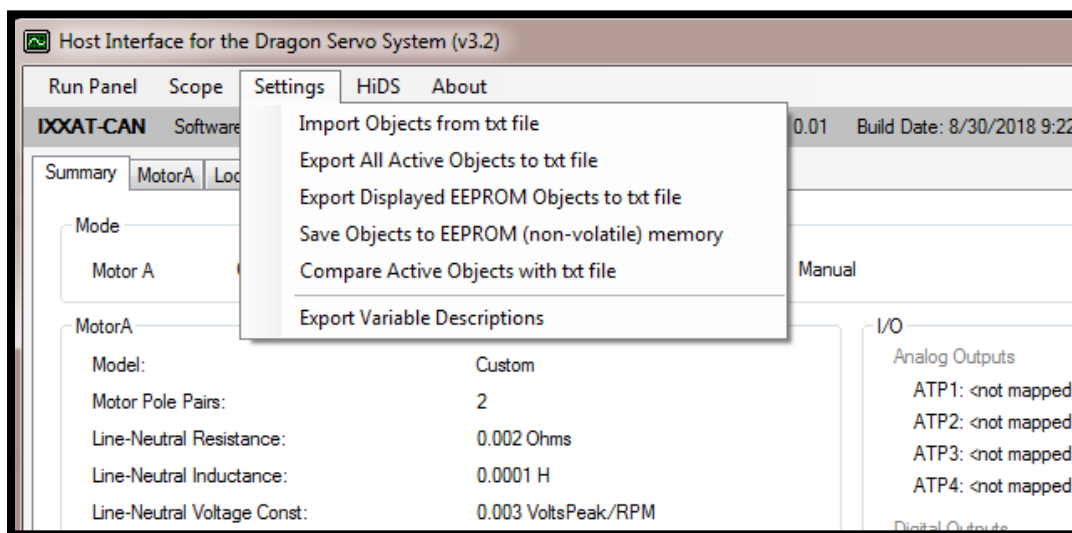
The limits tab provides checkboxes to enable or disable the available fault and warning checks. The fault and warning limits can be entered if it is enabled.

Fault Limits:		Warning Limits:	
Max Bus Voltage:	350 Volts	☒ Enable	262.5 ☐ Enable
Min Bus Voltage:	0 Volts	☒ Enable	0 ☐ Enable
Precharge:	275 Volts	☒ Enable	
Over Current Fault:	72 Amps	☒ Enable	11.25 ☐ Enable
Maximum Velocity:	10000 RPM	☒ Enable	7500 ☒ Enable
Motor Temperature:	170 C	☐ Enable	127.5 ☐ Enable
IGBT Temperature:	80 C	☒ Enable	60 ☒ Enable
DSP Temperature:	95 C	☐ Enable	80 ☒ Enable
Motor Feedback:	1.5 Volts	☒ Enable	0.75 ☐ Enable
I ² T:	126005.4 Amps ² * T	☐ Enable	
Regen Voltage:	1000 Volts	☐ Enable	
Positive Current:	60 Amps Peak		
Negative Current:	-60 Amps Peak		

5.2.6 Importing, Exporting and Saving Settings

In the Settings menu selection, one can export all device settings to a text file. Select **Settings->Export Objects to txt file**, and you will be prompted to select a file name and location where you would like the export file to be stored. This file will contain the current value of all objects in the device. This file could be imported; however, this large file is not typically imported as only a few variables need to be modified for a particular motor or operation.

Alternately **Settings->Export Displayed EEPROM Objects to txt file** allows you to export a text file that only has permanent variables written. The above export writes all values, which may be useful in some cases. However, if it is desired that a text file be exported that can then be subsequently imported, this option should be used. The “Displayed” qualifier limits the export to only those objects that have been selected to be displayed. See section 5.3.5 (Data Objects Filtering) below for more information.



You can use the **Import** function (select **Settings->Import Objects from txt file**) to replace the values of data objects that appear in the file you select from the Import file requestor. An import file must be *.txt and must follow a specific format. A simple two-variable import file is shown as follows:

```
Host Controller Interface System. Version: ESI 1.00,Sep 7 2012,15:45:03,  
0000,Ma.Kp,0.2,           //Motor ID1 gain  
// Motor ID1 maximum command:  
0000,Ma.MaxCurrentCommand,120,
```

The first line is required in the file but is not well parsed by HiDS (so the above first line could be used). All variables to be imported occur on individual lines and follow a common format: The leading 0000 is a variable identification, which is unused during import, so it could be four zeros. After comma separators, the variable name, its value; note the trailing comma is required on each line. The parsing of each line stops after the last comma, so comments can be added at the end of each line for improved readability and maintenance.

To save changes in non-volatile memory, select **Settings->Save Objects to non-volatile memory**, which will save the current data object values to non-volatile memory, so that the current values will remain even after power-cycling the device.

5.2.7 Advanced Tab

The **Advanced tab** provides a more complete, low-level interface to the device including access to all available data objects as well as features such as digital and analog test point configuration, digital test point capture, serial port testing, etc. The **Advanced tab** is discussed in detail in the next chapter.

5.2.8 Firmware Upgrades

The **Connection View**, which is displayed first after starting HiDS, also provides the ability to update the Firmware executing on the ESI Controller. Depending on your Controller, you can update the firmware via USB and/or CAN. Note while all Controllers support firmware upgrade via at least one interface, not all controllers support both USB and Can-based upgrades. To update the Controller firmware:

1. From the Connection View, select the USB or CAN tab.
2. Click on the [Update Firmware] button.
3. Browse to the firmware zip-file provided by ESI Motion. This file will be named ESI_<Customer name>_<ESI software number>_<version>.zip, where:
 - a) <Customer name> is probably your company name for custom controller, or "ESIMotion" for off-the-shelf controllers.
 - b) <ESI software number> is one of the following:
 - i) 800xxx number for Dragon1
 - ii) 810xxx number for Dragon2
 - iii) 830xxx number for the single-core Mite family
 - iv) 841xxx/851xxx for all multi-core controller families (Mite, Scorpion, Atom, Draco, etc)

v) 820xxx number for FPGA firmware.

c) <version> is a 3-digit number representing the version. For example, version 1.23 would be represented as 123.

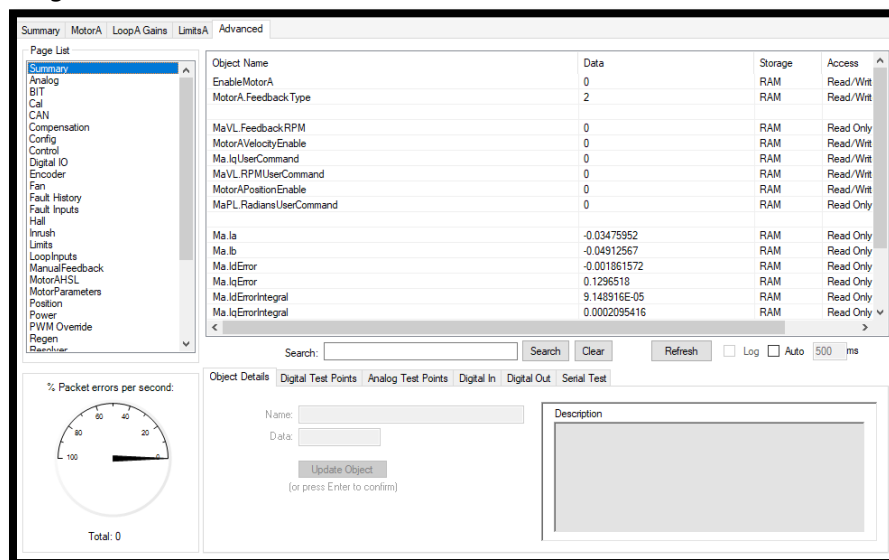
4. Select the firmware zip-file, click [Open], and [Continue]. The firmware update takes 1-3 minutes depending on the PC speed and target processor.
5. Should the upgrade fail, power cycle the controller and try again.

5.3 Advanced Device Management

The **Advanced tab** provides access to a number of tools that can be used for advanced device access and management. This is provided for advanced users in non-standard applications. Note that the advanced tab can bypass some of the self-protection features of the device – care must be taken when using the **Advanced tab**.

5.3.1 Page List

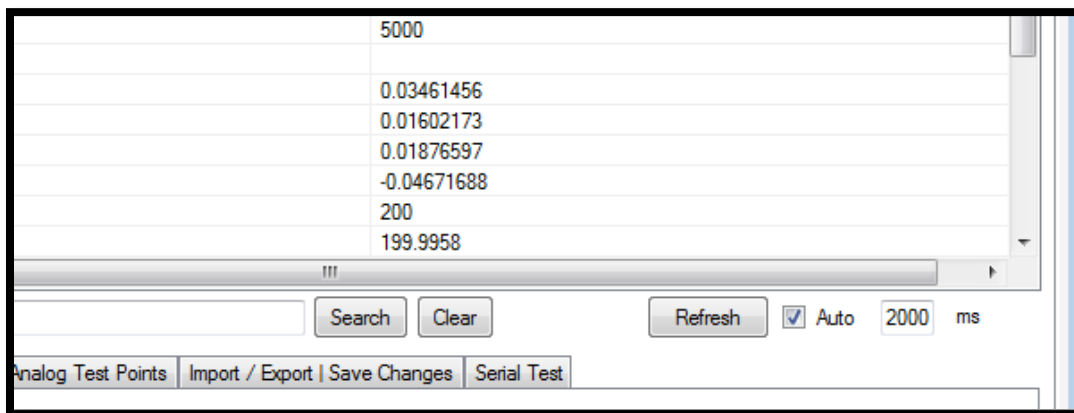
The **Page List** appears at the top, left of the **Advanced tab**. Pages provide a means of organizing data objects into distinct groups. If you click on a page from the page list area, all data objects associated with the selected page will appear in the grid at the right.



5.3.2 Data Objects List

The large grid at the top, right of the advanced tab is used to show data objects that are associated with the currently selected page or the results of a data object search. The first column shows the name of the data object and the second column (titled "data") shows the current value of the data object. Since some data object's values will change over time, you can use the **Refresh** button to reload the current values from the device.

You can also check the **Auto-Refresh** checkbox to force the application to reload the values from the device continually. Auto-Refresh requires substantial computing resources – the user may choose to slow the auto refresh by increasing the Auto Refresh rate.



5.3.3 Data Objects Storage

The third column in the Data Objects view is labeled Storage, and it indicates whether the variable is stored temporarily in RAM or can be permanently stored in non-volatile EEPROM. When **Settings->Export Displayed EEPROM Objects to txt file** is selected, only those variables that have a Storage type of EEPROM are saved.

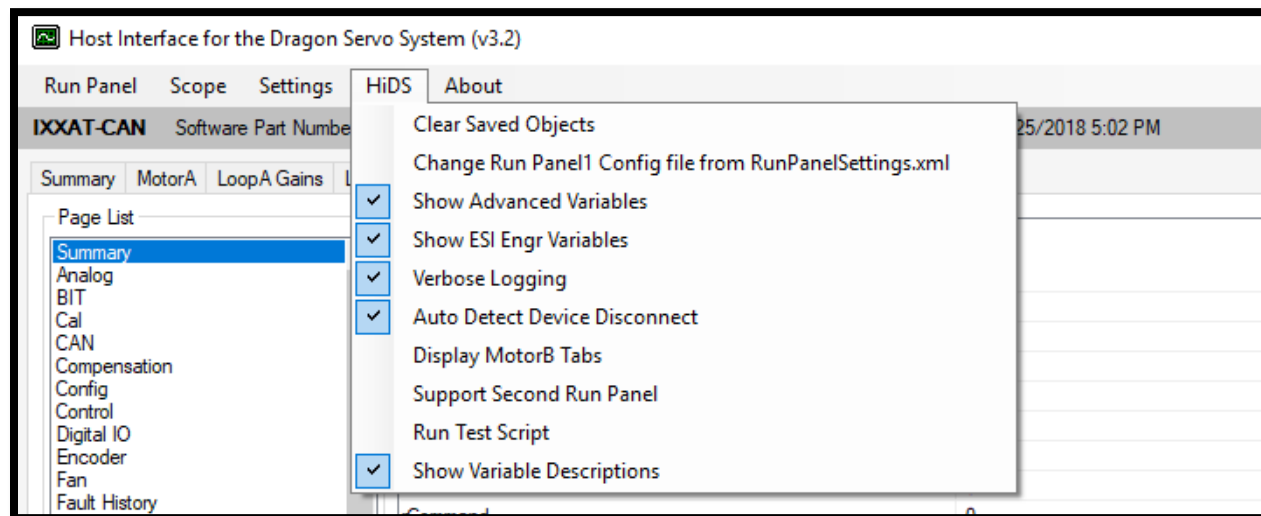
5.3.4 Data Objects Access

The fourth column in the Data Objects view is labeled Access, and it indicates whether the variable is intended to be Read Only from HiDS or is intended to be Read / Write. The HiDS interface provides direct access to all Controller variables, so technically all variables are Read / Write. However, a variable listed as Read Only is a run-time variable that will likely be over-written during normal operation. A variable listed as Read /Write is intended to be a configuration variable that will retain its settings.

5.3.5 Data Objects Filtering

The fifth and last column in the Data Objects view is labeled Type, and there are 3 variable types in the system:

1. ESI Engr (engineering): these variables are considered internal or excessively advanced as to be useful during customer monitoring and debug. They are available, but the HiDS display can be made simpler by hiding these variables – Uncheck **HiDS->Show ESI Engr Variables** to hide these variables.
2. Advanced: these variables are tagged as advanced to again limit the complexity of the system displayed. It is the intent that motor configuration and basic operation can occur with the Advanced variables hidden. Uncheck **HiDS->Show Advanced Variables** to hide these variables.
3. Required variables are those that have no descriptive type in the Type column. These are the variables that are considered fundamental to the Controller operation, and they cannot be hidden.



5.3.6 Searching for Objects

In the event that you already know all or part of the name of the data object you wish to view, you can use the **Search** feature to quickly find data points that match the text you provide. To use the search feature, begin typing the partial name of the data object in the textbox labeled "Search:" found just below the data objects list. Note that the application will continuously update the data objects list with all data objects whose name matches the text you have typed. It is not necessary to type the entire name of the data object and the search is not case sensitive. The **Clear** button can be used to clear the current search results.

When you are searching, the data objects list will show an additional column that will indicate in which page this data object is normally found. If you right-click your mouse on a data object shown in the search results, you can access the **Go to object's page** menu item and the application will navigate to show all objects found on the same object page as the search result item.

5.3.7 User-Defined Object Page

For convenience, you can also assign data objects to a custom user-defined object page. This allows you to view your personal most commonly accessed data objects in a single location. To add a data object to your user-defined object page, you can right-click your mouse on the data object you wish to add and select the **Add to User Defined List** menu item.

To view your user-defined page, you can scroll to the bottom of the Page List and click on the **User Defined** page. You can also remove an item from your user-defined page by right-clicking your mouse and selecting the **Remove From User Defined List** menu item.

5.3.8 Editing Object Values

To edit the value of a data object, click on the data object row from the data objects list grid and then refer to the **Object Details** tab below the data objects list. From here, you can modify the value in the textbox labeled "Data:"

As described in the **Editing Parameter Values** section 5.2.1 above, you can use the **Enter** key to commit the current value to the device or the **Esc** key to revert to the previously stored value. You can also commit the current value to the device by pressing the **Update Object** button.

5.3.9 Digital Test Points

The **Digital Test Points** tab found below the data objects list provides methods for capturing data from assigned digital test points. These features are described in detail in the **Scope / Digital Test Points** section 5.8 below.

5.3.10 Analog Test Points

The **Analog Test Points** tab found below the data objects list provides methods for assigning data object values to one of the four available analog test point outputs. These features are described in detail in the **Analog Test Points** section 5.4 below.

5.3.11 Digital In

The **Digital In** tab found below the data objects list provides methods for assigning Controller digital-inputs to any HiDS object. These features are described in detail in the **Digital In** section 5.5 below.

5.3.12 Digital Out

The **Digital Out** tab found below the data objects list provides methods for assigning Controller digital-outputs to any HiDS object. These features are described in detail in the **Digital Out** section 5.6 below.

5.3.13 Serial Test

This tab provides a system for testing the HiDS communications path (whether it be USB or CAN) and is typically used in cases where a faulty serial port, cable or device interface is suspected. Upon pressing the **Start Serial Test** button, the system will begin to send randomly generated messages to the device and will test to confirm that they are echoed back from the device without any corruption.

You can run this test for any length of time desired and the application will keep track of and display several parameters to indicate the success of the test. **Serial Test Time** will show how long the test has been running. **Successful Tests** will show a count of the number of test iterations that have succeeded without error.

The **Serial Data Errors** value shows the number of test iterations that have resulted in a serial error or data corruption. Finally, the **Serial Timeout Errors** value will indicate how many times the serial port access attempt has timed out before a response was received.

5.4 Analog Test Points

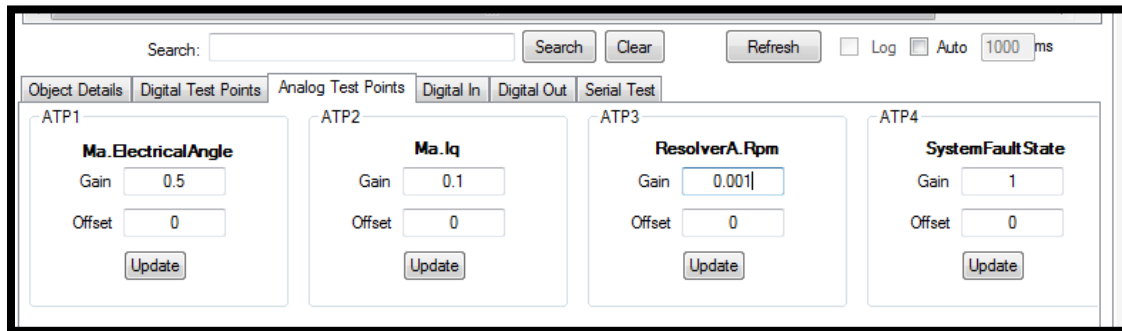
Analog test points allow for the values of data objects in the device to be accessed via the analog test point leads from the device, typically for use with external measurement and instrumentation equipment.

5.4.1 Assigning Data Objects to Analog Test Points

Most ESI Motion Controllers provide an Analog Test Point feature, which allows driving an analog signal that is based on any Controller measurement. For example, a scaled analog voltage based on the motor speed or position can be configured. Analog Test Points are updated at 20Khz or 40Khz, based on the Controller. This feature is especially useful for driving a Digital Signal Analyzer for bode plot analysis.

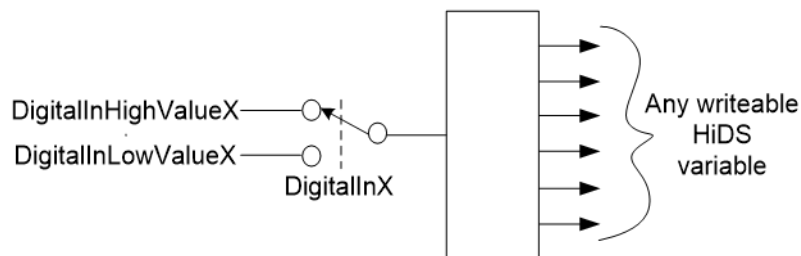
Any data object can be assigned to any one of the available analog test points available. To assign an analog test point, begin by locating the test point in the data objects list. Once displayed, right-click your mouse on the data object you wish to map to a digital test point and from the menu, select **Send To Test Point** and then select one of the analog test points from the following menu (labeled "ATP1"... "ATP4").

Once assigned, the **Analog Test Points** tab at the bottom of the screen will be selected and you should see your selection under the appropriate **ATP** area. From here you can also specify an **Offset** and **Gain** to apply to this test point. The Gain must be non-zero to use the test point. Once you have entered the desired offset and gain, be sure to press **Update** to apply these settings to the device.



5.5 Digital In

Most controllers contain some number of user-assignable digital inputs. Any HiDS object can be assigned to be updated via a digital input, which is useful to control Controller operation from an external device capable of driving digital outputs. The Digital In logic can be described by the following diagram:



Using DigitalIn1 as an example, any HiDS variable can be assigned to be updated via DigitalIn1 as described below. At a 1Khz rate, the Controller reads the DigitalIn1, if the DigitalIn1 is read as a 0, the assigned variable is set to DigitalInLowValue1; if the digital input is read as a 1, the assigned variable is set to DigitalInHighValue1. To detach (un-assign) a digital input, assign the input to variable "NullVariable" shown on the Utility page.

5.5.1 Assigning Data Objects to Digital In

Before control can be performed, you must associate one object with an available digital input. Refer to the Controller's Installation Manual for the specific Digital In pin assignments.

To assign a data object to a Digital In, you need to first make sure the data object is currently displayed in the data objects list by either clicking on the data objects' page or by searching for the data object name.

Once displayed, right-click your mouse on the data object you wish to map to a digital input and from the menu, select **Send To DigitalIO** and then select one of the digital inputs shown.

Once you have assigned a data object to a digital input, you will see it within the **Digital In** tab.

For example, to use Digital In1 to enable/disable the motor, the variable of interest is **EnableMotorA**.

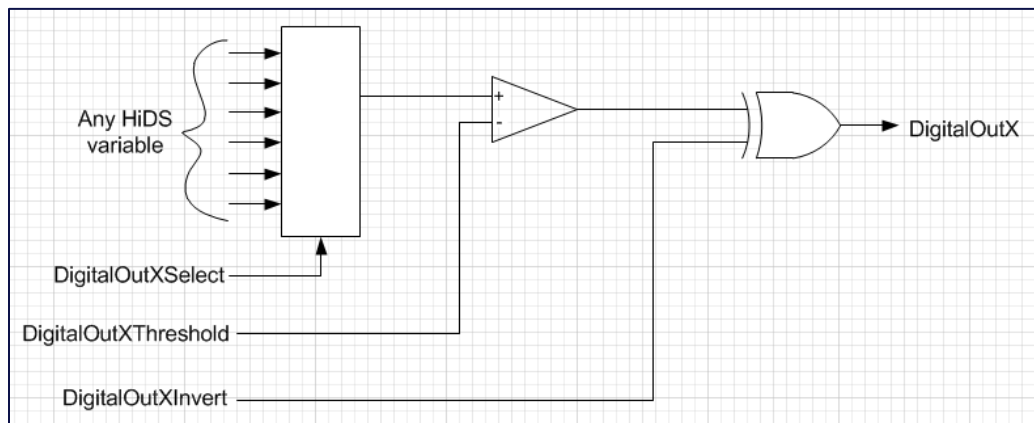
1. Right click on EnableMotorA, and select **Send to DigitalIO** and then select **Digital In1**.
2. Open the Digital In dialog by clicking the Digital In tab near the bottom of HiDS.

3. If it is desired that a DigitalIn1 at a logic high will enable the motor, and DigitalIn1 at a logic low will disable the motor, then set DigitalInLowValue1 to 0 and DigitalInHighValue1 to 1. If it is desired that a DigitalIn1 at a logic low will enable the motor, and DigitalIn1 at a logic high will disable the motor, then set DigitalInLowValue1 to 1 and DigitalInHighValue1 to 0.

5.6 Digital Out

Most controllers contain some number of user-assignable digital outputs. Refer to the Controller's Installation Manual for the specific Digital Out pin assignments.

Any HiDS object can be assigned to a digital output, which is useful to convey Controller status to an external device capable of reading digital inputs. The Digital Out logic can be described by the following diagram:



Using DigitalOut1 as an example, any HiDS variable can be assigned to DigitalOut1 as described below. At a 1Khz rate, the Controller compares the assigned variable to DigitalOut1Threshold (as shown in the comparator block above); note if invert is false, the compare pseudo-code is:

if variable > DigitalOut1Threshold, then DigitalOut1 = true.

Finally, if the application requires an inverted output, set DigitalOut1Invert = 1 (true).

5.6.1 Assigning Data Objects to Digital Out

Before control can be performed, you must associate one object with an available digital output. To assign a data object to a Digital Out, you need to first make sure the data object is currently displayed in the data objects list by either clicking on the data objects' page or by searching for the data object name.

Once displayed, right-click your mouse on the data object you wish to map to a digital output and from the menu, select **Send To DigitalIO** and then select one of the digital outputs shown.

Once you have assigned a data object to a digital output, you will see it within the **Digital Out** tab.

For example, to use Digital Out1 to indicate whether a fault condition is present, the variable of interest is **SystemFaultState**.

1. Right click on SystemFaultState, and select **Send to DigitalIO** and then select **Digital Out1**.
2. Open the Digital Out dialog by clicking the Digital Out tab near the bottom of HiDS.

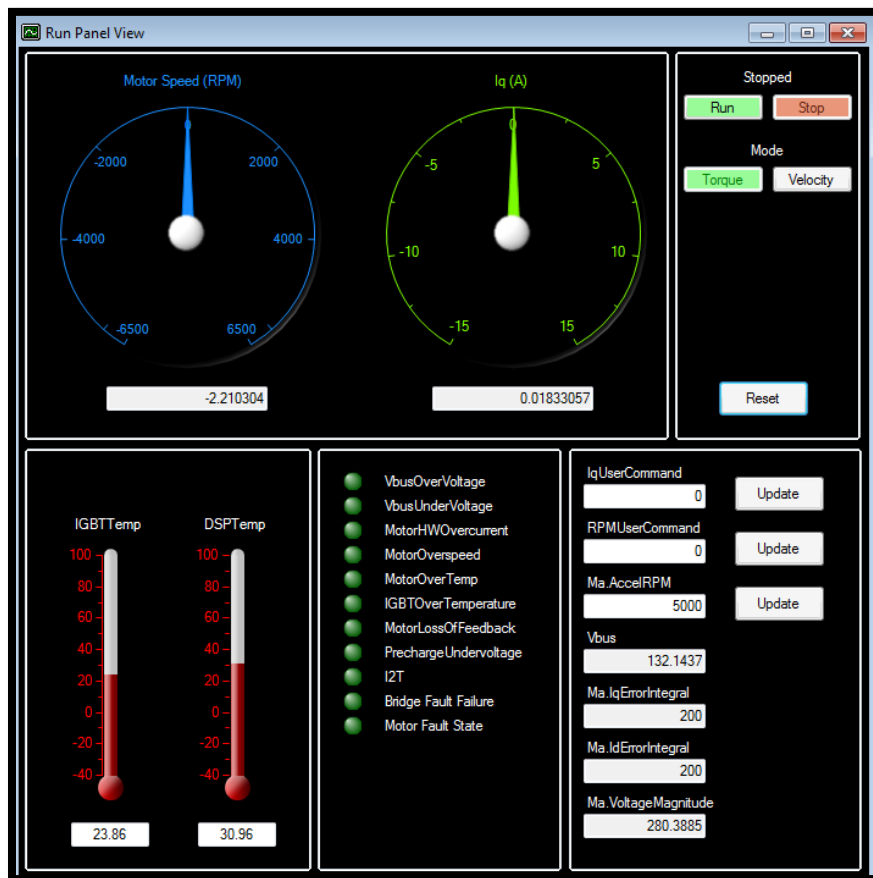
3. If it is desired that a DigitalOut1 at a logic high will indicate a fault-condition, and DigitalOut1 at a logic low will indicate no fault, then the Invert checkbox can be left unchecked. If it is desired that a DigitalOut1 at a logic low will indicate a fault condition, and DigitalOut1 at a logic high will indicate no fault, then check the Invert checkbox.

Because the variable at a value of one will produce a digital out of 1, then the **Gain** variable can be left at 1.

5.7 Run Panel

The **Run Panel** can be used to view key data fields from your device using a graphical interface. You can also start and stop the motor from this interface.

The **Run Panel** is intended to show all relevant status information for the motor. The Motor, Loop Gains, and Limits tabs described above provide the configuration screens, and the Run Panel shows the operational status.



Much of the Run Panel can be customized for specific applications. The default Run Panel configuration file is "RunPanelSettings.xml", which is located in the Windows ProgramData directory, which is Windows-OS-version specific. In Windows 7, this directory is in C:\ProgramData. In this directory, there is a \ESI Motion\HiDS\{version} directory (where version is the HiDS version). This default file should not be edited.

5.7.1 Run Panel Customization

To modify a new Run Panel control:

1. Copy the default RunPanelSettings.xml to another file, for example to RunPanelSettingsMyProduct.xml.

2. For each control in the Run Panel, there is an xml data structure that allows naming the control and maps the control to a product variable name as follows:
 - a. Control description: <ControlDesc>Ma.AccelRPM</ControlDesc> (name shown on Run Panel)
 - b. Variable name: <ObjectName>MaVL.AccelRPMPerSec</ObjectName> (variable that is read)
3. To bind this field to a different variable, for example the Phase-A current, Ma.Ia:
 - a. Change the description name from Ma.AccelRPM to A Current
 - b. Change the variable name from MaVL.AccelRPMPerSec to Ma.Ia
4. Save the xml file, and select it via the HiDS Settings menu: Change Run Panel Config file from...

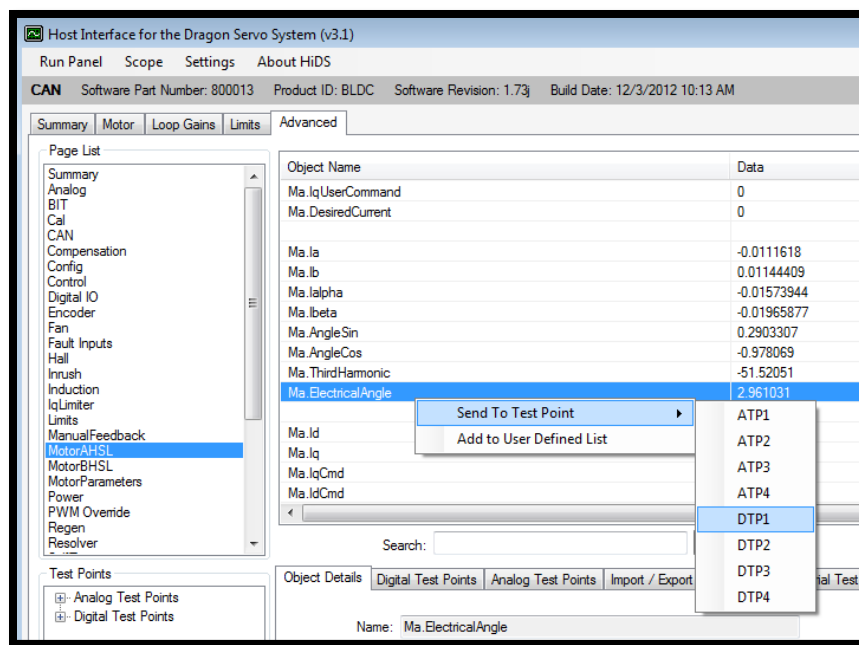
5.8 Scope / Digital Test Points

All devices are capable of capturing a buffer of up to 2048 samples of any data object in the device. Normally this data is displayed with the **Scope** feature. The comma delimited data can then be downloaded and viewed off-line with spreadsheet software to review or graph.

5.8.1 Assigning Data Objects to Digital Test Points

Before capturing can be performed, you must associate from one to four data objects with an available test point. To assign a data object to a test point, you need to first make sure the data object is currently displayed in the data objects list by either clicking on the data objects' page or by searching for the data object name.

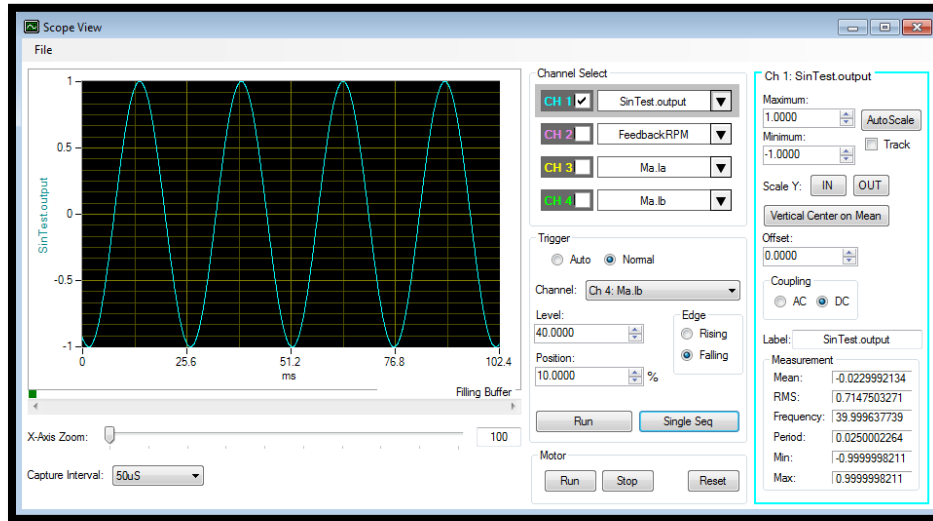
Once displayed, right-click your mouse on the data object you wish to map to a digital test point and from the menu, select **Send To Test Point** and then select one of the digital test points from the following menu (labeled "DTP1"... "DTP4").



Note assigning an object to a Digital Test Point here is the same as selecting that object in the Scope View (the Scope uses the same Digital Test Point capture mechanism).

The scope window can be opened by clicking on the **Scope** menu item at the top of the screen. The scope provides a familiar interface for accessing and viewing data objects from the device. The scope interface is divided into three vertical sections. The first section on the left contains the scope waveform display at the top and the controls for X-Axis scrolling, **X-Axis Zoom** and **X Time Base** just below.

The center section contains the **Channel Select** area at the top and the **Trigger** parameters at the bottom. The final section on the right contains detailed channel-based parameters for the currently selected channel from the **Channel Select** area.



5.8.2 Waveform Display

The waveform display area on the left will display the output from the currently active channels. The waveform will be color-coded to match the color shown in the **Channel Select** area. Depending on how many channels are selected, from one to four vertical (y-axis) scales will be presented along with a color-coded label containing the label for that channel (see **Channel Parameters** below for more details).

Below the waveform area, there is a slider labeled **X-Axis Zoom**. This can be used to zoom in on the waveform in the x-axis direction and the valid range of values is from 100% to 1000% (10x zoom). When zoomed above 100%, there will be a horizontal scroll bar directly below the waveform display that can be used to scroll horizontally to view the waveform.

The **X Time Base** drop down list can be used to modify the capture interval (from **Digital Test Points** above).

5.8.3 Channel Selection

The **Channel Select** area can be used to select digital test points and to activate or deactivate their display on the waveform view. To select a digital test point for one of the four available channels, click the down arrow icon to the right side of the channel display. This will present a dialog where you can choose a data object. The list box on the left of the dialog shows a list of object pages. After selecting an object page, the list box on the right will display data objects contained in the selected object page.

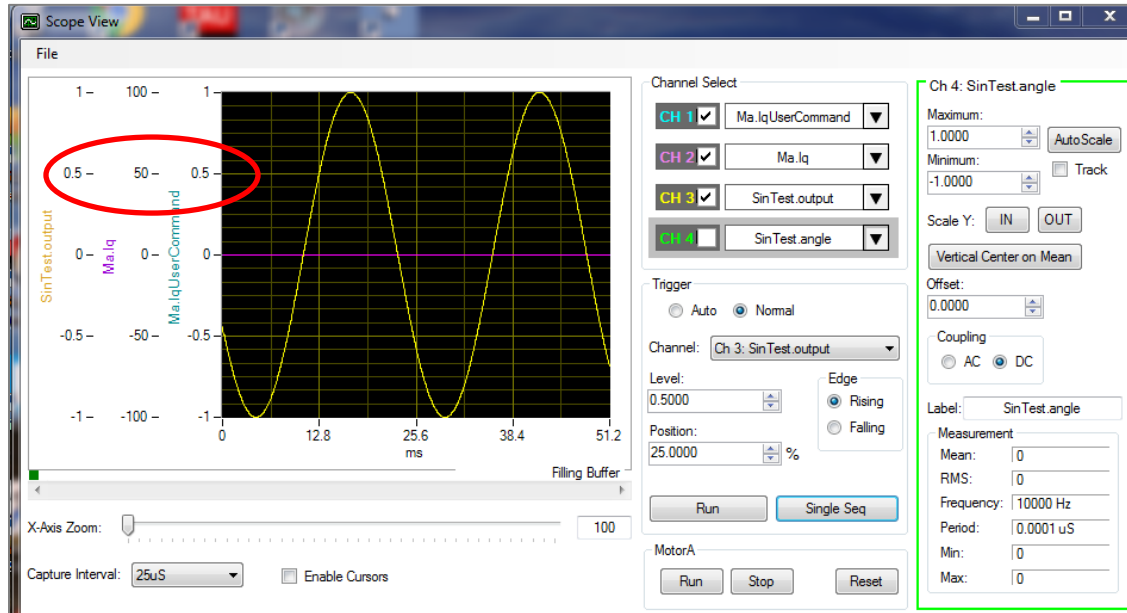
Note the user may find it easier to navigate to instead select the Scope channel via the HiDS user interface.

The check box on the left of each channel selector will toggle the visibility of that channel on the waveform display.

5.8.4 Trigger Settings

At the top of the **Trigger** section, you can select between the two trigger modes (**Auto** and **Normal**). The default mode is **Normal** and, in this mode, triggering is disabled. The **Auto** mode will use the parameters below to automatically orient the waveform based on the trigger.

The **Level**, **Position**, and **Edge** parameters of the trigger function are intended to emulate a physical oscilloscope. In the scope-example below, the scope is set to trigger on channel 3 (the internally generated sine wave, SinTest.output) at Level = 0.5, Position = 25%, and Edge = Rising. The trigger point-in-time is circled below, which occurs when the SinTest.out = 0.5 AND is rising, and the trigger location is at 12.8ms, which is 25% of 51.2ms.



The **Run** button can be used to continuously capture and display and the **Single Seq** button can be used to capture a single data buffer only.

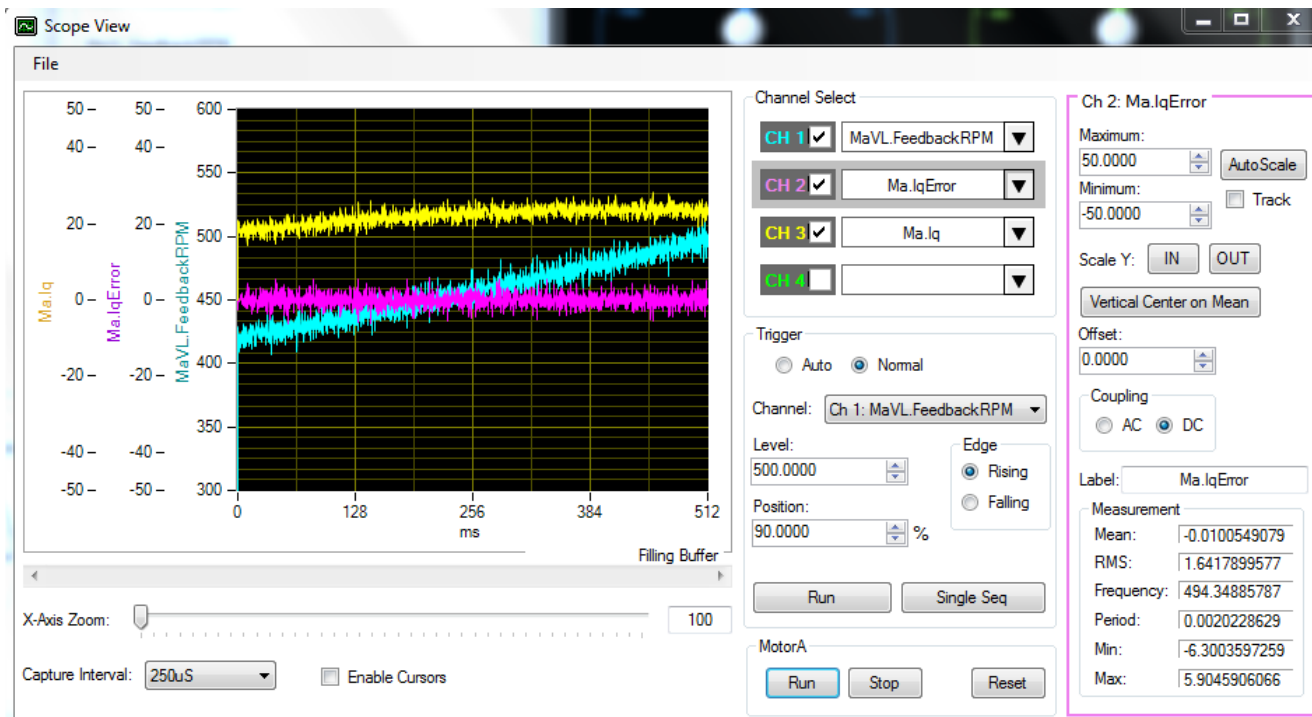
5.8.4.1 Trigger Examples

The **Auto** and **Normal** radio buttons and the **[Run]** and **[Single Seq]** buttons provide 4 triggering use-scenarios:

- Normal-Triggering, Single-Seq: Provides a single capture when the trigger-criteria is met: when the selected trigger-channel crosses the trigger-level while progressing in the direction of the trigger-edge. For example, to capture the measured IQ and IQError at the moment the motor accelerates past 500RPM, the following scope settings would be required:
 - Set DTP1 = MaVL.FeedbackRPM
 - Set DTP2 = Ma.IqError
 - Set DTP3 = Ma.Iq
 - Trigger Mode = Normal
 - Trigger Channel = Ch. 1 MaVL.FeedbackRPM
 - Trigger Edge = Rising (RPM is increasing)

- Position = any value from 0 to 100. To capture data preceding the trigger-moment, set the position to a larger number (like 90%). To capture data following the trigger-moment, set the position to a smaller number (like 10%).
- Click [Single Seq] – the scope display should read “Waiting for Trigger”

This example is shown in the following scope screen-shot:



- Normal-Triggering, Run: Will re-capture data every time the trigger-criteria is met.
- Auto-Triggering, Single-Seq: Will immediately capture the channel data – regardless of trigger settings.
- Auto-Triggering, Run: Will repeatedly capture the channel data – regardless of trigger settings.

5.8.5 Channel Parameters

The far-right side of the screen displays parameters specific to the currently selected channel from the **Channel Select** area. The currently selected channel is indicated by a medium gray bar around the channel display. The channel parameters area displays the currently selected channel number and data object name at the top of the display and the rectangular outline of the channel parameters area is also color-coded to match the channel's color.

As channels are selected for display, the **Y-Axis** scale will be automatically adjusted to contain the minimum and maximum values found in the capture buffer for that channel. The minimum and maximum Y-Axis values can be modified using the two text boxes labeled **Minimum** and **Maximum** and the values can be automatically determined at any time by pressing the **AutoScale** button. The **Track** checkbox will force the auto scale to occur on each capture of data buffer. This will result in a y-axis that is constantly changing but can be useful to keep the waveform visible in the display.

The **Scale Y In** and **Out** buttons can be used to adjust the **Minimum** and **Maximum Y-Axis** values in and out by 10% per click. The **Vertical Center on Mean** button will compute the mean of the waveform and then position the mean in the center of the display by adjusting the **Minimum** and **Maximum** values to be equidistant from the mean while keeping the overall vertical range the same.

The **Offset** text box can be used to vertically offset the waveform. This offset is added to the waveform values as the last step so the **Vertical Center on Mean** will not center the waveform vertically in the display if the **Offset** is non-zero.

The value entered in the **Label** text box will be displayed to the left of the waveform display to help identify that channel's vertical scale.

The **Measurement** area at the bottom shows various calculations related to the current channel when it is running including **Mean**, **RMS**, **Frequency** and **Period**.

5.8.6 Cursors and Capture Interval

Below the plot zoom bar the user has the option of modifying both the capture time scale and enable cursors to facilitate data analysis. The capture interval can set from 25 us to 12.5 ms in a series of pre-determined steps. The scope is limited to 2048 samples, so the maximum time duration of a scope capture is limited to 25.6 seconds. There will be independent cursors for each channel enabled and can be moved by left-clicking and dragging with the mouse.

5.8.7 Saving Captured Data

After capturing data the user can save the data to an excel document by going to File→Save Data. All four channels and the time will be exported to the file with their corresponding channel name as the header; the time column will be in units of microseconds.

5.9 Settings

Under the settings drop down menu at the top of the screen, the **Clear Saved Objects** command can be used to delete the cached data objects list for all previously and currently connected devices. This is typically used only if requested by technical support.

Also selectable from the settings drop down menu, the **Auto Detect Device Disconnect** option can be turned on and off. The default value is on which causes the application to periodically check for the presence of the currently connected device and perform an automatic disconnect if the device becomes unavailable (is unplugged or powered off). In the unlikely event that the application is erroneously detecting a disconnected device, this feature can be turned off.

5.9.1 Savings Values

In the HiDS **Settings** menu, click on the **Save Objects to EEPROM (non-volatile) memory** selection. This will save all of these parameters above into non-volatile memory.

Alternately, the resultant gain values and motor phase offset values can be saved into an importable text file, which can be imported as needed. Refer to the parameter-import section for instructions.

Loading new firmware will change saved-EEPROM variables back to the default settings, so it is recommended that an importable text file be created to ensure changes can be recovered.

5.9.2 Comparing Current Settings with a Text File

It is recommended that all configuration changes be saved to an importable text file. This ensures changes are not lost, because a firmware upgrade will change saved-EEPROM variables back to default.

To compare the current Controller configuration with the settings in a text file, in the HiDS **Settings** menu, click on the **Compare Active Objects with txt file** selection. This will compare all of the parameters in the text file with the current settings on the target. The changes will be shown on the resulting dialog, and if there are too many changes to display, all changes will be saved to the log file in the Windows ProgramData directory, which is Windows-OS-version specific. In

Windows 7, this directory is in C:\ProgramData. In this directory, there is a \ESI Motion\HiDS\{version} directory (where version is the HiDS version).

5.10 About HiDS

The **About** menu item opens a window that shows the current application version and build number. This can be helpful if contacting technical support with HiDS-related software issues. It also displays the path to where HiDS stores user and application related data.

5.11 Simulator

The Simulator allows HiDS usage without the PC physically connected to the Controller via USB or CAN, which may be useful for training or evaluation purposes. The HiDS installation contains a simulated product based on the ESI Motion dual-axis Controller.

5.11.1 Simulator Customization

The HiDS installation contains a simulated product based on the ESI Motion dual-axis Controller. HiDS simulates the Controller objects using the file Emulator.xml, which is located in the Windows ProgramData directory, which is Windows-OS-version specific. In Windows 7, this directory is in C:\ProgramData. In this directory, there is a \ESI Motion\HiDS\{version} directory (where version is the HiDS version).

To create a product-specific version of the simulator, perform these steps:

1. Connect HiDS to an actual ESI Motion Controller which is the Controller to be simulated.
2. In the \ESI Motion\HiDS\{version} directory, rename EmulatorDevice.xml to EmulatorDeviceOriginal.xml (back it up).
3. Find the most recent SavedDevices_x.xml file, where _x is an incremented number based on the number of different Controllers this PC has connected to.
4. Rename the most recent SavedDevices_x.xml file to EmulatorDevice.xml.
5. Select HiDS -> Clear Saved Objects.
6. Connect to HiDS Simulator.

5.12 Run Test Script

The Run Test Script view can be used to perform repetitive testing on your motor application. The Run Test Script view assumes the Controller has been properly configured for its Motor Parameters, Limits, and Loop Gains. Essentially the Controller should be properly configured to run well in your application.

A simple cycle-test script is included with HiDS, and the CycleTest.txt is located in the Windows ProgramData directory, which is Windows-OS-version specific. In Windows 7, this directory is in C:\ProgramData. In this directory, there is a \ESI Motion\HiDS\{version} directory (where version is the HiDS version). This default file should not be edited.

Once the Controller is properly configured, click HiDS->Run Test Script, and the last script file is automatically loaded. If this feature has never been used, then the default CycleTest.txt file is loaded.

Initially the text file will be checked for valid syntax. The valid constructs are as follows:



This Document does not contain Technical Data or Technology as defined in the ITAR Part 120.10 or EAR Part 772.

Construct (multiple): **variable = numeric_value**
 variable = <object>
 variable = variable2 + (or - or * or /) numeric_value
 variable = <object> + (or - or * or /) numeric_value
 variable = variable2 + (or - or * or /) variable
 variable = <object> + (or - or * or /) variable

Tokens: A leading a-z local variable plus an equal sign indicates an assign action. This can be a simple assignment (x=y) or can be combined with an operator (+, -, *, or /) and an additional operand.

The first token must be a local variable (a-z)

The second token can be a numeric value that is a string representing an integer or real number, or it can be <object> (where <object> is any valid HiDS variable), or it can be another local variable.

The third token can be a plus, minus, multiply, or divide operator.

The fourth token can be a numeric value that is a string representing an integer or real number, or it can be another local variable

Syntax: All tokens must be separated by one space minimum. Capitalization is ignored for all tokens except <object>.

Examples: x = 10.2
 y = FeedbackRPM
 z = b / 10
 z = FeedbackRPM / 10
 a = b + c
 j = FeedbackRPM * k

Construct: **Break**

Tokens: Break keyword will immediately exit an active for-loop.

Syntax: All tokens must be separated by one space minimum. Capitalization is ignored.
 The Break keyword must occur between a For and EndFor statement.

Example: Break

Construct: **Endif**

Tokens: Endif keyword terminates the execution specified by the previous If condition.

Syntax: All tokens must be separated by one space minimum. Capitalization is ignored.
 An unterminated if-condition must precede the Endif.

Example: Endif

Construct: **Endfor**

Tokens: Endfor keyword will immediately exit an active for-loop.

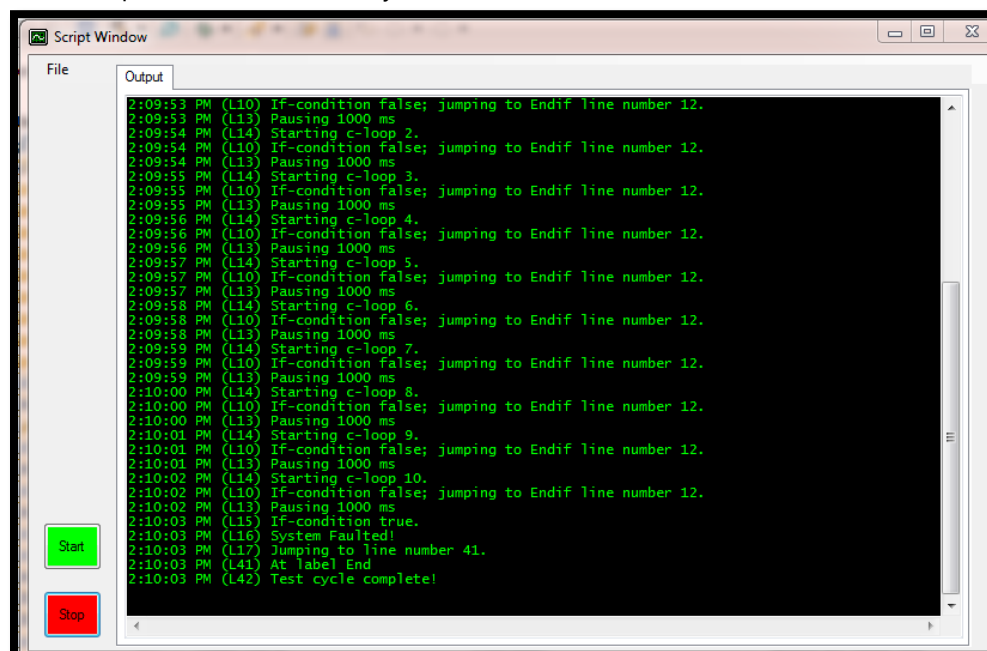
Syntax: All tokens must be separated by one space minimum. Capitalization is ignored.
 An active for-loop must precede the Endfor.

Example: Endfor

Construct:	For variable = x to y
Tokens:	For keyword will iterate the specified number of times through the following lines of codes until an endfor keyword is found. variable can only be a single alpha character a through z. = is the only valid symbol after variable x is the starting value for the for-loop to is the required fifth token y is the ending value for the for-loop
Syntax:	All tokens must be separated by one space minimum. Capitalization is ignored. Index must be uniquely used for this for-loop in this file. y must be greater than x; both must be integers The nested depth of for-loops cannot exceed 10.
Example:	For c = -1 to 10
Construct:	Goto label
Tokens:	Goto keyword will immediately transfer execution to the line specified by the label token. Label must be a unique and valid label in this file.
Syntax:	All tokens must be separated by one space minimum. Capitalization is ignored.
Example:	Goto End
Construct:	If variable (or <object>) = (or < or >) numeric_value (or variable)
Tokens:	If keyword will verify the specified condition, and if true, will execute the following lines of codes until an endif keyword is found. The second token can be a variable (a single alpha character a through z), or it can be <object> which is any valid HiDS variable. =, >, or < are the only valid symbols for If The fourth token can be a numeric_value, and must be a string that represents an integer or real number; or it can be a local variable (a through z).
Syntax:	All tokens must be separated by one space minimum. Capitalization is ignored for all tokens except <object>. Variable must be uniquely used for this if-condition in this file. y must be greater than x. The nested depth of for-loops cannot exceed 10.
Examples:	If z > 10 If FeedbackRPM < 500 If a = b If FeedbackRPM > k
Construct:	:Label
Tokens:	Label keyword can be used to separate code statements or can be used as a destination for a Goto.
Syntax:	All tokens must be separated by one space minimum. Capitalization is ignored. Must start with a :
Example:	:End
Construct:	Pause numeric_value
Tokens:	Pause keyword will suspend execution for the specified number of milliseconds. numeric_value must be a string that represents an integer.
Syntax:	All tokens must be separated by one space minimum. Capitalization is ignored.
Example:	Pause 1000

Construct:	Print string + <object>
Tokens:	Print keyword will print a blank line or display the value of a HiDS variable, a string, or a combination of both. <object> is any valid HiDS variable. string is a printable string. + is the only valid symbol for Print.
Syntax:	The Print keyword alone will print a blank line on the script window. All tokens must be separated by one space minimum. Capitalization is ignored for all tokens except <object>. There are multiple valid constructs for Print. 3 examples are shown below.
Example:	Print Print "System Faulted!" Print "1KRPM not reached!, last value = " + FeedbackRPM Print FeedbackRPM + "RPM reached!"
Construct:	Set <object> = numeric_value (or variable)
Tokens:	Set keyword will perform a HiDS write to update an object value. <object> is any valid HiDS variable. = is the only valid symbol for Set. The fourth token can be a local variable (a-z) or can be numeric_value. It must be a string that represents an integer or real number.
Syntax:	All tokens must be separated by one space minimum. Capitalization is ignored for all tokens except <object>.
Example:	Set MaVL.RPMCommand = 1000
Construct:	//
Tokens:	// keyword allows file comments to be added.
Syntax:	All characters on the line after the // are ignored.
Example:	//This script runs an overnight test

The following shows the output from the default CycleTest.txt:



6 Motor Feedback

Motor vector control requires precise knowledge of the actual rotor phase, so the alignment (phase-offset) between the rotor and the physical feedback (resolver, encoder, etc) must be known. Not all controllers support all of the following feedbacks (potentially due to connector limitations).

In addition, some combinations of mixed feedbacks (i.e. MotorA on Quadrature Encoder and MotorB on Serial Encoder) may not be supported by all products. While Resolver feedback can likely be mixed with a different feedback, Sensorless and the various encoder feedbacks share CPU resources, and may not be mixed.

6.1 Resolver

Description: A resolver is a type of rotary electrical transformer used for measuring degrees of rotation. It is considered an analog device.

Error Detection: The square-root of the sum of the square of the sine and cosine inputs $\sqrt{(\text{sine}^2 + \text{cosine}^2)}$ must be greater than `MxLossOfFeedback.limit`, otherwise a `MotorLossOfFeedback` error occurs.

Troubleshooting: The separate Sine and Cosine inputs can be verified by observing the raw measurements on the Analog page. The Sine input is shown by `Mx.ResolverASinAvg`, and Cosine input is shown by `Mx.ResolverACosAvg`. As the rotor shaft is rotated, one input should maximize while the other input minimizes. If resolver-feedback is selected, this should result in the `ResolverX.Degrees` (shown on the Resolver page) rotating from 0 to 360 degrees as the rotor is rotated one revolution.

6.2 Quadrature Encoder

Description: An encoder is an electromechanical device that can measure motion or position. Most encoders use optical sensors to provide electrical signals in the form of pulse trains, which can, in turn, be translated into motion, direction, or position.

Error Detection: None; all bit-patterns are valid, therefore there is no invalid bit-pattern to detect as an error condition.

Troubleshooting: There are 3 inputs used for the quadrature encoder: A, B, and I(index). If the encoder input is suspect, rotate the motor by hand or using Manual Feedback, and observe the `EncoderEQEP1aIn`, `EncoderEQEP1bIn`, and `EncoderEQEP1IndexIn` (or the QEP2 equivalents for MotorB) inputs on the Encoder page. As the motor shaft rotates, all 3 of these should alternate between 0 and 1. If `QuadEncoder-feedback` is selected, this should result in the `EncoderX.Degrees` (shown on the Encoder page) rotating from 0 to 360 degrees as the rotor is rotated one revolution.

Note a characteristic of the quadrature encoder is the position is invalid on power-up until the index (0 degrees) is past. For this reason, often a Hall-feedback is connected in parallel, and the Hall-input “seeds” the quadrature-encoder position on power up. This is enabled by setting `QepXData.UseHallForStartup = 1`.

6.3 Manual Feedback

Description: Manual Feedback is an open-loop test mode that allows simple, often unloaded, motor operation. Manual Feedback is useful to troubleshoot physical feedback (encoder, resolver, hall, etc.) problems, or other basic motor-control problems.

Error Detection: None; Manual Feedback is open-loop.

Troubleshooting: If Manual feedback cannot spin the motor, check the 3 phase currents during motor operation. Refer to section **8.5**. for additional information.

6.4 Sensorless Feedback

Description: The Sensorless Feedback estimates the rotor position based on the measured phase currents and a mathematical motor model.

Error Detection: Limited; the Sensorless algorithm will produce an estimated rotor angle with any non-zero current measurements. An incorrect Sensorless configuration will often lead to an overcurrent fault.

Troubleshooting: Refer to section 4.7 or additional information.

6.5 Hall Feedback

Description: For Hall Feedback, the rotor position is sensed using Hall-effect sensors embedded into the stator.

Error Detection: There are 6 valid bit combinations on the 3 binary Hall inputs – all 0's and all 1's are invalid, so that loss-of-feedback error indicates the Hall inputs are reading as all zeros or all ones.

Troubleshooting: There are 3 inputs used for the Hall Sensor: U, V, and W. If the Hall input is suspect, rotate the motor by hand or using Manual Feedback, and observe the HallX.U, HallX.V, and HallX.W inputs on the Hall page. As the motor shaft rotates, all 3 of these should alternate between 0 and 1. If Hall-feedback is selected, this should result in the HallX.Degrees (shown on the Hall page) rotating from 0 to 360 degrees (in 60-degree increments) as the rotor is rotated one revolution.

6.6 Serial Encoder Feedback

Description: Serial Encoders are optical sensors providing an absolute position in a digital stream.

Error Detection: The absence of a serial encoder data stream, or a CRC-error on a received packet will produce a MotorLossOfFeedback error.

Troubleshooting: The SerialEncoderXData.NumErrorPackets should not be incrementing while the SerialEncoderX.Degrees is showing a non-zero value. If not, check that there is a clock and data bit-stream on the Serial Encoder IO pins. Also, the raw-data received from the serial encoder is displayed in variables SerialEncoderAData.ReceiveWord0-15. The location of the first packet-bit may not be in the receive-word0, as the serial-encoder protocol allows for the packet to be shifted in time. For trouble-shooting purposes, you could verify these words are changing while the serial-encoder is connected.

6.7 Sin/Cos Feedback

Description: Sinusoidal encoders encode position information by providing a pair of quadrature sine and cosine signals as the shaft is rotated.

Error Detection: Similar to a Resolver feedback, the square-root of the sum of the sine and cosine inputs must be greater than MxLossOfFeedback.limit, otherwise a MotorLossOfFeedback error occurs.

Troubleshooting: The separate Sine and Cosine inputs can be verified by observing the raw measurements on the Analog page. The Sine input is shown by Mx.ResolverASinAvg, and Cosine input is shown by Mx.ResolverACosAvg. As the rotor shaft is rotated, one input should maximize while the other input minimizes. If resolver-feedback is selected, this should result in the ResolverX.Degrees (shown on the Resolver page) rotating from 0 to 360 degrees as the rotor is rotated one revolution.

6.8 Endat Feedback

Description: EnDat stands for Encoder Data, and The EnDat interface is a digital, bidirectional interface for encoders; essentially a different type of serial encoder. Due to the physical time duration of Endat encoder communication, only single-axis (MotorA only) products support Endat and only up to a 10Khz PWM rate.

Error Detection: The absence of an EnDat data stream or a CRC-error on a received packet will produce a MotorLossOfFeedback error.

Troubleshooting: The EndatARawData.PacketErrorCount should not be incrementing while the EnDatEncoderFeedbackA.Degrees is showing a non-zero value. If not, check that there is a clock and data bit-stream on the EnDat IO pins.

6.9 Secondary Feedback

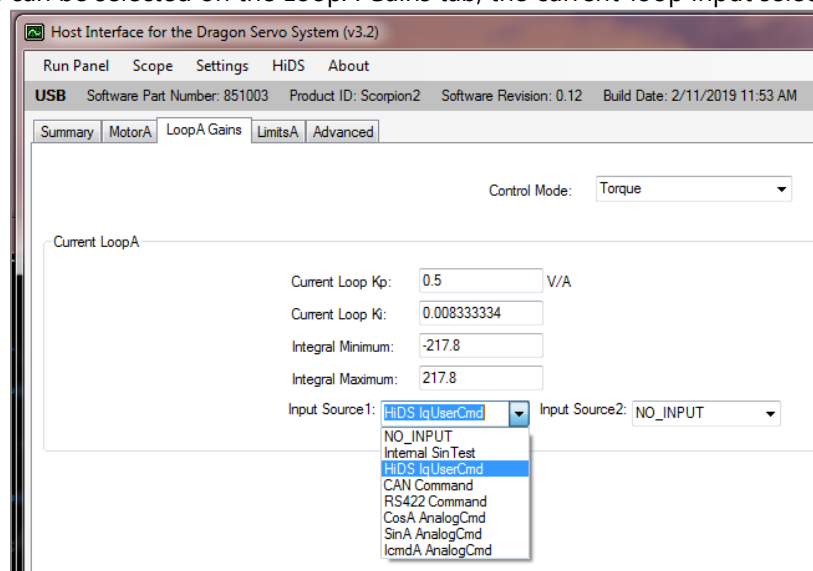
Some Controllers allow a secondary/alternate feedback to be used for the position-loop control. This is especially useful if there is gearing between the motor-rotor and the actual spinning mechanism. If the Controller supports multiple feedback inputs, then an available feedback can be selected for the position-loop feedback.

Note there are some restrictions for selecting multiple feedback. For example, no secondary position-loop feedback is available for a primary feedback of sensorless or manual (open-loop) feedback, because these feedbacks do not allow precisely stopping a motor at a specific position. In addition, some Controller hardware multiplexes some feedback inputs (like using the same inputs for Hall and Quadrature Encoder); refer to the Controller installation manual for details on feedback connectivity.

7 Control Loop Inputs

Every control loop has two MUX inputs selectable by drop down menus in the Loop Gains page. These two selectable inputs are then summed to produce the control input for the desired loop. For a diagram of these MUX inputs, see section 9.2.1 for the IQ command determination, section 9.3.1 for the RPM command determination, and section 9.4.1 **Error! Reference source not found.** for the radians command determination.

Generally, the loop-inputs can be selected on the LoopA Gains tab; the current-loop input selections are shown below.



7.1 Sine Test Command

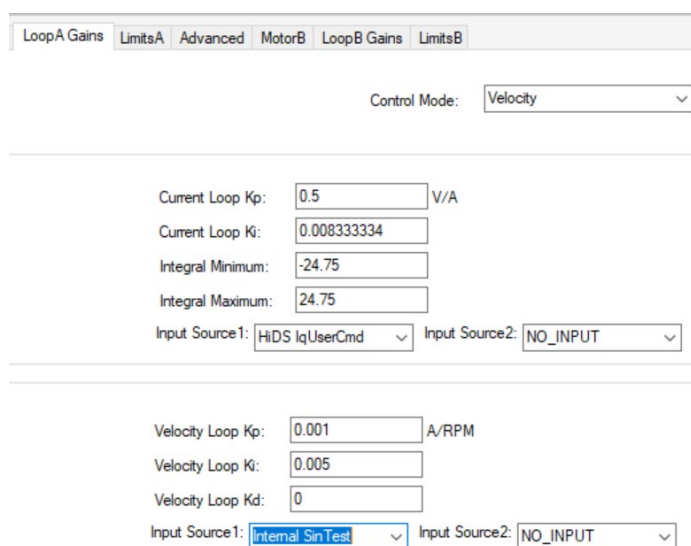
From the Utility Test page, the always running SinTest.output is a sinusoidal value between 0 and 1, with a frequency set by SinTest.frequency. Sine Test supports single frequency, swept sine, and square-wave outputs. Sine Test can be used as an input to any loop (Current, Velocity, Position). The test input amplitude to the current loop can be set via following variables depending on which loop is active:

- SinIqAmplitude (for the Torque/Current loop)
- SinVelAmplitude (for the Velocity loop)
- SinPosAmplitude (for the Position loop)

These 3 variables produce the 3 possible loop inputs as follows:

- SinIqTestOutput = SinTest.output * SinTest.iq_amplitude
- SinVelTestOutput = SinTest.output * SinTest.VelocityCommandAmp
- SinPosTestOutput = SinTest.output * SinTest.position_amp

As an example, to select a 10RPM sinusoidal input at 1Hz as an input to the velocity loop, one would set SinTest.frequency = 1 (hertz) and SinVelAmplitude = 10 (RPM) and select the Sine Test input to the velocity loop, as shown below.



LoopA Gains LimitsA Advanced MotorB LoopB Gains LimitsB

Control Mode: Velocity

Current Loop Kp: 0.5 V/A

Current Loop Ki: 0.008333334

Integral Minimum: -24.75

Integral Maximum: 24.75

Input Source1: HiDS IqUserCmd Input Source2: NO_INPUT

Velocity Loop Kp: 0.001 A/RPM

Velocity Loop Ki: 0.005

Velocity Loop Kd: 0

Input Source1: Internal SinTest Input Source2: NO_INPUT

7.2 User Command

The user command is input via the HiDS Run Panel. The variable updated varies on which mode is enabled:

- Torque mode, Mx.IqUserCommand
- Velocity mode, MxVL.RPMUserCommand
- Position mode, MxPL.RadiansUserCommand

7.3 CAN Command

The servo supports an ESI Motion defined CAN protocol that provides the users with the ability to command the servo in torque, velocity and position mode. Please refer to document 100211 CAN Protocol for more details.

7.4 Serial Command

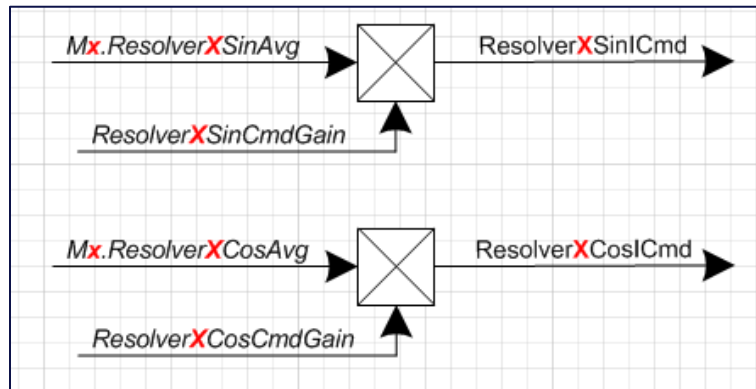
Similar to CAN, the servo also supports serial protocol that provides the users with the ability to command the servo in torque, velocity and position mode. Please refer to document 100121 RS422 Protocol for more details.

7.5 Resolver Commands

Both resolver inputs for Motor A and B can be reassigned to be control loop inputs. The SIN and COS signals can be used as independent commands, resulting in a total of 4 possible analog inputs (2 for MotorA and 2 for MotorB):

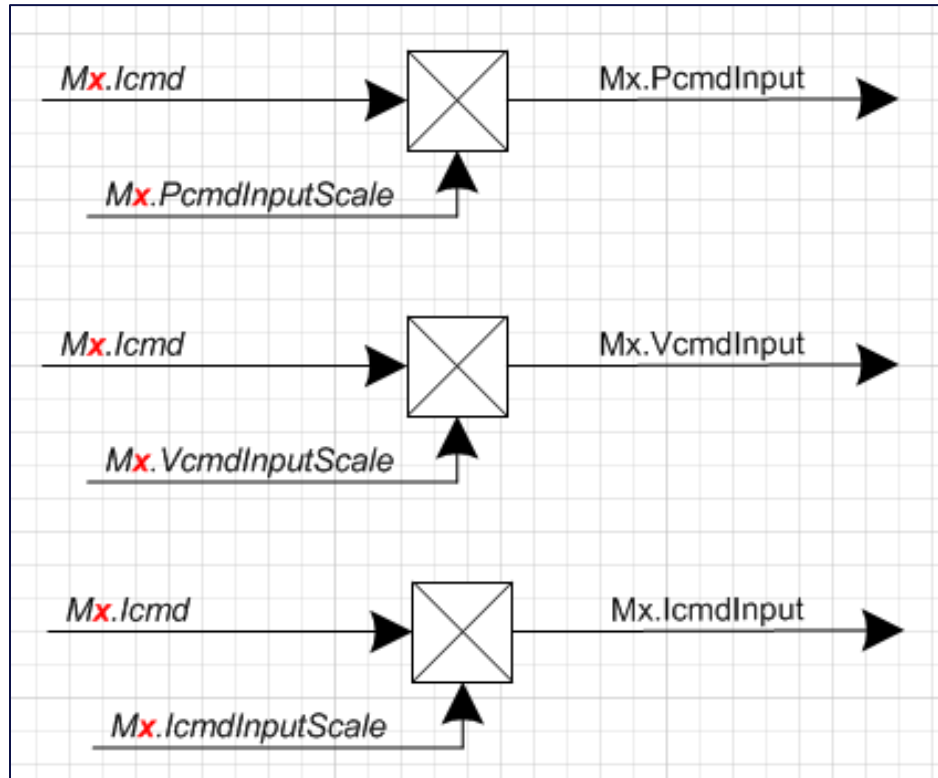
- Resolver MA COS
- Resolver MA SIN
- Resolver MB COS
- Resolver MB SIN

The inputs can be scaled using their corresponding ResolverXSinGain and ResolverXCosGain variables located in the LoopInputs page. The relationship between the raw input, the gain, and the resulting loop input is shown pictorially as follows:



7.6 Analog Commands

In the typical controller configuration, there are two allocated analog input signals (Ma.Icmd and Mb.Icmd) provided to command the servo. The Mx.IcmdInputScale, MxVL.VcmdInputScale, and MxPL.PcmdInputScale variables can be used to change the input gain (for Torque-mode, Velocity-mode, and Position-mode respectively). The relationship between the raw input, the gain, and the resulting loop input is shown pictorially as follows:



7.7 Loop Filters

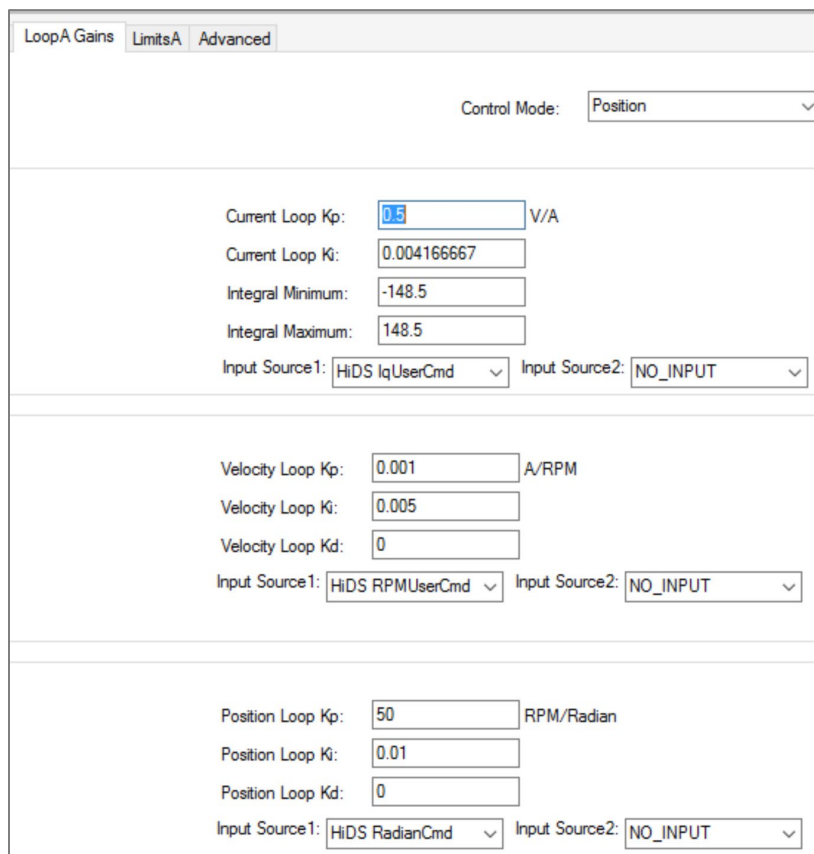
Todo: diagrams need to be updated showing the filter placement and configuration example?

7.8 Configuration Examples

7.8.1 HiDS Control

If accurate motor position is required, the Control-mode on the LoopA Gains tab must be set to Position. If accurate motor velocity is required, the Control-mode on the LoopA Gains tab must be set to Velocity. If accurate motor torque is required, the Control-mode on the LoopA Gains tab must be set to Torque; note current-loop and torque-loop are used interchangeably here, because the difference between the current-output and the motor-torque is a constant (the Motor Torque-constant K_t). The Control Mode is selected on the LoopA Gains as shown below.

HiDS is the default command interface for the current, velocity, and position loops; the HiDS-control configuration can be verified on the LoopA Gains tab by setting the desired Input Source1 to HiDS IqUserCmd (for the current/torque-loop), to HiDS RPMUserCmd (for the velocity-loop), and to HiDS RadianUserCmd (for the position-loop), as follows:



Note the Input Source2 is only used if 2 command-sources should be summed; for example, summing a sine-wave to a static command. This Input Source2 is typically set as NO_INPUT.

For current-loop or velocity-loop control, open the default Run Panel in HiDS. For current/torque-control, enter an **IqUserCommand** in the text-box and click [Run] – note there is **no velocity-control** in torque-mode, so the motor will spin as fast as it is capable at the applied voltage.

For velocity-loop or velocity-loop control, open the default Run Panel in HiDS. For velocity-control, enter an **RPMUserCommand** in the text-box and click [Run]; a properly tuned motor should accelerate at the Ma.AccelRPM (per second) up to the commanded RPM.

For position-loop or velocity-loop control, open the position-loop Run Panel in HiDS (Select HiDS->Change Run Panel1 Config file from..., and select file RunPanelSettingsPosition.xml). For position-loop-control, enter a **RadianUserCommand** in the text-box and click [Run]; a properly tuned motor should accelerate at MaPL.AccelRadiansPerSec (on the Position page) to the commanded radians position.

7.8.2 RS422 Control

The ESI RS422 Command/Status protocol is specified in ESI document 100121, which is available on ESI Motion.com at <https://www.esimotion.com/support/downloads/>.

The RS422 command interface must first be selected for the desired loop-input. For example, to set the position-loop command source as RS422, the LoopAGains tab would be set as follows:



This Document does not contain Technical Data or Technology as defined in the ITAR Part 120.10 or EAR Part 772.

LoopA Gains
LimitsA
Advanced

Control Mode:
Position

Current Loop Kp:
0.5
V/A

Current Loop Ki:
0.004166667

Integral Minimum:
-148.5

Integral Maximum:
148.5

Input Source1:
HiDS IqUserCmd
Input Source2:
NO_INPUT

Velocity Loop Kp:
0.001
A/RPM

Velocity Loop Ki:
0.005

Velocity Loop Kd:
0

Input Source1:
HiDS RPMUserCmd
Input Source2:
NO_INPUT

Position Loop Kp:
50
RPM/Radian

Position Loop Ki:
0.01

Position Loop Kd:
0

Input Source1:
RS422 Command
Input Source2:
NO_INPUT

Each-loop's input source is independent, so if velocity-control was required, the Velocity-loop input source would be set to RS422 Command.

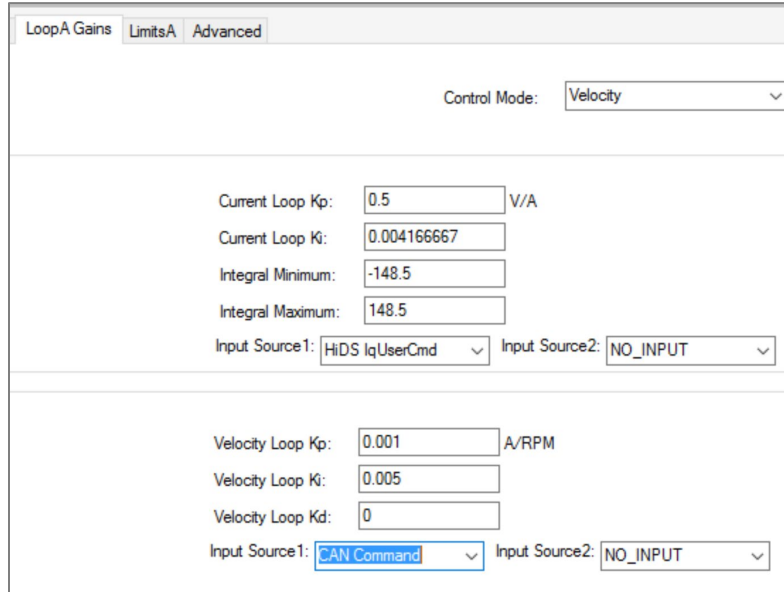
Note the Input Source2 is only used if 2 command-sources should be summed; for example, summing a sine-wave to a static command. This Input Source2 is typically set as NO_INPUT.

Sample current, velocity, and position-loop RS422 commands are described in the ESI document 100121, as is troubleshooting of the RS422 interface.

7.8.3 CAN Control

The ESI CAN Command/Status protocol is specified in ESI document 100211, which is available on ESI Motion.com at <https://www.esimotion.com/support/downloads/>.

The CAN command interface must first be selected for the desired loop-input. For example, to set the velocity-loop command source as CAN, the LoopAGains tab would be set as follows:



Each-loop's input source is independent, so if current-control was required, the Current-loop input source would be set to CAN Command.

Note the Input Source2 is only used if 2 command-sources should be summed; for example, summing a sine-wave to a static command. This Input Source2 is typically set as NO_INPUT.

Sample current, velocity, and position-loop CAN commands are described in the ESI document 100211, as is troubleshooting of the CAN interface.

8 Motor Control Examples

This section provides examples of the various ways to control the ESI Controller. This section assumes the motor-feedback phase has been calibrated (see section 4.4) and that the current and velocity loops have been tuned (see sections 4.3 and 4.5)

8.1 Analog Velocity Control with a Digital Motor Enable Input

An analog voltage input can be used as a command input into any of the 3 control loops (torque, velocity, or position). Different controller models may have a different number of analog-input connections, but the typical connections are shown below.

Mite / Scorpion / Atom:

The MotorA and MotorB sides each have 3 analog inputs referred to as:

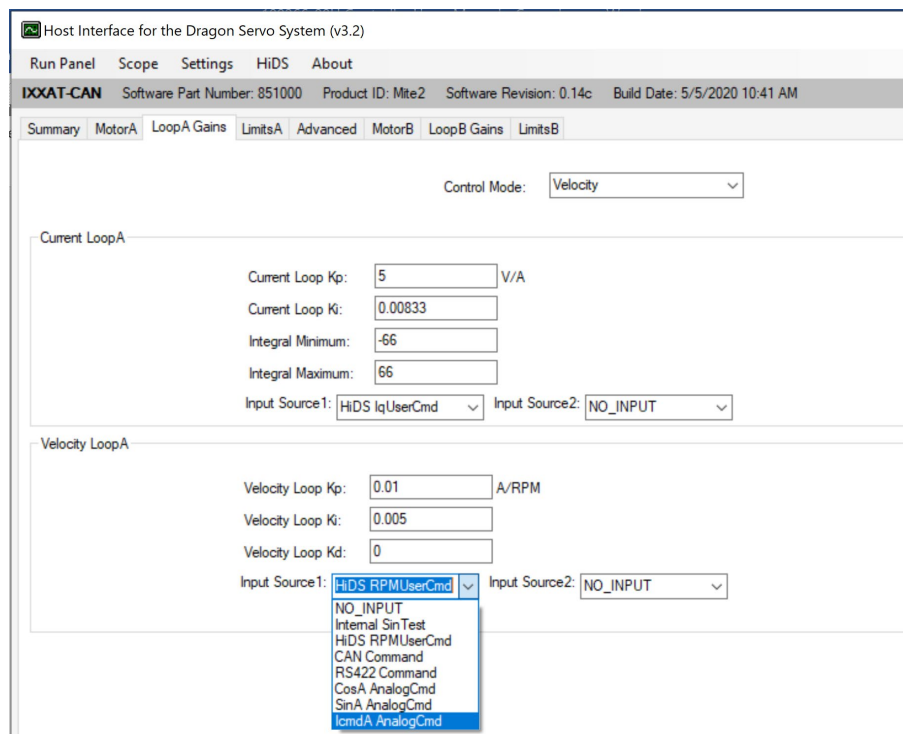
- CMD+_MA / CMD-_MA for MotorA, CMD+_MB / CMD-_MB for MotorB
- SIN+_MA / SIN-_MA for MotorA, SIN+_MB / SIN-_MB for MotorB
- COS+_MA / COS-_MA for MotorA, COS+_MB / COS-_MB for MotorB

Dragon:

The Dragon controller has only 1 differential analog input per motor-side.

- CMD+_MA / CMD-_MA for MotorA, CMD+_MB / CMD-_MB for MotorB

The loop-input can be selected on the loop Gains tab within HiDS via the Input Source1 dropdown list. The following example shows the selection of the CMD_MA input for the MotorA velocity-loop. Note the current-loop and position-loop inputs have their own independent dropdown lists.



8.1.1 Connection / Configuration Verification

- To check basic voltage connectivity:
 - On Mite/Scorpion, apply 2 volts between the CMD+_MA / CMD-_MA pins.
 - Connect to the controller via HiDS, and on the Advanced tab, Analog page. The variable **Ma.IcmdRaw** should read ~2 volts. If it does not, verify the analog-voltage connectivity to the controller.
- To verify command scaling:
 - With motor-control disabled (motor not running),
 - For current-loop control, the 2volts shown by variable **Ma.IcmdRaw** is multiplied by **Ma.IcmdInputScale** (default = 1); this result sets variable **Ma.IcmdInput**, which is the Amps command input to the current loop.
 - For velocity-loop control, the 2volts shown by variable **Ma.IcmdRaw** is multiplied by **MaVL.VcmdInputScale** (default = 1); this result sets variable **MaVL.VcmdInput**, which is the RPM command input to the velocity loop.
 - For position-loop control, the 2volts shown by variable **Ma.IcmdRaw** is multiplied by **MaPL.PcmdInputScale** (default = 1); this result sets variable **MaPL.PcmdInput**, which is the Radians command input to the position loop.
- To verify command multiplexer setting:

- a. With motor-control disabled (motor not running), if the loop-input is properly configured as shown in the Loop Gains screen-shot above:
 - i. For the current-loop, the scaled value of **Ma.IcmdInput** sets variable **Ma.IqCmdSrc1** (the current-loop Amps command), shown on the Loop Inputs page.
 - ii. For the velocity-loop, the scaled value of **Ma.VL.VcmdInput** sets variable **Ma.VL.VelCmdSrc1** (the velocity-loop RPM command), shown on the Loop Inputs page.
 - iii. For the position-loop, the scaled value of **Ma.PL.PcmdInput** sets variable **Ma.PL.PosCmdSrc1** (the position-loop radians command), shown on the Loop Inputs page.

8.2 Digital Input Control

Digital inputs / outputs can also be used for simple command and status. As an example, we want to control a motor-velocity in either +500RPM or -500RPM directions.

- DigitalInput1 is assigned as the motor-enable
- DigitalInput2 is assigned as the motor-speed / direction.
- DigitalOutput1 indicates the motor is running or not.
- DigitalOutput2 indicates the controller is ready or in a fault state.

This example can be accomplished as follows (all within HiDS):

1. On the Summary page, right-click on variable EnableMotorA, and select Send to Digital IO -> Digital In1.
 - a. On the Digital In tab, the default states will be used: A logic low input will set this variable to 0 (motor disabled), and a logic high input will set this variable to 1 (motor enabled).
2. On the VelocityLoop page, right-click on variable MaVL.RPMUserCommand, and select Send to Digital IO -> Digital In2.
 - a. On the Digital In tab, the "LoValue" will be set to -500 and the "HiValue" will be set to +500. This will set the motor speed to -500 when the input is a logic low, and the motor speed to +500 when the input is a logic high.
3. On the Control page, right-click on variable MotorAState, and select Send to Digital IO -> Digital Out1.
 - a. On the Digital Out tab, the threshold is set to 9.5, because MotorAState = 10 when the motor-state machine is fully running. When MotorAState=10, this output is a logic high. Anything lower than 10, and the output will be a logic low.
4. On the Control page, right-click on variable SystemFaultState, and select Send to Digital IO -> Digital Out2.
 - a. On the Digital Out tab, the default threshold of 0.5 can be used. If the controller is in a fault-state, this output is a logic high. If the controller is ready (not in a fault state), the output will be a logic low.

8.3 RS422 Control

Command and status-packet examples are included in ESI RS422 Protocol document (100121) available on ESI Motion.com.

8.4 CAN Control

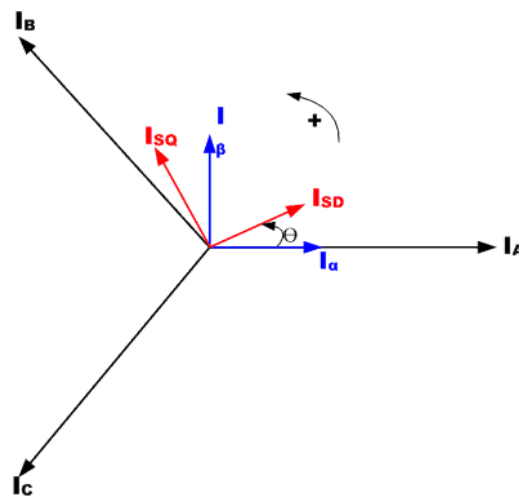
Command and status-packet examples are included in ESI CAN Protocol document (100211) available on ESI Motion.com.

9 Theory of Operation

This section describes theory of operations of the ESI Servo Controller current-loop and velocity-loop. It is the objective that the reader, with adequate knowledge of motor operations, could configure the motor and monitor its performance via HiDS.

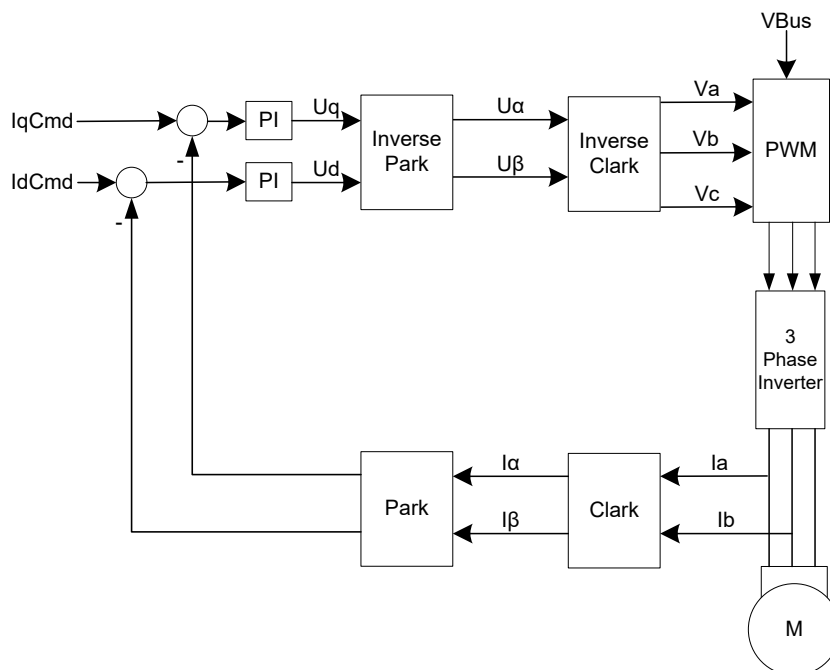
9.1 The Current Loop

In summary, the alternating phase-currents are measured by the Controller. Through the Clark and Park mathematical transformations using the rotor position, the alternating-currents are transformed to the direct / non-cyclical I_Q (torque vector) and I_D (flux vector), which can be controlled via a standard PID (Proportional Integral Derivative) compensator, which produces the voltage vectors U_Q and U_D . U_Q and U_D are then mathematically transformed to the 3 alternating phase voltages via the inverse Clark and Park transformations. The I_Q and I_D vectors are 90° out of phase as shown:



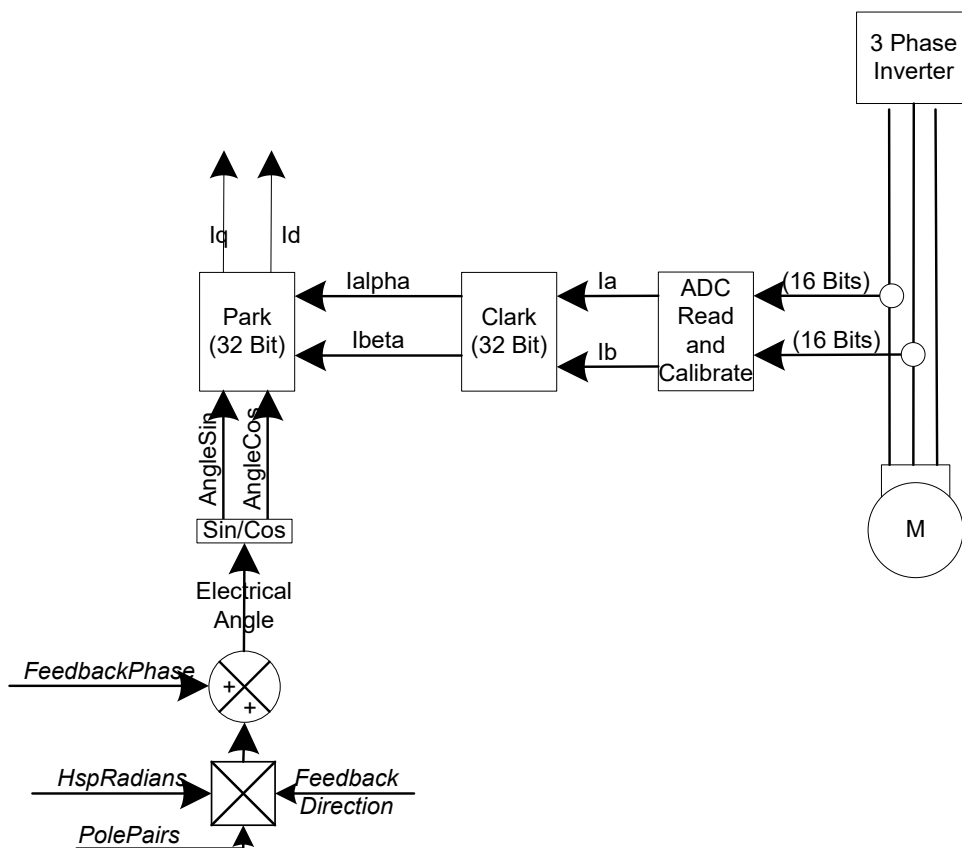
9.1.1 Clark and Park Transforms

An in-depth description of vector-control of current is not described here. However, a simplistic view of the current loop is shown below.



For simplicity, the current loop can be divided into sections below.

9.2 Clark and Park Transforms



For each motor controlled, the currents $Mx.Ia$ and $Mx.Ib$ (alternately the U and V phase) are measured. It is not necessary to measure $Mx.Ic$ (W phase) as it is assumed that $Ia + Ib + Ic = 0$. Through 2 mathematical transformations, called the Clark transform and the Park transform, the time-varying measured currents are transformed into a time-invariant reference frame that is rotating with the electrical angle.

The output from the Clark transform is $Mx.Ialpha$ and $Mx.Ibeta$, and in addition the Park transform requires precise knowledge of the electrical angle between the torque and flux vectors; this is accomplished via a mechanical measurement of the angle between the rotor and stator via a physical feedback mechanism such as encoder or resolver, or an estimation via a sensor-less algorithm.

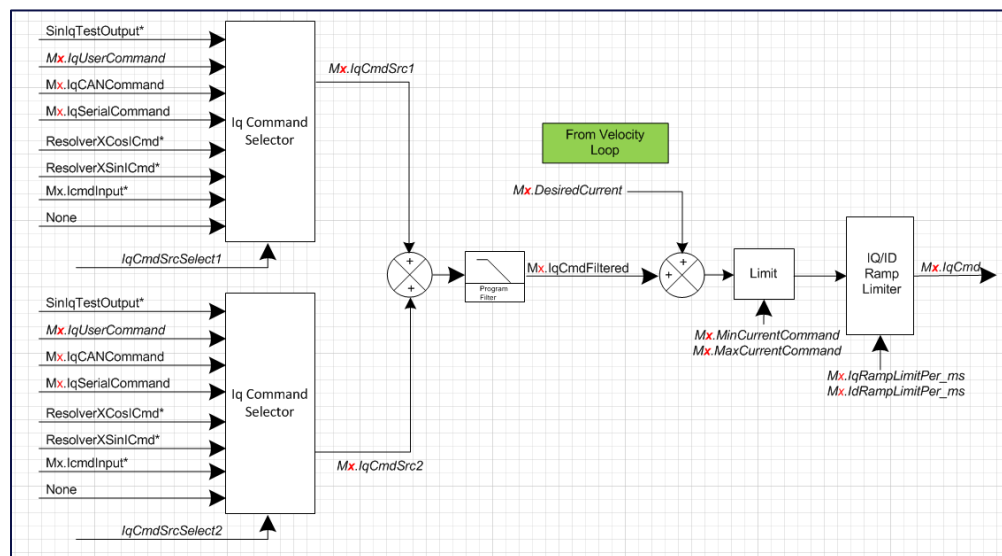
To achieve the proper electrical angle, the motor parameters must be properly configured with the Motor pole pairs. In addition, if a motor phase is swapped, the $MotorX.FeedbackDirection$ can be changed (-1 or +1) to swap the sign of the feedback.

Finally, if the feedback mechanism is not aligned with the motor, the $MotorX.FeedbackPhase$ must be entered to offset the feedback angle from the motor position. Refer to section 4.4 for details on motor phase calibration.

The output from the Park transform is the torque vector $Mx.Iq$ and the flux vector $Mx.Id$, which are the measured/calculated vectors in the rotating frame to be used in compensating for error from the desired $Mx.IqCmd$ and $Mx.IdCmd$.

9.2.1 IQ Command Determination

Each loop has 2 user-configurable input-source multiplexers, which are summed to produce a single loop input. Generally, only one input source would be selected, but the summing node allows (for example) a sinusoidal input to be summed with a DC input from another source. This $Mx.IqCmd$ output is the input to the current-loop diagrams shown in sections 9.1.1 and 11.

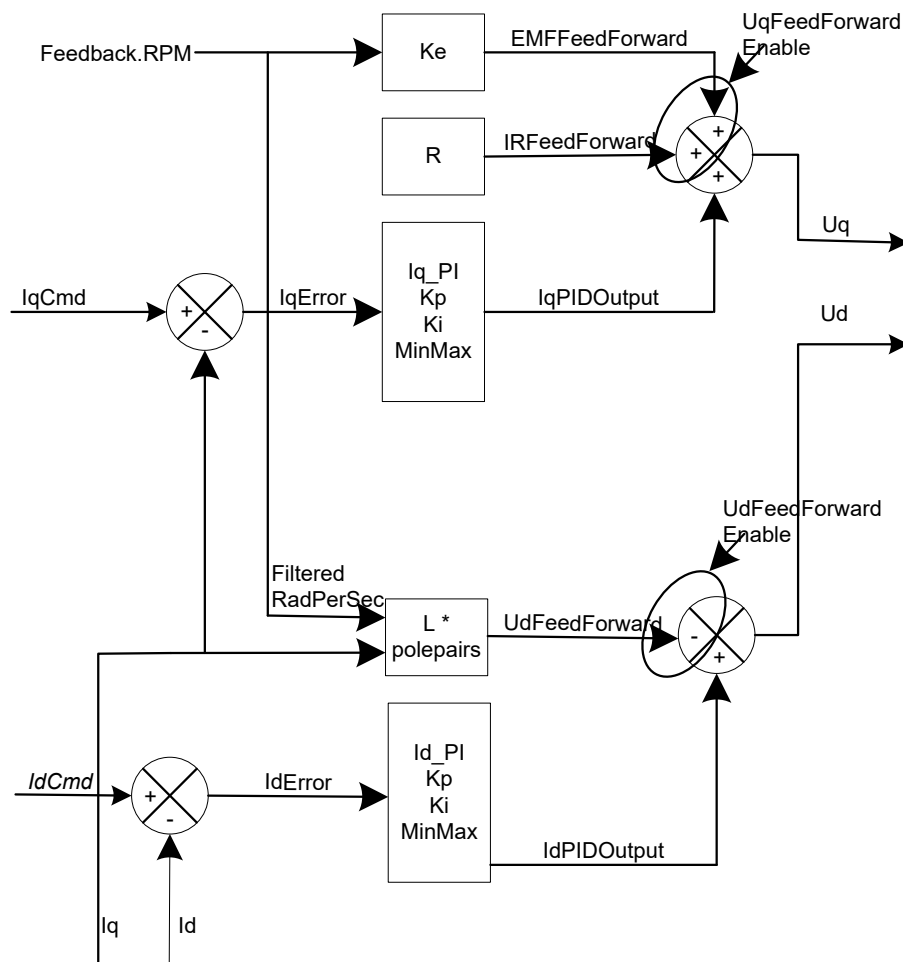


In the rotating reference frame, the resulting currents, $Mx.Iq$ and $Mx.Id$, are used as feedback to compare against the input IQ command ($Mx.IqCmd$), which is the torque component and the ID command ($Mx.IdCmd$), which is the flux component. Generally $Mx.IqCmd$ is the sole commanded current, and the $Mx.IdCmd$ is kept at zero to maximize torque.

The input current command into the current-loop, $Mx.IqCmd$, is the sum of 2 user selectable current commands ($Mx.IqCmdSrc1$ and $Mx.IqCmdSrc2$). Please refer to Appendix D for more detail. In addition, while running in Velocity mode, the Velocity-loop output produces an input ($Mx.DesiredCurrent$) to the Current Loop.

Note there is no automatic zeroing of these input variables, so for example if running in Velocity mode, you should insure the $Mx.IqUserCommand$ is zero (or it will be summed in).

The final $Mx.IqCmd$ output is then limited by the Configuration settings of $Mx.MinCurrentCommand$ and $Mx.MaxCurrentCommand$. Note these are software limits of the current commands. Exceeding these limits simply restrains the current command input to the control system – no error occurs if these limits are exceeded. $Mx.IqCmd$ is thus the desired IQ command to the control system, prior to error compensation.



The $Mx.Iq$ output is the torque component of the calculated Park transform from the measured $Mx.Ia$ and $Mx.Ib$ currents and the feedback (resolver, encoder, sensor-less) angle. Thus $Mx.IqError$ is the error between the desired IQ command and the resultant IQ, and it is the goal to reduce this IQ error to zero.

First the $Mx.IqError$ is reduced via a Proportional and Integral compensation using the configurations of $Mx.Kp$ (Gain), $Mx.Ki$ (Integral), and $Mx.IntegralMin$ and $Mx.IntegralMax$ (to limit the integral error buildup). Note if $Mx.AutoSetIntegralMinMax$ is set to 1 (true), the $Mx.IntegralMin$ and $Mx.IntegralMax$ are automatically set based on the maximum bus voltage.

Secondly there can be feed-forward error compensation to help counter the effects of motor-back-EMF-voltage (Mx.EMFFeedForward) and the effects of IR increases (Mx.IRFeedForward). Of course, these feed-forward error compensations rely on reasonably accurate configurations of the motor voltage constant MotorX.Ke and the motor phase resistance MotorX.Ohms. Typically this feed-forward is enabled (1), but it can be disabled by setting Mx.UqFeedForwardEnable = 0 (false).

The error-compensation of IQ results in an intermediate voltage of Mx.Uq.

Somewhat independent from the IQ error compensation, the ID error is reduced in a similar way. The Mx.Id output is the flux component of the calculated Park transform from the measured Mx.Ia and Mx.Ib currents and the feedback (resolver, encoder, sensor-less) angle. To maximize torque, the ID command, Mx.IdCmd, is typically set to zero. Thus Mx.IdError is the error between the desired ID command (zero) and the resultant ID, and it is the goal to reduce this ID error to zero.

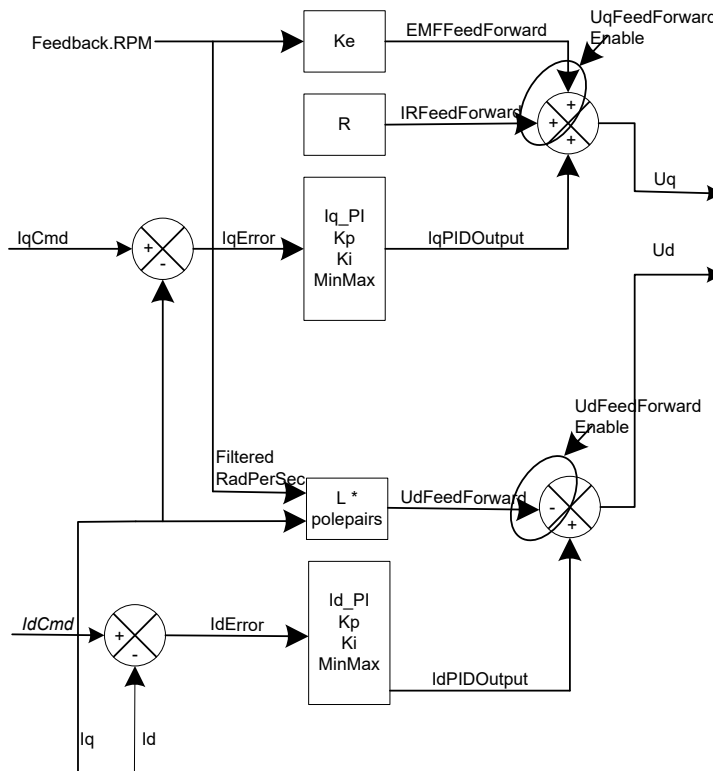
First the Mx.IdError is reduced via a Proportional and Integral compensation using the configurations of Mx.Kp (Gain), Mx.Ki (Integral), and Mx.IntegralMin and Mx.IntegralMax (to limit the integral error buildup).

Secondly there is feed-forward error compensation to help counter the effects of reactive losses (Mx.UdFeedForward) with increasing frequency. Of course, this feed-forward error compensation relies on reasonably accurate configurations of the motor phase inductance MotorX.Inductance. Typically this feed-forward is enabled (1), but it can be disabled by setting Mx.UdFeedForwardEnable = 0 (false).

The error-compensation of ID results in an intermediate voltage of Mx.Ud.

Note the unit change during this error correction process. The inputs to the error correction stage are currents, yet the outputs are in units of voltage. This is the unit transition point, which prepares us for the final output of (phase) voltages.

9.2.2 IQ and ID Error Compensation



The **Mx.Iq** output is the torque component of the calculated Park transform from the measured **Mx.Ia** and **Mx.Ib** currents and the feedback (resolver, encoder, sensor-less) angle. Thus **Mx.IqError** is the error between the desired IQ command and the resultant IQ, and it is the goal to reduce this IQ error to zero.

First the **Mx.IqError** is reduced via a Proportional and Integral compensation using the configurations of **Mx.Kp** (Gain), **Mx.Ki** (Integral), and **Mx.IntegralMin** and **Mx.IntegralMax** (to limit the integral error buildup). Note if **Mx.AutoSetIntegralMinMax** is set to 1 (true), the **Mx.IntegralMin** and **Mx.IntegralMax** are automatically set based on the maximum bus voltage.

Secondly there can be feed-forward error compensation to help counter the effects of motor-back-EMF-voltage (**Mx.EMFFeedForward**) and the effects of IR increases (**Mx.IRFeedForward**). Of course, these feed-forward error compensations rely on reasonably accurate configurations of the motor voltage constant **MotorX.Ke** and the motor phase resistance **MotorX.Ohms**. Typically this feed-forward is enabled (1), but it can be disabled by setting **Mx.UqFeedForwardEnable** = 0 (false).

The error-compensation of IQ results in an intermediate voltage of **Mx.Uq**. Somewhat independent from the IQ error compensation, the ID error is reduced in a similar way. The **Mx.Id** output is the flux component of the calculated Park transform from the measured **Mx.Ia** and **Mx.Ib** currents and the feedback (resolver, encoder, sensor-less) angle. To maximize torque, the ID command, **Mx.IdCmd**, is typically set to zero. Thus **Mx.IdError** is the error between the desired ID command (zero) and the resultant ID, and it is the goal to reduce this ID error to zero.

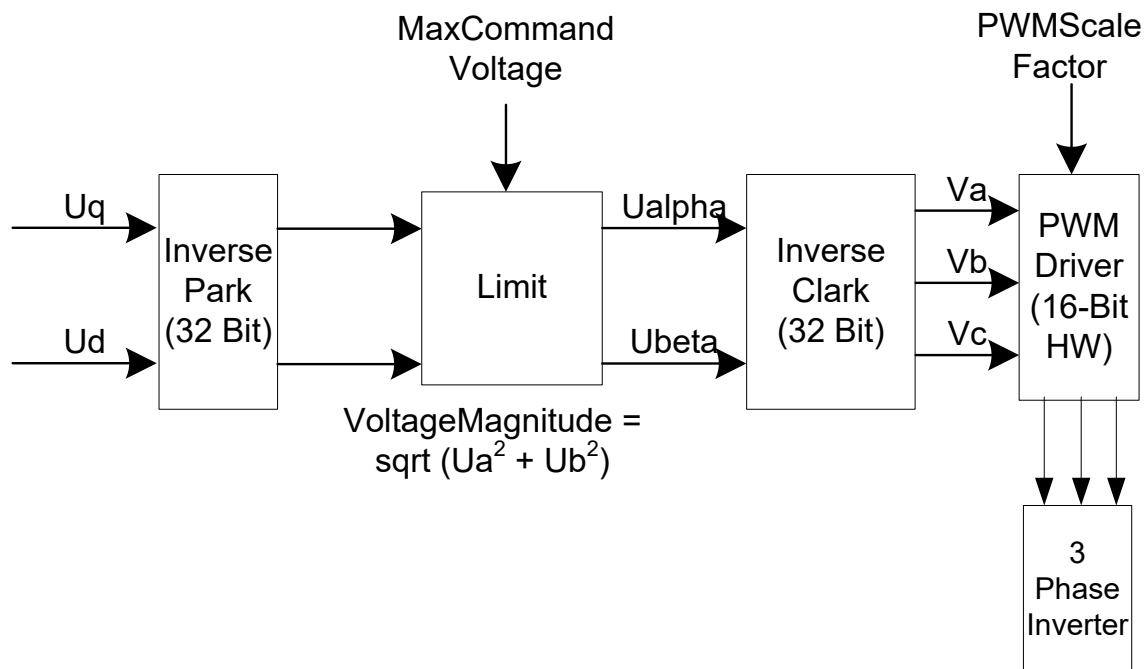
First the **Mx.IdError** is reduced via a Proportional and Integral compensation using the configurations of **Mx.Kp** (Gain), **Mx.Ki** (Integral), and **Mx.IntegralMin** and **Mx.IntegralMax** (to limit the integral error buildup).

Secondly there is feed-forward error compensation to help counter the effects of reactive losses (Mx.UdFeedForward) with increasing frequency. Of course, this feed-forward error compensation relies on reasonably accurate configurations of the motor phase inductance MotorX.Inductance. Typically this feed-forward is enabled (1), but it can be disabled by setting Mx.UdFeedForwardEnable = 0 (false).

The error-compensation of ID results in an intermediate voltage of Mx.Ud.

Note the unit change during this error correction process. The inputs to the error correction stage are currents, yet the outputs are in units of voltage. This is the unit transition point, which prepares us for the final output of (phase) voltages.

9.2.3 The Inverse Clark and Park



The error-compensated IQ and ID result in the intermediate voltages of Mx.Uq and Mx.Ud, which are fed into the Inverse Park transform.

To limit the bounds of the error correction, the Inverse Park voltage outputs, Ualpha' and Ubeta' (not readable via HiDS), are checked against a maximum obtainable command voltage. The voltage limit is MaxCommandVoltage, which is approximately equal to the VBus (Mx.VBus) input ÷ 2 (the actual divider depends on the hardware configuration, but this is very close). The voltage checked is the Uα and Uβ voltage vector scalar, Mx.VoltageMagnitude = $\sqrt{U_\alpha^2 + U_\beta^2}$. The resultant Mx.Ualpha and Mx.Ubeta outputs are the Ualpha' and Ubeta' Inverse Park transform outputs multiplied by the ratio of MaxCommandVoltage ÷ Mx.VoltageMagnitude. This "saturated" condition is often a warning to an improperly controlled motor or possibly an error-condition, and the condition is flagged by Mx.Saturated = 1. The precursor to saturation is indicated by Mx.PreSaturated = 1; the Controller can often tolerate pre-saturation for some time, but the saturated condition is effectively an error state. The VBus utilization can also be monitored via Mx.VBusUtilization, which should be between 0 and 1.

The saturation-limited Mx.Ualpha and Mx.Ubeta outputs from the Inverse Park transform are inputs to the Inverse Clark transform, which produces the motor phase voltages to apply, Mx.Va, Mx.Vb, and Mx.Vc; for a 3-level bridge application, the voltages are represented by Mx.VaRaw, Mx.VbRaw, and Mx.VcRaw.

The phase voltages are then applied to the motor via a pulse-width-modulated Vbus (ADCResults.VBus).

For the complete diagram, refer to Appendix B.

9.3 The Velocity Loop

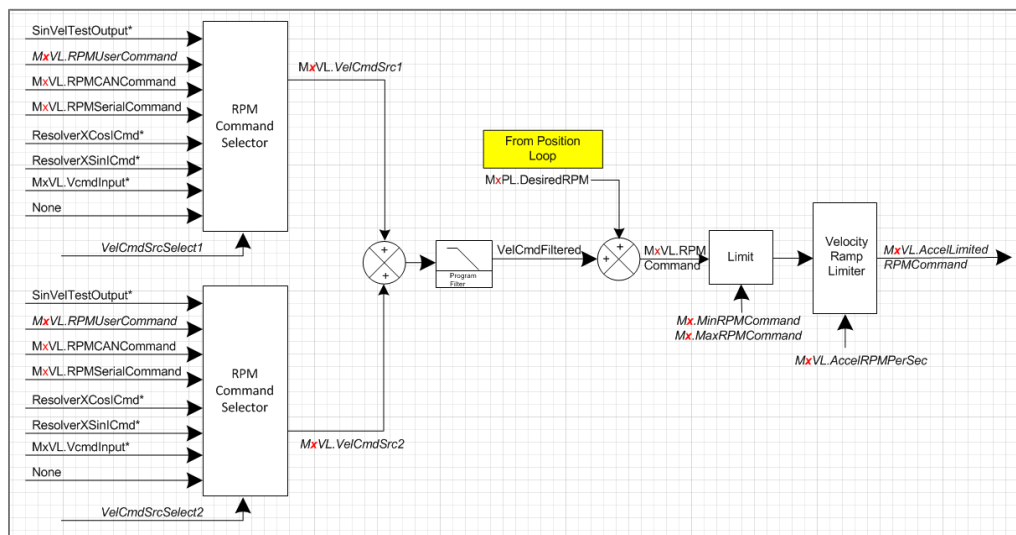
The Velocity loop is effectively an independent control loop outside the Current loop, which then inputs a current (Mx.DesiredCurrent) into the Current loop.

Once again for simplicity, the velocity loop can be divided into sections below.

9.3.1 RPM Command Determination

Each loop has 2 user-configurable input-source multiplexers, which are summed to produce a single loop input. Generally, only one input source would be selected, but the summing node allows (for example) a sinusoidal input to be summed with a DC input from another source. This MxVL.AccelLimitedRPMCommand output is the input to the velocity-loop diagram shown in section 12.

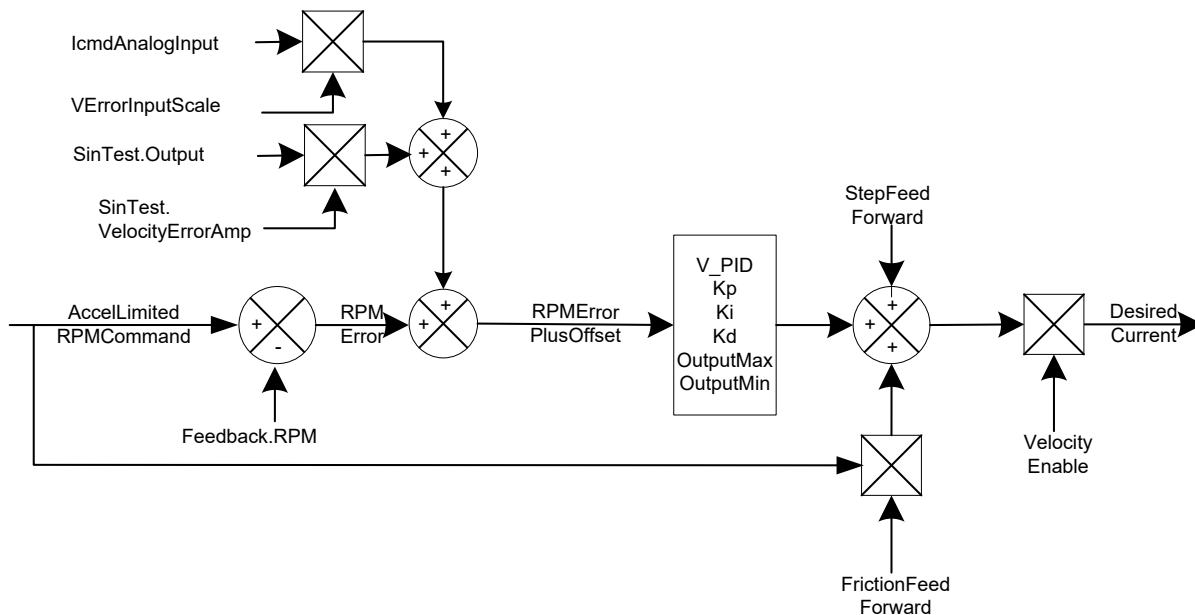
The preliminary



MxVL.RPMCommand output is the sum of 2 user selectable velocity commands (MxVL.VelCmdSrc1 and MxVL.VelCmdSrc2). In addition, while running in Position mode, the Position Loop output produces an input (MxPL.DesiredRPM) to the Velocity Loop.

The final MxVL.AccelLimitedRPMCommand output is then limited by the user Configuration setting of MxVL.AccelRPMPerSec. Note this is a software limit of the rate of change of the velocity command (acceleration / deceleration). Exceeding this limit simply restrains the velocity command input to the control system – no error occurs if this limit is exceeded. MxVL.AccelLimitedRPMCommand is thus the desired RPM command to the control system, prior to error compensation.

9.3.2 RPM Error Compensation



MxVL.RPMError is the error between the desired RPM command and the measured velocity via the available feedback (resolver, encoder, or sensor-less), and it is the goal to reduce this RPM error to zero.

First the MxVL.RPMError is reduced via a Proportional and Integral compensation using the configurations of MxVL.VelocityKp (Gain), MxVL.VelocityKi (Integral), and MxVL.VelocityIntegralMin and MxVL.VelocityIntegralMax (to limit the integral error buildup). Note if Mx.VelocityAutoSetIntegralMinMax is set to 1 (true), the Mx.VelocityIntegralMin and Mx.VelocityIntegralMax are automatically set to the maximum current limit.

Secondly there are feed-forward error compensation to help counter the effects of friction (MxVL.FrictionFeedForward), and the effects of inertia (MxVL.StepFeedForward). Typically, these feed-forwards are disabled (0), but they can be enabled by setting these values to non-zero.

Note the unit change during this error correction process. The inputs to the error correction stage are velocities, yet the outputs are in units of current. This is the unit transition point, which prepares us for the final output of (command) current.

Finally if MxVL.VelocityEnable = 0 (false), which is torque mode, the Mx.DesiredCurrent value is discarded. If MxVL.VelocityEnable = 1 (true), which is velocity mode, the Mx.DesiredCurrent value is summed into the Current loop input described above.

For the complete diagram, refer to Appendix C.

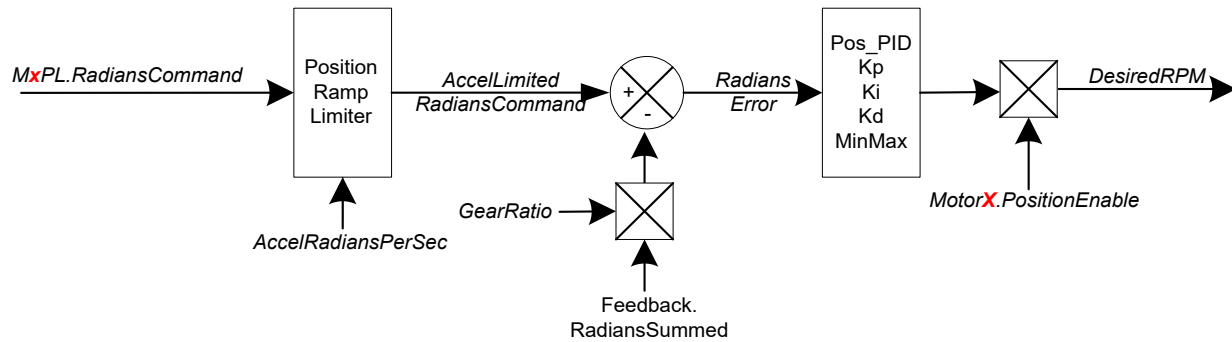
9.4 The Position Loop

The Position loop is effectively an independent control loop outside the Velocity loop, which then inputs an RPM (Mx.DesiredRPM) into the Velocity loop.

A block diagram of the position loop is shown below:

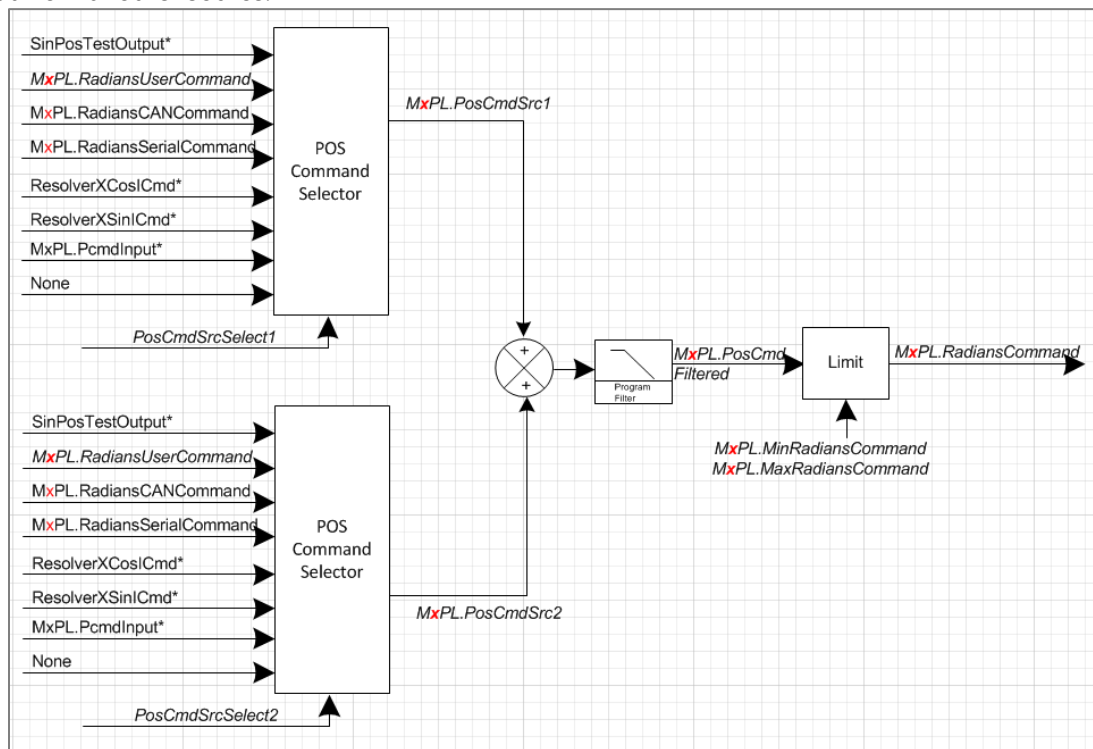


This Document does not contain Technical Data or Technology as defined in the ITAR Part 120.10 or EAR Part 772.



9.4.1 Radians Command Determination

Each loop has 2 user-configurable input-source multiplexers, which are summed to produce a single loop input. Generally only one input source would be selected, but the summing node allows (for example) a sinusoidal input to be summed with a DC input from another source.



9.5 Manual Feedback

Manual Feedback is an open-loop test mode that allows simple, often unloaded, motor operation. Manual Feedback is useful to troubleshoot physical feedback (encoder, resolver, hall, etc.) problems, or other basic motor-control problems. When Manual Feedback is selected, the "measured" feedback angle is automatically calculated based on the user-supplied RPM (**MaVL.RPMUserCommand**) value. Manual Feedback is often used unloaded (without significant mass connected to the motor) because, in this mode, there is no physical feedback to determine the necessary phase-voltage changes to maintain motor/torque control.

Manual Feedback essentially provides an AC voltage to the motor phases with the frequency determined by RPMUserCommand. To use Manual Feedback, the following procedure can be used:



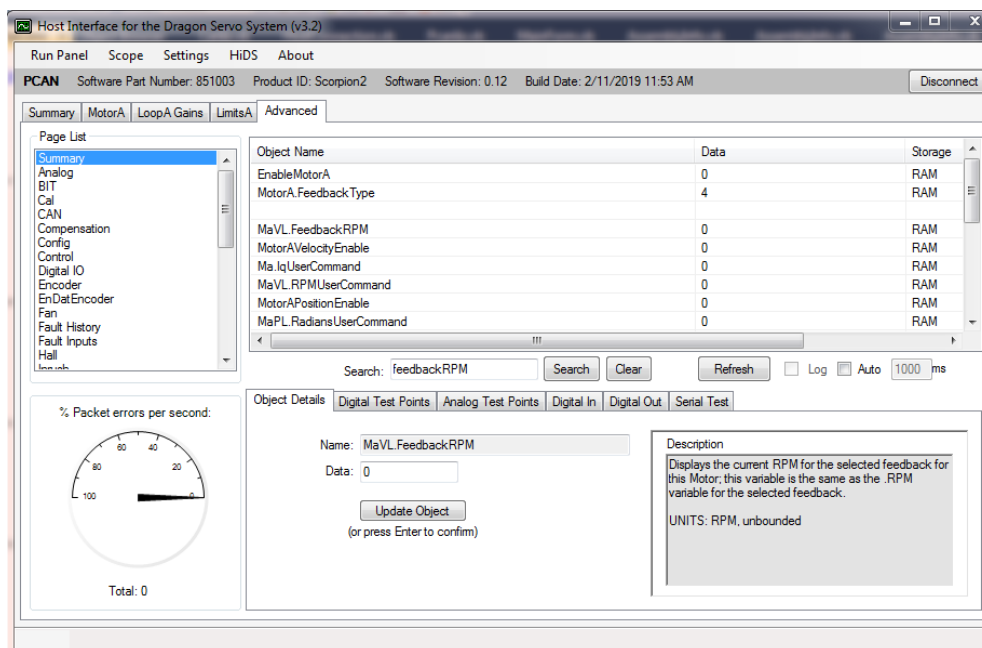
This Document does not contain Technical Data or Technology as defined in the ITAR Part 120.10 or EAR Part 772.

1. In the HiDS MotorA tab, select the Manual feedback method in the Feedback dropdown list.
2. In HiDS Loop Gains tab, select Torque method in the Control Mode drop-down list.
3. In the HiDS Run Panel, set variable RPMUserCommand to the desired final RPM. This RPM must be “easy” to achieve for the motor and is often less than 20% of the rated motor speed.
4. In the HiDS Run Panel, set variable Ma.AccelRPM to the desired acceleration. This acceleration must also be “easy” to achieve for the motor. A typical value is 25% of the RPMUserCommand.
5. In the HiDS Run Panel, set the IqUserCommand to 25% of the rated motor current. This value must be large enough to force commutation of the motor, essentially without feedback.
6. In the HiDS Run Panel, click [Run]. Note there may be some small movement in the motor as the DC-current may align the rotor to stator magnets. If the motor fails to spin, click [Stop] on the Run Panel and try increasing the IqUserCommand to as much as 50% of the rated motor current.

10 Appendix A: HiDS Variable Glossary

All HiDS variables are now include in the HiDS installation. The variable descriptions can be viewed three ways:

1. Within HiDS by clicking on the variable name (within the Advanced tab):



2. Within HiDS, select Settings->Export Variable Descriptions, and a txt file will be exported to the shown location.
3. For convenience, all common-controller variables are also listed here.

The data-types and bounds shown below should be considered when setting and reading a HiDS variable; however, all Controller variables are exchanged with HiDS as single-precision floating-point numbers, so care should be given to insure variables are configured within their defined bounds or only using the acceptable values.

All MotorA variables descriptions below apply to Motor B and Motor C variables for multiple-axis configurations.

10.1 Summary

Description: All variables shown on this page are copies of variables from other pages. This page only congregates commonly viewed variables. The descriptions of each of these are shown below.

Variable name: MaVL.FeedbackRPM
Units: RPM, unbounded
HiDS Location: Summary page
Description: Displays the current RPM for the selected feedback for this Motor; this variable is the same as the RPM variable for the selected feedback.

10.2 Analog

Variable name: Vbus
Units: Volts
HiDS Location: Analog page
Description: High/Motor voltage measurement in Volts. Depending on the hardware configuration, this measurement may only be available while the controller is enabled (measured inside the bus-switch).

Variable name: VbusPreInrush
Units: Volts
HiDS Location: Analog page
Description: High/Motor voltage measurement in Volts, measured prior to the bus switch; not available on all controllers.

Variable name: D5VMon
Units: Volts
HiDS Location: Analog page
Description: Internal DC reference voltage, expected to be close to +5V.

Variable name: DNeg5VMon
Units: Volts
HiDS Location: Analog page
Description: Internal DC reference voltage, expected to be close to -5V.

Variable name: Vref1
Units: Volts
HiDS Location: Analog page
Description: Internal DC reference voltage, hardware-dependent; typically +1 or +2V.

Variable name: Vref2
Units: Volts
HiDS Location: Analog page



This Document does not contain Technical Data or Technology as defined in the ITAR Part 120.10 or EAR Part 772.

Description:	Internal DC reference voltage, hardware-dependent; typically +1 or +2V.
Variable name:	DSPTemp
Units:	Degrees Celsius
HiDS Location:	Analog page
Description:	Measured temperature of the internal DSP
Variable name:	BoardId
Units:	Volts
HiDS Location:	Analog page
Description:	DSP configuration identifier
Variable name:	Mx.IcmdRaw
Units:	Volts
HiDS Location:	Analog page
Description:	External analog input command measured in volts. This is the raw voltage measured; pre scaling and pre deadband.
Variable name:	Mx.IcmdDeadband
Units:	Volts
HiDS Location:	Analog page
Description:	If the external analog input is used as a control-loop command, an analog input of 0 volts may measure a few millivolts. This deadband allows the loop to ignore near-zero values.
Variable name:	Mx.Icmd
Units:	Volts
HiDS Location:	Analog page
Description:	External analog input command measured in volts (post deadband-ignore). Once scaled, it can be selected as an input source to the Torque loop (refer to Mx.IcmdInput)
Variable name:	Mx.Brakel
Units:	Amps
HiDS Location:	Analog page
Description:	The measured current to the external motor-brake (if available in hardware).
Variable name:	Mx.IFan
Units:	Amps
HiDS Location:	Analog page
Description:	The measured current to the external Controller-cooling fan.
Variable name:	Mx.IBus
Units:	Amps
HiDS Location:	Analog page
Description:	The measured DC current from the high/motor DC-voltage input.
Variable name:	Mx.IGBTTemp
Units:	Degrees Celsius
HiDS Location:	Analog page
Description:	Measured temperature at the IGBT or MOSFET.

Variable name: Ma.ResolverASinAvg
Units: Volts
HiDS Location: Analog page
Description: Averaged raw resolver-A SIN input

Variable name: Ma.ResolverACosAvg
Units: Volts
HiDS Location: Analog page
Description: Averaged raw resolver-A COS input

Variable name: Ma.ResolverASinMon
Units: Volts
HiDS Location: Analog page
Description: Scaled resolver-A SIN input

Variable name: Ma.ResolverACosMon
Units: Volts
HiDS Location: Analog page
Description: Scaled resolver-A COS input

Variable name: Mb.ResolverBSinAvg
Units: Volts
HiDS Location: Analog page
Description: Averaged raw resolver-B SIN input

Variable name: Mb.ResolverBCosAvg
Units: Volts
HiDS Location: Analog page
Description: Averaged raw resolver-B COS input

Variable name: Mb.ResolverBSinMon
Units: Volts
HiDS Location: Analog page
Description: Scaled resolver-B SIN input

Variable name: Mb.ResolverBCosMon
Units: Volts
HiDS Location: Analog page
Description: Scaled resolver-B COS input

Variable name: Ma.ResolverExeMon
Units: Volts
HiDS Location: Analog page
Description: Raw measured resolver excitation

Variable name: MaxCommandVoltage
Units: Volts L-N
HiDS Location: Analog page

Description: This is the dynamically calculated maximum voltage, in Volts, that is available to commutate the motor. Once Mx.VoltageMagnitude exceeds the MaxCommandVoltage, the current-loop is referred to as saturated and is no longer able to produce the desired current, velocity, or position.

10.3 BIT (Built In Test)

The BIT variables are shown in groups as follows:

- <measurement name> BIT.enable indicates whether the BIT is enabled(1) or disabled(0).
- <measurement name> BIT.limit indicates the failure threshold, which is set to the units of the measurement being tested; read/write.
- <measurement name> BIT.current_count indicates the number of failed tests; note this count is cumulative, not consecutive; read only.
- <measurement name> BIT.max_count indicates the maximum number of failed tests. This variable is not used for a pass or fail determination, but can be used to indicate the BIT may be close to failure; read only.
- <measurement name> BIT.trip_count indicates the number of failed counts to show fault; note this count is cumulative, not consecutive. Unless otherwise indicated, each test occurs every 1 millisecond; read only.
- <measurement name> BIT.WarningEnable indicates whether the BIT warning check is enabled(1) or disabled(0). Warnings have no effect on run-time, and are typically reported via CAN or RS422 to the upper-layer control system.
- <measurement name> BIT.WarningLimit indicates the warning threshold, which is set to the units of the measurement being tested; read/write.
- <measurement name> BIT.WarningCount indicates the number of warning-exceeded tests; note This count is cumulative, not consecutive; read only.

The measurement limits can be configured in the [Limits] tab, or they can be changed by adjusting the <measurement name> BIT.limit variable located on the BIT page in the Advanced Tab:

- | | |
|----------------|--|
| Variable name: | VbusOvervoltageBIT.Enable |
| Description: | Boolean, 0=Disabled, 1=Enabled |
| HiDS Location: | BIT page |
| Description: | Enables monitoring of the motor bus voltage /motor power supply voltage for an over voltage condition; this BIT is run regardless of whether the motor is enabled. This fault should not be disabled as it can protect the Controller from damage. |
| | |
| Variable name: | VbusOvervoltageBIT.limit |
| Units: | Volts |
| HiDS Location: | BIT page |
| Description: | Indicates the failure threshold, which is set to the units of the measurement being tested. This limit should not be configured over the maximum rating of the controller. |
| | |
| Variable name: | VbusOvervoltageBIT.current_count |
| Units: | Integer, positive only |
| HiDS Location: | BIT page |
| Description: | Indicates the number of failed tests; note this count is cumulative, not consecutive. |

Variable name:	VbusOvervoltageBIT.max_count
Units:	Integer, positive only
HiDS Location:	BIT page
Description:	Indicates the maximum number of failed tests. This variable is not used for a pass or fail determination, but can be used to indicate the BIT may be close to failure.
Variable name:	VbusOvervoltageBIT.trip_count
Units:	Integer, positive only
HiDS Location:	BIT page
Description:	Indicates the number of failed counts to show fault; note this count is cumulative, not consecutive. Unless otherwise indicated, each test occurs every 1 millisecond.
Variable name:	VbusOvervoltageBIT.WarningEnable
Description:	Boolean, 0=Disabled, 1=Enabled
HiDS Location:	BIT page
Description:	Indicates whether the BIT warning check is enabled(1) or disabled(0). Warnings have no effect on run-time, and are typically reported via CAN or RS422 to the upper-layer control system.
Variable name:	VbusOvervoltageBIT.WarningLimit
Units:	Volts
HiDS Location:	BIT page
Description:	Indicates the warning threshold, which is set to the units of the measurement being tested.
Variable name:	VbusOvervoltageBIT.WarningCount
Units:	Integer, positive only
HiDS Location:	BIT page
Description:	Indicates the number of warning-exceeded tests; note this count is cumulative, not consecutive.
Variable name:	VbusUndervoltageBIT.Enable
Description:	Boolean, 0=Disabled, 1=Enabled
HiDS Location:	BIT page
Description:	Enables monitoring of motor bus voltage /motor power supply voltage for an under voltage condition; this BIT is only run when the motor is enabled.
Variable name:	VbusUndervoltageBIT.limit
Units:	Volts
HiDS Location:	BIT page
Description:	Indicates the failure threshold, which is set to the units of the measurement being tested
Variable name:	VbusUndervoltageBIT.current_count
Units:	Integer, positive only
HiDS Location:	BIT page
Description:	Indicates the number of failed tests; note this count is cumulative, not consecutive.
Variable name:	VbusUndervoltageBIT.max_count
Units:	Integer, positive only
HiDS Location:	BIT page
Description:	Indicates the maximum number of failed tests. This variable is not used for a pass or fail determination, but can be used to indicate the BIT may be close to failure.

Variable name:	VbusUndervoltageBIT.trip_count
Units:	Integer, positive only
HiDS Location:	BIT page
Description:	Indicates the number of failed counts to show fault; note this count is cumulative, not consecutive. Unless otherwise indicated, each test occurs every 1 millisecond.
Variable name:	VbusUndervoltageBIT.WarningEnable
Description:	Boolean, 0=Disabled, 1=Enabled
HiDS Location:	BIT page
Description:	Indicates whether the BIT warning check is enabled(1) or disabled(0). Warnings have no effect on run-time, and are typically reported via CAN or RS422 to the upper-layer control system.
Variable name:	VbusUndervoltageBIT.WarningLimit
Units:	Volts
HiDS Location:	BIT page
Description:	Indicates the warning threshold, which is set to the units of the measurement being tested.
Variable name:	VbusUndervoltageBIT.WarningCount
Units:	Integer, positive only
HiDS Location:	BIT page
Description:	Indicates the number of warning-exceeded tests; note this count is cumulative, not consecutive.
Variable name:	Precharge.Enable
Description:	Boolean, 0=Disabled, 1=Enabled
HiDS Location:	BIT page
Description:	Enables monitoring of inrush circuit. If enabled indicates that during motor startup, the measured Vbus was less than the Inrush.VbusOnThreshold threshold. This failure often indicates a disconnected or zero-voltage VBus (motor power supply).
Variable name:	Precharge.PrechargeState
Description:	Boolean, 0=OFF, 1=PRECHARGING
HiDS Location:	BIT page
Description:	Conveys the state of the precharge logic. If precharge is currently in process the value will be (1) otherwise (0); precharge should happen immediately after enabling the bus switch (servo enable).
Variable name:	Precharge.BusOnState
Description:	Boolean, 0=OFF, 1=ON
HiDS Location:	BIT page
Description:	Conveys the state of the bus switch. If the bus switch was turned on with no motor faults, this will be 1.
Variable name:	Precharge.MinCount
Units:	Integer, positive only
HiDS Location:	BIT page
Description:	Indicates the minimum charge time before checking if the VBUS is within expected threshold (refer to Vbus and Inrush.VbusOnThreshold); where each count is ~10ms.
Variable name:	Precharge.VbusOnThreshold

Units:	Volts
HiDS Location:	BIT page
Description:	Defines the minimum Vbus voltage that is acceptable after Precharge.MinCount time has elapsed; else a fault will be issued.
Variable name:	Precharge.WarningEnable
Description:	Boolean, 0=Disabled, 1=Enabled
HiDS Location:	BIT page
Description:	Indicates whether the BIT warning check is enabled(1) or disabled(0). Warnings have no effect on run-time, and are typically reported via CAN or RS422 to the upper-layer control system.
Variable name:	Precharge.VbusOnWarning
Description:	Boolean, 0=No Warning, 1=Warning Flagged
HiDS Location:	BIT page
Description:	Defines the minimum Vbus voltage that is acceptable after Precharge.MinCount time has elapsed; else a warning will be issued.
Variable name:	DSPTemperatureBIT.Enable
Description:	Boolean, 0=Disabled, 1=Enabled
HiDS Location:	BIT page
Description:	Enables temperature monitoring of the internal DSP (Digital Signal Processor) internal thermistor, in degrees C
Variable name:	DSPTemperatureBIT.limit
Units:	Degrees C
HiDS Location:	BIT page
Description:	Indicates the failure threshold, which is set to the units of the measurement being tested
Variable name:	DSPTemperatureBIT.current_count
Units:	Integer, positive only
HiDS Location:	BIT page
Description:	Indicates the number of failed tests; note this count is cumulative, not consecutive.
Variable name:	DSPTemperatureBIT.max_count
Units:	Integer, positive only
HiDS Location:	BIT page
Description:	Indicates the maximum number of failed tests. This variable is not used for a pass or fail determination, but can be used to indicate the BIT may be close to failure.
Variable name:	DSPTemperatureBIT.trip_count
Units:	Integer, positive only
HiDS Location:	BIT page
Description:	Indicates the number of failed counts to show fault; note this count is cumulative, not consecutive. Unless otherwise indicated, each test occurs every 1 millisecond.
Variable name:	DSPTemperatureBIT.WarningEnable
Description:	Boolean, 0=Disabled, 1=Enabled
HiDS Location:	BIT page
Description:	Indicates whether the BIT warning check is enabled(1) or disabled(0). Warnings have no effect on

run-time, and are typically reported via CAN or RS422 to the upper-layer control system.

Variable name: DSPTemperatureBIT.WarningLimit
Units: Degrees C
HiDS Location: BIT page
Description: Indicates the warning threshold, which is set to the units of the measurement being tested.

Variable name: DSPTemperatureBIT.WarningCount
Units: Integer, positive only
HiDS Location: BIT page
Description: Indicates the number of warning-exceeded tests; note this count is cumulative, not consecutive.

Variable name: VLogicUndervoltageBIT.Enable
Description: Boolean, 0=Disabled, 1=Enabled
HiDS Location: BIT page
Description: Enables the monitoring of logic power (in volts) for an under voltage condition.

Variable name: VLogicUndervoltageBIT.limit
Units: Volts
HiDS Location: BIT page
Description: Indicates the failure threshold, which is set to the units of the measurement being tested.

Variable name: VLogicUndervoltageBIT.current_count
Units: Integer, positive only
HiDS Location: BIT page
Description: Indicates the number of failed tests; note this count is cumulative, not consecutive.

Variable name: VLogicUndervoltageBIT.max_count
Units: Integer, positive only
HiDS Location: BIT page
Description: Indicates the maximum number of failed tests. This variable is not used for a pass or fail determination, but can be used to indicate the BIT may be close to failure.

Variable name: VLogicUndervoltageBIT.trip_count
Units: Integer, positive only
HiDS Location: BIT page
Description: Indicates the number of failed counts to show fault; note this count is cumulative, not consecutive. Unless otherwise indicated, each test occurs every 1 millisecond.

Variable name: VLogicUndervoltageBIT.WarningEnable
Description: Boolean, 0=Disabled, 1=Enabled
HiDS Location: BIT page
Description: Indicates whether the BIT warning check is enabled(1) or disabled(0). Warnings have no effect on run-time, and are typically reported via CAN or RS422 to the upper-layer control system.

Variable name: VLogicUndervoltageBIT.WarningLimit
Units: Volts
HiDS Location: BIT page
Description: Indicates the warning threshold, which is set to the units of the measurement being tested.

Variable name:	VLogicUndervoltageBIT.WarningCount
Units:	Integer, positive only
HiDS Location:	BIT page
Description:	Indicates the number of warning-exceeded tests; note this count is cumulative, not consecutive.
Variable name:	AccessoryLOBIT.WarningEnable
Description:	Boolean, 0=Disabled, 1=Enabled
HiDS Location:	BIT page
Description:	Enables the monitoring of the internal Fan, Inrush, Regen, and external-5V switches.
Variable name:	AccessoryLOBIT.WarningCount
Units:	Integer, positive only
HiDS Location:	BIT page
Description:	Indicates the number of warning-exceeded tests; note this count is cumulative, not consecutive.
Variable name:	AccessoryLOBIT.trip_count
Units:	Integer, positive only
HiDS Location:	BIT page
Description:	Indicates the number of failed counts to show the warning; note this count is cumulative, not consecutive. Unless otherwise indicated, each test occurs every 1 millisecond.
Variable name:	Cpu2ExecutionBIT.Enable
Description:	Boolean, 0=Disabled, 1=Enabled
HiDS Location:	BIT page
Description:	If enabled this will monitor whether or not the second core is running.
Variable name:	Cpu2ExecutionBIT.WarningEnable
Description:	Boolean, 0=Disabled, 1=Enabled
HiDS Location:	BIT page
Description:	Indicates whether the BIT warning check is enabled(1) or disabled(0). Warnings have no effect on run-time, and are typically reported via CAN or RS422 to the upper-layer control system.
Variable name:	MxOverspeedPos.Enable
Description:	Boolean, 0=Disabled, 1=Enabled
HiDS Location:	BIT page
Description:	If enabled, verifies that the absolute value of the measured RPM does not exceed the specified limit
Variable name:	MxOverspeedPos.limit
Units:	RPM
HiDS Location:	BIT page
Description:	Indicates the failure threshold, which is set to the units of the measurement being tested
Variable name:	MxOverspeedPos.current_count
Units:	Integer, positive only
HiDS Location:	BIT page
Description:	Indicates the number of failed tests; note this count is cumulative, not consecutive.

Variable name:	MxOverspeedPos.max_count
Units:	Integer, positive only
HiDS Location:	BIT page
Description:	Indicates the maximum number of failed tests. This variable is not used for a pass or fail determination, but can be used to indicate the BIT may be close to failure.
Variable name:	MxOverspeedPos.trip_count
Units:	Integer, positive only
HiDS Location:	BIT page
Description:	Indicates the number of failed counts to show fault; note this count is cumulative, not consecutive. Unless otherwise indicated, each test occurs every 1 millisecond.
Variable name:	MxOverspeedPos.WarningEnable
Description:	Boolean, 0=Disabled, 1=Enabled
HiDS Location:	BIT page
Description:	Indicates whether the BIT warning check is enabled(1) or disabled(0). Warnings have no effect on run-time, and are typically reported via CAN or RS422 to the upper-layer control system.
Variable name:	MxOverspeedPos.WarningLimit
Units:	RPM
HiDS Location:	BIT page
Description:	Indicates the warning threshold, which is set to the units of the measurement being tested.
Variable name:	MxOverspeedPos.WarningCount
Units:	Integer, positive only
HiDS Location:	BIT page
Description:	Indicates the number of warning-exceeded tests; note this count is cumulative, not consecutive.
Variable name:	MxVL.DisplayedFilteredRPM
Units:	RPM, unbounded
HiDS Location:	BIT page
Description:	Motor-X filtered RPM, calculated using selected feedback
Variable name:	MxMotorTempBit.Enable
Description:	Boolean, 0=Disabled, 1=Enabled
HiDS Location:	BIT page
Description:	If enabled indicates if the Motor temperature exceeded the specified limit
Variable name:	MxMotorTempBit.limit
Units:	Degrees C
HiDS Location:	BIT page
Description:	Indicates the failure threshold, which is set to the units of the measurement being tested
Variable name:	MxMotorTempBit.current_count
Units:	Integer, positive only
HiDS Location:	BIT page
Description:	Indicates the number of failed tests; note this count is cumulative, not consecutive.
Variable name:	MxMotorTempBit.max_count

Units:	Integer, positive only
HiDS Location:	BIT page
Description:	Indicates the maximum number of failed tests. This variable is not used for a pass or fail determination, but can be used to indicate the BIT may be close to failure.
Variable name:	MxMotorTempBit.trip_count
Units:	Integer, positive only
HiDS Location:	BIT page
Description:	Indicates the number of failed counts to show fault; note this count is cumulative, not consecutive. Unless otherwise indicated, each test occurs every 1 millisecond.
Variable name:	MxMotorTempBit.WarningEnable
Description:	Boolean, 0=Disabled, 1=Enabled
HiDS Location:	BIT page
Description:	Indicates whether the BIT warning check is enabled(1) or disabled(0). Warnings have no effect on run-time, and are typically reported via CAN or RS422 to the upper-layer control system.
Variable name:	MxMotorTempBit.WarningLimit
Units:	Degrees C
HiDS Location:	BIT page
Description:	Indicates the warning threshold, which is set to the units of the measurement being tested.
Variable name:	MxMotorTempBit.WarningCount
Units:	Integer, positive only
HiDS Location:	BIT page
Description:	Indicates the number of warning-exceeded tests; note this count is cumulative, not consecutive.
Variable name:	MxIGBTTemperatureBIT.Enable
Description:	Boolean, 0=Disabled, 1=Enabled
HiDS Location:	BIT page
Description:	If enabled, notifies user that the temperature measured at the Servo IGBT (or MOSFET) has exceeded the specified limit
Variable name:	MxIGBTTemperatureBIT.limit
Units:	Degrees C
HiDS Location:	BIT page
Description:	Indicates the failure threshold, which is set to the units of the measurement being tested
Variable name:	MxIGBTTemperatureBIT.current_count
Units:	Integer, positive only
HiDS Location:	BIT page
Description:	Indicates the number of failed tests; note this count is cumulative, not consecutive.
Variable name:	MxIGBTTemperatureBIT.max_count
Units:	Integer, positive only
HiDS Location:	BIT page
Description:	Indicates the maximum number of failed tests. This variable is not used for a pass or fail determination, but can be used to indicate the BIT may be close to failure.

Variable name:	MxIGBTTemperatureBIT.trip_count
Units:	Integer, positive only
HiDS Location:	BIT page
Description:	Indicates the number of failed counts to show fault; note this count is cumulative, not consecutive. Unless otherwise indicated, each test occurs every 1 millisecond.
Variable name:	MxIGBTTemperatureBIT.WarningEnable
Description:	Boolean, 0=Disabled, 1=Enabled
HiDS Location:	BIT page
Description:	Indicates whether the BIT warning check is enabled(1) or disabled(0). Warnings have no effect on run-time, and are typically reported via CAN or RS422 to the upper-layer control system.
Variable name:	MxIGBTTemperatureBIT.WarningLimit
Units:	Degrees C
HiDS Location:	BIT page
Description:	Indicates the warning threshold, which is set to the units of the measurement being tested.
Variable name:	MxIGBTTemperatureBIT.WarningCount
Units:	Integer, positive only
HiDS Location:	BIT page
Description:	Indicates the number of warning-exceeded tests; note this count is cumulative, not consecutive.
Variable name:	MxLossOfFeedback.Enable
Description:	Boolean, 0=Disabled, 1=Enabled
HiDS Location:	BIT page
Description:	If enabled, the software will verify that the selected feedback is connected; if disconnected, motor will be disabled
Variable name:	MxLossOfFeedback.limit
Units:	Depends on feedback
HiDS Location:	BIT page
Description:	Indicates the failure threshold, which is set to the units of the measurement being tested
Variable name:	MxLossOfFeedback.current_count
Units:	Integer, positive only
HiDS Location:	BIT page
Description:	Indicates the number of failed tests; note this count is cumulative, not consecutive.
Variable name:	MxLossOfFeedback.max_count
Units:	Integer, positive only
HiDS Location:	BIT page
Description:	Indicates the maximum number of failed tests. This variable is not used for a pass or fail determination, but can be used to indicate the BIT may be close to failure.
Variable name:	MxLossOfFeedback.trip_count
Units:	Integer, positive only
HiDS Location:	BIT page
Description:	Indicates the number of failed counts to show fault; note this count is cumulative, not consecutive. Unless otherwise indicated, each test occurs every 1 millisecond.

Variable name:	MxLossOfFeedback.WarningEnable
Description:	Boolean, 0=Disabled, 1=Enabled
HiDS Location:	BIT page
Description:	Indicates whether the BIT warning check is enabled(1) or disabled(0). Warnings have no effect on run-time, and are typically reported via CAN or RS422 to the upper-layer control system.
Variable name:	MxLossOfFeedback.WarningLimit
Units:	Depends on feedback
HiDS Location:	BIT page
Description:	Indicates the warning threshold, which is set to the units of the measurement being tested.
Variable name:	MxLossOfFeedback.WarningCount
Units:	Integer, positive only
HiDS Location:	BIT page
Description:	Indicates the number of warning-exceeded tests; note this count is cumulative, not consecutive.
Variable name:	AutoSetI2TParameters
Description:	Boolean, 0=Disabled, 1=Enabled
HiDS Location:	BIT page
Description:	Automatically sets the I2T values below based on the MaxCurrentCommand setting for this motor.
Variable name:	I2Tx.Enable
Description:	Boolean, 0=Disabled, 1=Enabled
HiDS Location:	BIT page
Description:	I2T (read I-squared-T) is an estimation of the energy content in current transient conditions. Because a motor can often handle short current transients in excess of rated currents, this test can help protect against motor overheating or damage due to excess current transients.
Variable name:	I2Tx.ContinuousRmsAmps
Units:	Amps-RMS
HiDS Location:	BIT page
Description:	Sets the continuous current rating of the motor.
Variable name:	I2Tx.OverloadRmsAmps
Units:	Amps-RMS
HiDS Location:	BIT page
Description:	Sets the peak/overload current rating of the motor; this parameter is associated with the I2Tx.OverloadSeconds.
Variable name:	I2Tx.OverloadSeconds
Units:	Time in seconds
HiDS Location:	BIT page
Description:	Sets the time associated with the peak/overload current rating of the motor.
Variable name:	I2Tx.Threshold
Units:	Amps-Amps-Seconds
HiDS Location:	BIT page

Description:	Calculated as follows: $(\text{OverloadRmsAmps2} - \text{ContinuousRmsAmps2}) * \text{OverloadSeconds}$
Variable name:	I2Tx.Measurement
Units:	Amps-Amps-Seconds
HiDS Location:	BIT page
Description:	Each measurement occurs every 0.1 seconds. This value = $(0.5 * \text{IqSlowFiltered2} - \text{ContinuousRmsAmps2}) * \text{ContinuousRmsAmps} * 0.1\text{seconds}$. Note the first 0.5 term is because IqSlowFiltered is Amps peak.
Variable name:	I2Tx.Sum
Units:	Amps-Amps-Seconds
HiDS Location:	BIT page
Description:	Is the I2Tx. Measurement summed every cycle as long as the motor is enabled. If this value exceeds the Threshold, then a fault occurs.
Variable name:	Mx.IqSlowFiltered
Units:	Amps
HiDS Location:	BIT page
Description:	The output of a low pass first order filter with the Mx.Iq vector as input
Variable name:	MxVelocityTrackingBIT.Enable
Description:	Boolean, 0=Disabled, 1=Enabled
HiDS Location:	BIT page
Description:	If enabled, verifies that the RPM-error (commanded - actual) does not exceed the specified limit
Variable name:	MxVelocityTrackingBIT.limit
Units:	RPM
HiDS Location:	BIT page
Description:	Indicates the failure threshold, which is set to the units of the measurement being tested
Variable name:	MxVelocityTrackingBIT.current_count
Units:	Integer, positive only
HiDS Location:	BIT page
Description:	Indicates the number of failed tests; note this count is cumulative, not consecutive.
Variable name:	MxVelocityTrackingBIT.max_count
Units:	Integer, positive only
HiDS Location:	BIT page
Description:	Indicates the maximum number of failed tests. This variable is not used for a pass or fail determination, but can be used to indicate the BIT may be close to failure.
Variable name:	MxVelocityTrackingBIT.trip_count
Units:	Integer, positive only
HiDS Location:	BIT page
Description:	Indicates the number of failed counts to show fault; note this count is cumulative, not consecutive. Unless otherwise indicated, each test occurs every 1 millisecond.
Variable name:	MxVelocityTrackingBIT.WarningEnable
Description:	Boolean, 0=Disabled, 1=Enabled

HiDS Location:	BIT page
Description:	Indicates whether the BIT warning check is enabled(1) or disabled(0). Warnings have no effect on run-time, and are typically reported via CAN or RS422 to the upper-layer control system.
Variable name:	MxVelocityTrackingBIT.WarningLimit
Units:	RPM
HiDS Location:	BIT page
Description:	Indicates the warning threshold, which is set to the units of the measurement being tested.
Variable name:	MxVelocityTrackingBIT.WarningCount
Units:	Integer, positive only
HiDS Location:	BIT page
Description:	Indicates the number of warning-exceeded tests; note this count is cumulative, not consecutive.
Variable name:	MxPositionBIT.Enable
Description:	Boolean, 0=Disabled, 1=Enabled
HiDS Location:	BIT page
Description:	If enabled, in position mode verifies that measured Motor position is within the set limits
Variable name:	MxPositionBIT.limit
Units:	Radians
HiDS Location:	BIT page
Description:	Indicates the failure threshold, which is set to the units of the measurement being tested
Variable name:	MxPositionBIT.current_count
Units:	Integer, positive only
HiDS Location:	BIT page
Description:	Indicates the number of failed tests; note this count is cumulative, not consecutive.
Variable name:	MxPositionBIT.max_count
Units:	Integer, positive only
HiDS Location:	BIT page
Description:	Indicates the maximum number of failed tests. This variable is not used for a pass or fail determination, but can be used to indicate the BIT may be close to failure.
Variable name:	MxPositionBIT.trip_count
Units:	Integer, positive only
HiDS Location:	BIT page
Description:	Indicates the number of failed counts to show fault; note this count is cumulative, not consecutive. Unless otherwise indicated, each test occurs every 1 millisecond.
Variable name:	MxPositionBIT.WarningEnable
Description:	Boolean, 0=Disabled, 1=Enabled
HiDS Location:	BIT page
Description:	Indicates whether the BIT warning check is enabled(1) or disabled(0). Warnings have no effect on run-time, and are typically reported via CAN or RS422 to the upper-layer control system.
Variable name:	MxPositionBIT.WarningLimit
Units:	Radians

HiDS Location:	BIT page
Description:	Indicates the warning threshold, which is set to the units of the measurement being tested.
Variable name:	MxPositionBIT.WarningCount
Units:	Integer, positive only
HiDS Location:	BIT page
Description:	Indicates the number of warning-exceeded tests; note this count is cumulative, not consecutive.

10.4 CAL

Variable name:	Various
Summary:	The current-calibration values, as well as any other customer or application specific calibrations.
HiDS location:	Shown on Cal page in Advanced tab.
Description:	Read Only. Besides ESI-internal debug variables, this page only contains customer-relevant variables should there be customer or application specific calibrations. Please refer to the customer-specific HiDS addendum for these variables.

10.5 CAN

Variable name:	CanAAddresses.ChannelNumber
Units:	Integer, 0-15
HiDS Location:	CAN page
Description:	For Controllers using CAN communication for HiDS, this is the channel number shown on the initial connection view when HiDS first starts. For an application that has 2 or more Controllers connected to the same PC, this channel number can be assigned uniquely to each Controller, which allows the PC to address each Controller differently. To change this value, one should insure the USB interface works first, because making a mistake here may require USB connectivity to correct. To take effect, change the channel number (to 1 for instance), select Settings->Save Objects to EEPROM, and reset the Controller
Variable name:	CanAAddresses.BitRate
Units:	Baud/bits per second, valid values are: 100000, 250000, 500000, and 1000000.
HiDS Location:	CAN page
Description:	For Controllers using CAN communication for HiDS or control, the Controller will ship with a default bit-rate (often 1Mbps). Should another value be desired, this variable can be changed to 500, 250, or 100 (Kbps). Note should an incorrect value be entered and saved here, the only way to recover would be to login via USB and fix the value.
Variable name:	CanAAddresses.IncommingHids
Units:	Integer, valid values are 1536 (0x600) to 1551 (0x60F) for incoming messages.
HiDS Location:	CAN page
Description:	For Controllers using CAN communication for HiDS, these are the CAN addresses of the HiDS connection. All ESI Motion Controllers ship using 1536 (0x600) for the incoming address. The actual addresses used are (1536 + CanAddresses.ChannelNumber) for incoming. If another CAN device is one a shared bus, and if an address conflict occurs, then CanAddresses.ChannelNumber can be changed to offset the HiDS addresses.
Variable name:	CanAAddresses.OutgoingHids

Description:	Integer, valid values are 1552 (0x610) to 1567 (0x61F) for outgoing messages.
HiDS Location:	CAN page
Description:	For Controllers using CAN communication for HiDS, these are the CAN addresses of the HiDS connection. All ESI Motion Controllers ship using 1552 (0x610) for the outgoing address. The actual addresses used are (1552 + CanAddresses.ChannelNumber) for outgoing. If another CAN device is on a shared bus, and if an address conflict occurs, then CanAddresses.ChannelNumber can be changed to offset the HiDS addresses.
Variable name:	CanControlEnable
Units:	Boolean, 0 or 1
HiDS Location:	CAN page
Description:	For Controllers using CAN (non-HiDS) communication for control, this variable can enable or disable that control interface. Note in order to use HiDS for motor control, it is likely that the external CAN control interface must be disabled.
Variable name:	CANControlMsgId
Units:	Integer, 1-2047
HiDS Location:	CAN page
Description:	For Controllers using CAN (non-HiDS) communication for control, this variable sets the address / message ID of the incoming CAN command packet; note this is a decimal (not hexadecimal) value.
Variable name:	CANCommandRate_ms
Units:	Integer, 0 to unbounded
HiDS Location:	CAN page
Description:	For Controllers using CAN (non-HiDS) communication for control, this variable sets CAN- (received)command timeout to report an error. A value of 0 disables the timeout check. If non-zero, and if CAN-packets are not received in this many milliseconds, a fault occurs and motor-control stops.
Variable name:	CANFeedbackEnable
Units:	Boolean, 0 or 1
HiDS Location:	CAN page
Description:	For Controllers using CAN (non-HiDS) communication for status, this variable enables or disables that message. Setting this variable to 0 disables the message and 1 enables it.
Variable name:	CANFeedbackMsgId
Units:	Integer, 1-2047
HiDS Location:	CAN page
Description:	For Controllers using CAN (non-HiDS) communication for status, this variable sets the address / message ID of the outgoing CAN feedback / command packet; note this is a decimal (not hexadecimal) value.
Variable name:	CANFeedbackRate_ms
Units:	Time in milliseconds
HiDS Location:	CAN page
Description:	For Controllers using CAN (non-HiDS) communication for status, this variable sets the interval in milliseconds between CAN status packet transmissions.
Variable name:	CANPacketsReceived

Units:	Integer, unbounded
HiDS Location:	CAN page
Description:	Unless purposely cleared, keeps a counter for the number of successful CAN control packets received.
Variable name:	CanAErrorCount
Units:	Integer, unbounded
HiDS Location:	CAN page
Description:	Reads the error-counters from the DSP CAN peripheral.
Variable name:	CanATxErrors
Units:	Integer, unbounded
HiDS Location:	CAN page
Description:	From the DSP CAN peripheral error counter (CanAErrorCount), this is the transmit errors only.
Variable name:	CanARxErrors
Units:	Integer, unbounded
HiDS Location:	CAN page
Description:	From the DSP CAN peripheral error counter (CanAErrorCount), this is the receive errors only.

10.6 Compensation

Variable name:	Mx.Kp
Units:	Volts per amp, positive values only
HiDS Location:	Compensation page
Description:	Used to adjust the Proportional Gain component for reducing the error between the Iq command (Mx.IqCmd) and Iq actual (Mx.Iq).
Variable name:	Mx.Ki
Units:	Unitless, positive values only
HiDS Location:	Compensation page
Description:	Used to adjust the Integral Gain component for reducing the error between the Iq command (Mx.IqCmd) and Iq actual (Mx.Iq).
Variable name:	Mx.IqErrorIntegral
Units:	Volts, bounded by Mx.IntegralMin and Mx.IntegralMax
HiDS Location:	Compensation page
Description:	Displays the IQ integral value of the current loop for the Motor
Variable name:	Mx.IntegralMax
Units:	Volts, Mx.IntegralMax should be positive; Mx.IntegralMin should be negative. The bounds are application specific, but should be less than the motor-bus voltage available
HiDS Location:	Compensation page
Description:	Integral Gain compensation is a slower moving compensation, and under certain conditions is subject to error buildup. These settings limit the integral value to the limits specified. Note these are automatically set to the maximum output limits of the loop if variable Mx.AutoSetIntegralMinMax is set to true (1).

Variable name:	Mx.IntegralMin
Units:	Volts, Mx.IntegralMax should be positive; Mx.IntegralMin should be negative. The bounds are application specific, but should be less than the motor-bus voltage available
HiDS Location:	Compensation page
Description:	Integral Gain compensation is a slower moving compensation, and under certain conditions is subject to error buildup. These settings limit the integral value to the limits specified. Note these are automatically set to the maximum output limits of the loop if variable Mx.AutoSetIntegralMinMax is set to true (1).
Variable name:	Mx.AutoSetIntegralMinMax
Units:	Boolean, 0 or 1
HiDS Location:	Compensation page
Description:	If 1 (true), automatically sets the Mx.IntegralMax to the voltage set by the VbusOvervoltageBIT.limit variable, and sets the Mx.IntegralMin to the negative of the VbusOvervoltageBIT.limit variable.
Variable name:	Mx.IdKp
Units:	Volts per amp, positive values only
HiDS Location:	Compensation page
Description:	Used to adjust the Proportional Gain component for reducing the error between the ID command (Mx.IdCmd) and Iq actual (Mx.Id). Note IdCmd is typically 0.
Variable name:	Mx.IdKi
Units:	Unitless, positive values only
HiDS Location:	Compensation page
Description:	Used to adjust the Integral Gain component for reducing the error between the ID command (Mx.IdCmd) and ID actual (Mx.Id). Note IdCmd is typically 0.
Variable name:	Mx.IdErrorIntegral
Units:	Volts, bounded by Mx.IdIntegralMin and Mx.IdIntegralMax
HiDS Location:	Compensation page
Description:	Displays the ID integral value of the current-loop for Motor A.
Variable name:	Mx.IdIntegralMax
Units:	Volts, Mx.IdIntegralMax should be positive; Mx.IdIntegralMin should be negative. The bounds are application specific, but should be less than the motor-bus voltage available
HiDS Location:	Compensation page
Description:	Integral Gain compensation is a slower moving compensation, and under certain conditions is subject to error buildup. These settings limit the integral value to the limits specified. Note these are automatically set to the maximum output limits of the loop if variable Mx.IdAutoSetIntegralMinMax is set to true (1).
Variable name:	Mx.IdIntegralMin
Units:	Volts, Mx.IdIntegralMax should be positive; Mx.IdIntegralMin should be negative. The bounds are application specific, but should be less than the motor-bus voltage available
HiDS Location:	Compensation page
Description:	Integral Gain compensation is a slower moving compensation, and under certain conditions is subject to error buildup. These settings limit the integral value to the limits specified. Note these are automatically set to the maximum output limits of the loop if variable

Mx.IdAutoSetIntegralMinMax is set to true (1).

Variable name: Mx.IdAutoSetIntegralMinMax
Units: Boolean, 0 or 1
HiDS Location: Compensation page
Description: If 1 (true), automatically sets the Mx.IdIntegralMax to the voltage set by the VbusOvervoltageBIT.limit variable, and sets the Mx.IdIntegralMin to the negative of the VbusOvervoltageBIT.limit variable.

Variable name: Mx.IdAutoSetTolqGainValues
Units: Boolean, 0 or 1
HiDS Location: Compensation page
Description: If 1 (true), automatically sets the Mx.IdKp and Mx.IdKi to the Mx.Kp and Mx.Ki settings, which is the most common.

Variable name: Mx.VelocityKp
Units: Amps per RPM, positive values only
HiDS Location: Compensation page
Description: Used to adjust the Proportional Gain component for reducing the error between the acceleration-limited Velocity command (MxVL.AccelLimitedRPMCommand) and actual measured velocity (either ResolverA.Rpm, EncoderA.Rpm, Etc).

Variable name: Mx.VelocityKi
Units: Unitless, positive values only.
HiDS Location: Compensation page
Description: Used to adjust the Integral Gain component for reducing the error between the acceleration-limited Velocity command (MxVL.AccelLimitedRPMCommand) and actual measured velocity (either ResolverA.Rpm, EncoderA.Rpm, Etc).

Variable name: Mx.VelocityKd
Units: Amps, positive values only
HiDS Location: Compensation page
Description: Used to adjust the Derivative Gain component for reducing the error between the acceleration-limited Velocity command (MxVL.AccelLimitedRPMCommand) and actual measured velocity (either ResolverA.Rpm, EncoderA.Rpm, Etc). Note the derivative term is typically zero as derivative compensation can often lead to loop instability.

Variable name: Mx.VelocityErrorIntegral
Units: Amps, bounded by Mx.VelocityIntegralMin and Mx.VelocityIntegralMax.
HiDS Location: Compensation page
Description: The velocity loop current integral value.

Variable name: Mx.VelocityIntegralMax
Units: Amps, Mx.VelocityIntegralMax should be positive; Mx.VelocityIntegralMin should be negative. The bounds are application specific, but should be less than the over-current value (Mx.HardwareOvercurrent).
HiDS Location: Compensation page
Description: Integral Gain compensation is a slower moving compensation, and under certain conditions is subject to error buildup. These settings limit the integral value to the limits specified. Note these

are automatically set to the maximum output limits of the loop if variable Mx.VelocityAutoSetIntegralMinMax is set to true (1).

Variable name: Mx.VelocityIntegralMin
Units: Amps, Mx.VelocityIntegralMax should be positive; Mx.VelocityIntegralMin should be negative. The bounds are application specific, but should be less than the over-current value (Mx.HardwareOvercurrent).
HiDS Location: Compensation page
Description: Integral Gain compensation is a slower moving compensation, and under certain conditions is subject to error buildup. These settings limit the integral value to the limits specified. Note these are automatically set to the maximum output limits of the loop if variable Mx.VelocityAutoSetIntegralMinMax is set to true (1).

Variable name: Mx.VelocityAutoSetIntegralMinMax
Units: Boolean, 0 or 1
HiDS Location: Compensation page
Description: If 1 (true), automatically sets the Mx.VelocityIntegralMax to the amperage set by the Mx.HardwareOvercurrent variable, and sets the Mx.VelocityIntegralMin to the negative of the Mx.HardwareOvercurrent variable.

Variable name: MxSmo.VelocityKp
Units: Amps per RPM, positive values only
HiDS Location: Compensation page
Description: When sensorless feedback is selected, used to adjust the Proportional Gain component for reducing the error between the acceleration-limited Velocity command (MxVL.AccelLimitedRPMCommand) and actual measured velocity (SensorlessA.RPM).

Variable name: MxSmo.VelocityKi
Units: Unitless, positive values only.
HiDS Location: Compensation page
Description: When sensorless feedback is selected, used to adjust the Integral Gain component for reducing the error between the acceleration-limited Velocity command (MxVL.AccelLimitedRPMCommand) and actual measured velocity (SensorlessA.RPM).

Variable name: MxSmo.VelocityIntegralMax
Units: Amps, MxSmo.VelocityIntegralMax should be positive; MxSmo.VelocityIntegralMin should be negative. The bounds are application specific, but should be less than the over-current value (Mx.HardwareOvercurrent).
HiDS Location: Compensation page
Description: Integral Gain compensation is a slower moving compensation, and under certain conditions is subject to error buildup. These settings limit the integral value to the limits specified. Note these are automatically set to the maximum output limits of the loop if variable Mx.VelocityAutoSetIntegralMinMax is set to true (1).

Variable name: MxSmo.VelocityIntegralMin
Units: Amps, MxSmo.VelocityIntegralMax should be positive; MxSmo.VelocityIntegralMin should be negative. The bounds are application specific, but should be less than the over-current value (Mx.HardwareOvercurrent).
HiDS Location: Compensation page

Description:	Integral Gain compensation is a slower moving compensation, and under certain conditions is subject to error buildup. These settings limit the integral value to the limits specified. Note these are automatically set to the maximum output limits of the loop if variable Mx.VelocityAutoSetIntegralMinMax is set to true (1).
Variable name:	MxSmo.VelocityErrorIntegral
Units:	Amps, bounded by MxSmo.VelocityIntegralMin and MxSmo.VelocityIntegralMax.
HiDS Location:	Compensation page
Description:	The velocity loop current integral value when in sensorless mode
Variable name:	Mx.PositionKp
Units:	RPM per Radian, positive values only.
HiDS Location:	Compensation page
Description:	Used to adjust the Proportional Gain component for reducing the error between the acceleration-limited Position command (MxPL.AccelLimitedRadiansCommand) and actual measured Position (MxPL.AbsolutePosition).
Variable name:	Mx.PositionKi
Units:	Unitless, positive values only.
HiDS Location:	Compensation page
Description:	Used to adjust the Integral Gain component for reducing the error between the acceleration-limited Position command (MxPL.AccelLimitedRadiansCommand) and actual measured Position (MxPL.AbsolutePosition).
Variable name:	Mx.PositionKd
Units:	RPM, positive values only.
HiDS Location:	Compensation page
Description:	Used to adjust the Derivative Gain component for reducing the error between the acceleration-limited Position command (MxPL.AccelLimitedRadiansCommand) and actual measured Position (MxPL.AbsolutePosition). Note the derivative term is typically zero as derivative compensation can often lead to loop instability.
Variable name:	MaPL.RadiansErrorIntegral
Units:	Amps, bounded by Mx.PositionIntegralMin and Mx.PositionIntegralMax.
HiDS Location:	Compensation page
Description:	Displays the integral value of the position loop.
Variable name:	Mx.PositionIntegralMax
Units:	RPM, Mx.PositionIntegralMax should be positive; Mx.PositionIntegralMin should be negative. The bounds are application specific, but should be less than the over-speed value (MaOverspeedPos.limit).
HiDS Location:	Compensation page
Description:	Integral Gain compensation is a slower moving compensation, and under certain conditions is subject to error buildup. These settings limit the integral value to the limits specified. Note these are automatically set to the maximum output limits of the loop if variable Mx.PositionAutoSetIntegralMinMax is set to true (1).
Variable name:	Mx.PositionIntegralMin
Units:	RPM, Mx.PositionIntegralMax should be positive; Mx.PositionIntegralMin should be negative. The

bounds are application specific, but should be less than the over-speed value (MaOverspeedPos.limit).

HiDS Location: Compensation page
Description: Integral Gain compensation is a slower moving compensation, and under certain conditions is subject to error buildup. These settings limit the integral value to the limits specified. Note these are automatically set to the maximum output limits of the loop if variable Mx.PositionAutoSetIntegralMinMax is set to true (1).

Variable name: Mx.PositionAutoSetIntegralMinMax
Units: Boolean, 0 or 1
HiDS Location: Compensation page
Description: If 1 (true), automatically sets the Mx.PositionIntegralMax to the RPM set by the MaOverspeedPos.limit variable, and sets the Mx.PositionIntegralMin to the negative of the MaOverspeedPos.limit variable.

10.7 Config

Variable name: RevertSettingsToFactoryDefault
Units: Boolean, 0 or 1
HiDS Location: Config page
Description: Will reset the Controller and will reset all stored parameters to the original factory default. It is highly recommended that all Controller settings be exported (in HiDS, select Settings->Export all Displayed EEPROM...) prior to executing this, as all customer-configuration-changes will be lost.

Variable name: Cpu2SwPartNumber
Units: Integer 800000-899999
HiDS Location: Config page
Description: For dual-core processors only, this is the ESI software part number of the software running on CPU2.

Variable name: Cpu2SwRevision
Units: x.y format: 0.01 to 9.99
HiDS Location: Config page
Description: For dual-core processors only, this is the revision of the software running on CPU2.

10.8 Control

Variable name: EnableMotorX
Units: Boolean, 0 or 1
HiDS Location: Control page
Description: Enables the motor state machine to begin active PWM of the motor-phases. This is the same variable set to True by clicking [Run] in the Run Panel

Variable name: MotorXState
Units: Integer, 0-10
HiDS Location: Control page
Description: Indicates the current power state of the Motor

0 = Drive Disabled
 1 = Idle
 2 = Precharging
 4 = Assert clear faults
 6 = Verify motor faults are clear
 8 = Enabling Bridge
 9 = Enabling PWM
 10 = Running

Variable name: Mx.BrakeEngaged
 Units: Boolean, 0 or 1
 HiDS Location: Control page
 Description: Indicates where brake is engaged (1), or disengaged (0)

Variable name: EnableAutoBrakeControl
 Units: Boolean, 0 or 1
 HiDS Location: Control page
 Description: If EnableAutoBrakeControl = 1, the motor brakes are automatically engaged when motor-PWMing are stopped, and the brakes are automatically released mid-way into the motor-state machine (to reduce slippage). Setting EnableAutoBrakeControl = 0 allows manual manipulation of the brake variable Mx.BrakeEngaged.

Variable name: MotorBSlaveFromMotorAEnable
 Units: Boolean, 0 or 1
 HiDS Location: Control page
 Description: For dual-axis controllers, if both motors are connected mechanically (i.e. via a common-shaft), then the MotorA enable/disable actions can control MotorB. Setting this to 1 (true) will enable MotorB whenever MotorA is enabled.

Variable name: MotorBSlaveCurrentPolarity
 Units: Sign, the valid values are -1 and 1
 HiDS Location: Control page
 Description: For dual-axis controllers, if both motors are connected mechanically (i.e. via a common-shaft), this variable allows MotorB to spin in the same direction as MotorA (MotorBSlaveCurrentPolarity = 1), or in opposite directions (MotorBSlaveCurrentPolarity = -1).

Variable name: EnableMotorAll
 Units: Boolean, 0 or 1
 HiDS Location: Control page
 Description: For dual-axis controllers, these variables allow a single control to enable or disable both motors.

Variable name: DisableMotorAll
 Units: Boolean, 0 or 1
 HiDS Location: Control page
 Description: For dual-axis controllers, these variables allow a single control to enable or disable both motors.

10.9 Digital IO

Variable name: DigitalInput1
Units: Boolean, 0 or 1
HiDS Location: Digital IO page
Description: Indicates the state of the digital input signal.

Variable name: DigitalInLowValue1
Units: Float, unbounded
HiDS Location: Digital IO page
Description: If the Digital-input is read as a zero/low, then the assigned variable is set to this value. At a 1Khz rate, the Controller reads the DigitalIn1, if the DigitalIn1 is read as a 0, the assigned variable is set to DigitalInLowValue1; if the digital input is read as a 1, the assigned variable is set to DigitalInHighValue1. To detach (un-assign) a digital input, assign the input to variable "NullVariable" shown on the Utility page.

Variable name: DigitalInHighValue1
Units: Float, unbounded
HiDS Location: Digital IO page
Description: If the Digital-input is read as a zero/low, then the assigned variable is set to this value. At a 1Khz rate, the Controller reads the DigitalIn1, if the DigitalIn1 is read as a 0, the assigned variable is set to DigitalInLowValue1; if the digital input is read as a 1, the assigned variable is set to DigitalInHighValue1. To detach (un-assign) a digital input, assign the input to variable "NullVariable" shown on the Utility page.

Variable name: DigitalOutputX
Units: Boolean, 0 or 1
HiDS Location: Digital IO page
Description: Indicates the state of the digital output signal.

Variable name: DigitalOutThresholdX
Units: Float, unbound
HiDS Location: Digital IO page
Description: Sets the compare-value to use against the assigned HiDS variable. At a 1Khz rate, the Controller compares the assigned variable to DigitalOut1Threshold. If variable > DigitalOut1Threshold, then DigitalOut1 = true. Finally, if the application requires an inverted output, set DigitalOut1Invert = 1 (true).

Variable name: DigitalOutInvertX
Units: Boolean, 0 or 1
HiDS Location: Digital IO page
Description: If 0 (false), the digital-output is driven directly from the compare result with DigitalOutThresholdX. Setting this to 1 (true) inverts the digital output.

Not all Controllers support the Discharge feature. Refer to your Controller's specifications. If supported, the Discharge feature timing can be set on the Digital IO page within the Advanced tab.

Variable name: Discharge.Enable
Units: Boolean, 0 or 1
HiDS Location: Digital IO page
Description: Enables or disables the Discharge feature. If enabled, the Discharge.State variable is set to 1 (true) for Discharge.MaxOn_ms milliseconds on motor-disable. Then Discharge.State variable is set to 0 (false) for Discharge.MaxOff_ms milliseconds after that. This feature could be used with an ESI Motion discharge board or external logic to support a vbus discharge feature.

Variable name: Discharge.State
Units: Boolean, 0 or 1
HiDS Location: Digital IO page
Description: Indicates the state of the discharge output. To be used with external discharge logic, this variable can be bound to an available digital output.

Variable name: Discharge.MaxOn_ms
Units: Time in milliseconds
HiDS Location: Digital IO page
Description: Defines the maximum time that discharge will remain active.

Variable name: Discharge.MaxOff_ms
Units: Time in milliseconds
HiDS Location: Digital IO page
Description: Defines the maximum time that discharge will remain off before it can go active again.

Variable name: DigitalInX_index
Units: Integer
HiDS Location: Digital IO page
Description: Indicates the index of the HiDS-variable that this digital-input is assigned to. The hiDS-variable indexes can be viewed via a Settings->export-all-variables text file.

Variable name: DigitalOutX_index
Units: Integer
HiDS Location: Digital IO page
Description: Indicates the index of the HiDS-variable that this digital-output is assigned to. The hiDS-variable indexes can be viewed via a Settings->export-all-variables text file.

10.10 Encoder

Variable name: EncoderX.Degrees
Units: Degrees, 0-360
HiDS Location: Encoder page
Description: Shows the motor position in degrees, normalized to 0 to 360°.

Variable name: EncoderX.Radians
Units: Radians, valid range is from 0 to 2 π
HiDS Location: Encoder page
Description: Shows the motor position in radians, normalized to 0 to 2 π .

Variable name: EncoderX.RadiansSummed
Units: Radians, unbounded
HiDS Location: Encoder page
Description: Unless explicitly cleared, this variable shows the total accumulated motor position in radians, since power on. Note the sign of the total motor movements is considered in the summing. For example, from 0, a command of +10 radians, plus a command of +5 radians results in RadiansSummed = 5 (because the second movement was -5 radians).

Variable name: EncoderX.RadiansPerSecond
Units: Radians per seconds, unbounded
HiDS Location: Encoder page
Description: Shows the unfiltered motor velocity in radians per second

Variable name: EncoderX.FilteredRadPerSec
Units: Radians per seconds, unbounded
HiDS Location: Encoder page
Description: Shows the filtered motor velocity in radians per second

Variable name: EncoderX.Rpm
Units: Radians per minute
HiDS Location: Encoder page
Description: Shows the unfiltered motor velocity in revolutions per minute.

Variable name: EncoderX.FilteredRpm
Units: Radians per minute
HiDS Location: Encoder page
Description: Shows the filtered motor velocity in revolutions per minute.

Variable name: QepXData.PulsePerRev
Units: Pulses per revolution
HiDS Location: Encoder page
Description: Used to configure how many pulses the encoder outputs per shaft revolution

Variable name: QepXData.SecondaryPulsePerRev
Units: Pulses per revolution
HiDS Location: Encoder page
Description: Used to configure how many pulses the encoder outputs per shaft revolution for the secondary feedback

Variable name: QepXData.EncoderInputType
Units: Boolean, 0 or 1
HiDS Location: Encoder page
Description: Allows user to setup the encoder feedback as either single-ended (0) or differential(1)

Variable name: QepXData.EncoderEnableIndex
Units: Boolean, 0 or 1
HiDS Location: Encoder page
Description: This should be enabled (1) if the encoder used includes an index/reference single. This signal

which a single pulse per revolution used for precise determination of a reference position. Note this can only be used with the QepXData.StartupMethod of 1 (using the Hall to initialize the encoder).

Variable name: QepXData.SetRadSummedOnFirstIndex

Units: Boolean, 0 or 1

HiDS Location: Encoder page

Description: If QepXData.SetRadSummedOnFirstIndex = 1, when the first encoder index is read, the RadiansSummed (accumulated motor) position will be initialized to QepXData.RadiansSummedToIndexOffset.

Variable name: QepXData.RadiansSummedToIndexOffset

Units: Boolean, 0 or 1

HiDS Location: Encoder page

Description: If QepXData.SetRadSummedOnFirstIndex = 1, when the first encoder index is read, the RadiansSummed (accumulated motor) position will be initialized to QepXData.RadiansSummedToIndexOffset.

Variable name: QepXData.StartupMethod

Units: Integer 0, 1, or 2

HiDS Location: Encoder page

Description: If 1, on initial power up the start-up angle will be determined by the Hall sensors. After the start-up angle is determined, only the encoder will be used to track angle. If set to 2, the first motor-start after power-on will result in an open-loop start until the first index is received. 0 disables any start-method.

Variable name: QepXData.OpenLoopRPM

Units: RPM, positive only

HiDS Location: Encoder page

Description: If QepXData.StartupMethod = 2, this sets the open-loop RPM used to start the motor.

Variable name: QepXData.OpenLoopCurrent

Units: Amps, positive only

HiDS Location: Encoder page

Description: If QepXData.StartupMethod = 2, this sets the open-loop current-command used to start the motor; this should generally be ~25% of the motor's maximum current rating.

Variable name: QepXData.HallFeedbackDir

Units: Signed multiplier, the valid values are -1 or 1

HiDS Location: Encoder page

Description: If QepXData.StartupMethod = 1, this variable sets the feedback-direction of the Hall interface alone, which could be different from the Encoder feedback-direction due to a swapped motor-phase or a swapped wire on the quadrature encoder. The valid values are 1 (normal) or -1 (reversed).

Variable name: QepXData.HallPhaseOffset

Units: Degrees, -360° to 360°

HiDS Location: Encoder page

Description: If QepXData.StartupMethod = 1, this variable sets the degree-error between 0 degrees of the Hall

reading and the zero-crossing of the phase-A current -- of the Hall interface alone. This phase-offset could be different from the Encoder phase-offset due to a swapped motor-phase or the physical mounting of the quadrature encoder to the rotor.

Variable name: EncoderEQEP1aln
Units: Boolean, 0 or 1
HiDS Location: Encoder page
Description: Indicates the detected Quadrature encoder 1-A input.

Variable name: EncoderEQEP1bln
Units: Boolean, 0 or 1
HiDS Location: Encoder page
Description: Indicates the detected Quadrature encoder 1-B input.

Variable name: EncoderEQEP1IndexIn
Units: Boolean, 0 or 1
HiDS Location: Encoder page
Description: Indicates the detected Quadrature encoder 1-Index input.

Variable name: EncoderEQEP2aln
Units: Boolean, 0 or 1
HiDS Location: Encoder page
Description: Indicates the detected Quadrature encoder 2-A input.

Variable name: EncoderEQEP2bln
Units: Boolean, 0 or 1
HiDS Location: Encoder page
Description: Indicates the detected Quadrature encoder 2-B input.

Variable name: EncoderEQEP2IndexIn
Units: Boolean, 0 or 1
HiDS Location: Encoder page
Description: Indicates the detected Quadrature encoder 2-Index input.

10.11 Endat Encoder

Variable name: EnDatEncoderFeedbackA.Degrees
Units: Degrees, 0-360
HiDS Location: EnDatEncoder page
Description: Shows the motor position in degrees, normalized to 0 to 360°.

Variable name: EnDatEncoderFeedbackA.Radians
Units: Radians, valid range is from 0 to 2PI
HiDS Location: EnDatEncoder page
Description: Shows the motor position in radians, normalized to 0 to 2π.

Variable name: EnDatEncoderFeedbackA.RadiansSummed

Units: Radians, unbounded
 HiDS Location: EnDatEncoder page
 Description: Unless explicitly cleared, this variable shows the total accumulated motor position in radians, since power on. Note the sign of the total motor movements is considered in the summing. For example, from 0, a command of +10 radians, plus a command of +5 radians results in RadiansSummed = 5 (because the second movement was -5 radians).

Variable name: EnDatEncoderFeedbackA.RadiansPerSecond
 Units: Radians per seconds, unbounded
 HiDS Location: EnDatEncoder page
 Description: Shows the unfiltered motor velocity in radians per second

Variable name: EnDatEncoderFeedbackA.FilteredRadPerSec
 Units: Radians per seconds, unbounded
 HiDS Location: EnDatEncoder page
 Description: Shows the filtered motor velocity in radians per second

Variable name: EnDatEncoderFeedbackA.Rpm
 Units: Radians per minute
 HiDS Location: EnDatEncoder page
 Description: Shows the unfiltered motor velocity in revolutions per minute.

Variable name: EnDatEncoderFeedbackA.FilteredRpm
 Units: Radians per minute
 HiDS Location: EnDatEncoder page
 Description: Shows the filtered motor velocity in revolutions per minute.

10.12 Fan

Note not all controllers support the Fan feature. The Fan state is updated every 1 second.

Variable name: Fan.Hysteresis
 Units: Degrees C, unbounded
 HiDS Location: Fan page
 Description: The fan (digital IO) will be disabled when the measured IGBT temperature is less than (Fan.OnDegreesC - Fan.Hysteresis).

Variable name: Fan.OnDegreesC
 Units: Degrees C, unbounded
 HiDS Location: Fan page
 Description: The fan (digital IO) will be enabled when the measured IGBT temperature is greater than Fan.OnDegreesC.

Variable name: Fan.State
 Units: Boolean, 0 or 1
 HiDS Location: Fan page
 Description: If the controller has a fan, indicates whether the Fan is on (1) or off (0)

Variable name: Fan.FaultState
Units: Boolean, 0 or 1
HiDS Location: Fan page
Description: Indicates whether the fan is not turning on/off. In a faulted state the value will be 1, otherwise 0

Variable name: Fan.FaultLatch
Units: Boolean, 0 or 1
HiDS Location: Fan page
Description: Once a fault is detected it is latched. In order to clear a fault the fan must be in a non-fault condition and the latch cleared (0).

10.13 Fault History

For Controllers that support this feature, the last 4 faults are saved in non-volatile memory for retrieval and analysis after the fact. With each fault saved, a number of the run-time measurements are saved at moment of fault. The description of the run-time values including SecondsSinceReset and the variables from VBus to PositionIntegral can be searched for within this document. In addition, the records saved for each fault are:

Variable name: LastFaultRecordNumber
Units: Integer, 1-4
HiDS Location: Fault History
Description: The software stores up to 4 fault records. This value indicates the previous record number to which a fault occurrence was stored.

Variable name: Fault1 Record Number
Units: Integer, 1-4
HiDS Location: Fault History
Description: Captures the value of the record ID number for this fault record. This allows the user to easily find which faults and in which order they occurred.

Variable name: FH1_AbsoluteFaultNumber
Units: Integer, 0-900
HiDS Location: Fault History
Description: A rolling fault ID.

Variable name: FH1_SecondsSinceReset
Units: Time in seconds
HiDS Location: Fault History
Description: The number of seconds that have elapsed since the controller has been reset or powered up

Variable name: FH1_SystemFaultBitFieldLsw
Units: Integer
HiDS Location: Fault History
Description: Holds the lower 16 bits of the 32 bit fault status word (least significant word). Each bit represents a different fault and a value of 0 indicates no faults, the fault breakdown is as follows:
Bit0: DSP Overtemperature
Bit1: VBUS Overvoltage
Bit2: VBUS Undervoltage

Bit3: Precharge Fault
 Bit4: None
 Bit5: CPU2 Execution Fault
 Bit6: External Memory Fault
 Bit7: Logic power undervoltage
 Bit8: Servo-A IGBT Overtemperature
 Bit9: Servo-A Loss of feedback
 Bit10: Motor-A Overtemperature
 Bit11: Servo-A Overcurrent
 Bit12: Motor-A Overspeed
 Bit13: Motor-A Tracking Fault
 Bit14: Servo-A Bridgefault
 Bit15: Servo-A I2T Fault

Variable name: FH1_SystemFaultBitFieldMsw
 Units: Integer
 HiDS Location: Fault History
 Description: Holds the lower 16 bits of the 32 bit fault status word (least significant word). Each bit represents a different fault and a value of 0 indicates no faults, the fault breakdown is as follows:
 Bit16: Servo-B IGBT Overtemperature
 Bit17: Servo-B Loss of feedback
 Bit18: Motor-B Overtemperature
 Bit19: Servo-B Overcurrent
 Bit20: Motor-B Overspeed
 Bit21: Motor-B Tracking Fault
 Bit22: Servo-B Bridgefault
 Bit23: Servo-B I2T Fault

Variable name: FH1_VBus
 Units: Volts
 HiDS Location: Fault History
 Description: Captures the VBUS voltage at the point the fault record was created

Variable name: FH1_MalqCmd
 Units: Amps
 HiDS Location: Fault History
 Description: Captures the Iq command (current command into the Torque loop) at the point the fault record was created

Variable name: FH1_MaRpmCommand
 Units: Revolutions per minute
 HiDS Location: Fault History
 Description: Captures the RPM command (velocity command into the velocity loop) at the point the fault record was created

Variable name: FH1_MaRadiansCommand
 Units: Radians
 HiDS Location: Fault History

Description: Captures the Radians commands (position command into the position loop) at the point the fault record was created

Variable name: FH1_MalGBTTemp

Units: Degrees Celsius

HiDS Location: Fault History

Description: Captures the Servo-A IGBT temperature at the point the fault record was created

Variable name: FH1_MaMotorTemp

Units: Degrees Celsius

HiDS Location: Fault History

Description: Captures the Motor-A temperature at the point the fault was recorded

Variable name: FH1_MalqErrorIntegral

Units: Volts, bounded by Mx.IntegralMin and Mx.IntegralMax

HiDS Location: Fault History

Description: Captures the IQ integral value of the current loop for Motor-A at the point the fault was record was created

Variable name: FH1_MalErrorIntegral

Units: Volts, bounded by Mx.IntegralMin and Mx.IntegralMax

HiDS Location: Fault History

Description: Captures the ID integral value of the current loop for Motor-A at the point the fault was record was created

Variable name: FH1_MaPreSaturation

Units: Boolean, 0 or 1

HiDS Location: Fault History

Description: Indicates whether the servo was in a pre-saturated state at the point the fault record was created. This "saturated" condition is often a warning to an improperly controlled motor or possibly an error-condition.

Variable name: FH1_MaVelocityIntegral

Units: Amps, bounded by Mx.VelocityIntegralMin and Mx.VelocityIntegralMax.

HiDS Location: Fault History

Description: Captures the velocity loop current integral value at the point the fault record was created.

Variable name: FH1_MaPositionIntegral

Units: Radians, bounded by Mx.PositionIntegralMin and Mx.PositionIntegralMax.

HiDS Location: Fault History

Description: Captures the position loop current integral value at the point the fault record was created.

Variable name: FH1_MblqCmd

Units: Amps

HiDS Location: Fault History

Description: Captures the Iq command (current command into the Torque loop) at the point the fault record was created

Variable name: FH1_MbRpmCommand

Units:	Revolutions per minute
HiDS Location:	Fault History
Description:	Captures the RPM command (velocity command into the velocity loop) at the point the fault record was created
Variable name:	FH1_MbIGBTTemp
Units:	Degrees Celsius
HiDS Location:	Fault History
Description:	Captures the Motor-B IGBT or MOSFET temperature at the point the fault record was created
Variable name:	FH1_MbMotorTemp
Units:	Degrees Celsius
HiDS Location:	Fault History
Description:	Captures the Motor-B temperature at the point the fault was recorded
Variable name:	FH1_MbIqErrorIntegral
Units:	Volts, bounded by Mx.IntegralMin and Mx.IntegralMax
HiDS Location:	Fault History
Description:	Captures the IQ integral value of the current loop for Motor-B at the point the fault was record was created
Variable name:	FH1_MbIdErrorIntegral
Units:	Volts, bounded by Mx.IntegralMin and Mx.IntegralMax
HiDS Location:	Fault History
Description:	Captures the ID integral value of the current loop for Motor-B at the point the fault was record was created
Variable name:	FH1_MbPreSaturation
Units:	Boolean, 0 or 1
HiDS Location:	Fault History
Description:	Indicates whether the servo was in a pre-saturated state at the point the fault record was created. This "saturated" condition is often a warning to an improperly controlled motor or possibly an error-condition.
Variable name:	FH1_MbVelocityIntegral
Units:	Amps, bounded by Mx.VelocityIntegralMin and Mx.VelocityIntegralMax.
HiDS Location:	Fault History
Description:	Captures the velocity loop current integral value at the point the fault record was created.
Variable name:	Fault2 Record Number
Units:	Integer, 1-4
HiDS Location:	Fault History
Description:	Captures the value of the record ID number for this fault record. This allows the user to easily find which faults and in which order they occurred.
Variable name:	Fault3 Record Number
Units:	Integer, 1-4
HiDS Location:	Fault History
Description:	Captures the value of the record ID number for this fault record. This allows the user to easily find

which faults and in which order they occurred.

Variable name: Fault4 Record Number
Units: Integer, 1-4
HiDS Location: Fault History
Description: Captures the value of the record ID number for this fault record. This allows the user to easily find which faults and in which order they occurred.

Variable name: ClearFaultRecords
Units: Boolean, 0 or 1
HiDS Location: Fault History
Description: If 1, Clears all four fault records and resets LastFaultRecordNumber

10.14 Fault Inputs

For each of the BITs (Built In Tests), a latched fault bit can occur. If the Run Panel shows a fault, or if the motor fails to run, these fault flags will indicate what failure has occurred.

If after clicking [Reset] on the Run Panel, and one or more fault flags fail to clear, then this is a persistent failure condition that must be resolved before continuing.

All of the following variables are located on the Fault Inputs page in the Advanced Tab:

Variable name: ResetFaultFlags
Units: Boolean, 0 or 1
HiDS Location: Fault Inputs
Description: If 1, clears all latched faults

Variable name: SystemFaultState
Units: Boolean, 0 or 1
HiDS Location: Fault Inputs
Description: Indicates whether any faults are latched on either Servo A or B

Variable name: LastFaultBitField
Units: 32 bit word
HiDS Location: Fault Inputs
Description: Saves the last reported fault status. Each bit represents a different fault and a value of 0 indicates no faults, the fault breakdown is as follows:
Bit0: DSP Overtemperature
Bit1: VBUS Overvoltage
Bit2: VBUS Undervoltage
Bit3: Precharge Fault
Bit4: None
Bit5: CPU2 Execution Fault
Bit6: External Memory Fault
Bit7: Logic power undervoltage
Bit8: Servo-A IGBT Overtemperature

Bit9: Servo-A Loss of feedback
 Bit10: Motor-A Overtemperature
 Bit11: Servo-A Overcurrent
 Bit12: Motor-A Overspeed
 Bit13: Motor-A Tracking Fault
 Bit14: Servo-A Bridgefault
 Bit15: Servo-A I2T Fault
 Bit16: Servo-B IGBT Overtemperature
 Bit17: Servo-B Loss of feedback
 Bit18: Motor-B Overtemperature
 Bit19: Servo-B Overcurrent
 Bit20: Motor-B Overspeed
 Bit21: Motor-B Tracking Fault
 Bit22: Servo-B Bridgefault
 Bit23: Servo-B I2T Fault

Variable name: LastFaultHighWord
 Units: 16 bit word
 HiDS Location: Fault Inputs
 Description: Stores the 16 most significant bits of LastFaultBitField

Variable name: LastFaultLowWord
 Units: 16 bit word
 HiDS Location: Fault Inputs
 Description: Stores the 16 least significant bits of LastFaultBitField

Variable name: MotorAFaultState
 Units: Boolean, 0 or 1
 HiDS Location: Fault Inputs
 Description: Indicates whether Servo-A is in a fault condition

Variable name: MotorBFaultState
 Units: Boolean, 0 or 1
 HiDS Location: Fault Inputs
 Description: Indicates whether Servo-B is in a fault condition

Variable name: VbusOverVoltage
 Units: Boolean, 0 or 1
 HiDS Location: Fault Inputs
 Description: Indicates whether there an over voltage condition for the either of the motor bus voltage /motor power supply voltage

Variable name: VbusUnderVoltage
 Units: Boolean, 0 or 1
 HiDS Location: Fault Inputs
 Description: Indicates whether there is an under voltage condition for either of the motor bus voltage /motor power supply voltage

Variable name: PrechargeUndervoltageFailure

Units:	Boolean, 0 or 1
HiDS Location:	Fault Inputs
Description:	Indicates if during motor startup (either A or B), the measured Vbus was less than the Inrush.VbusOnThreshold threshold. This failure often indicates a disconnected or zero-voltage VBus (motor power supply).
Variable name:	VLogicUnderVoltage
Units:	Boolean, 0 or 1
HiDS Location:	Fault Inputs
Description:	Indicates an under voltage condition for logic power.
Variable name:	MotorHWOvercurrent
Units:	Boolean, 0 or 1
HiDS Location:	Fault Inputs
Description:	Indicates whether the measured current for either servo exceeds the value set by Mx.HardwareOvercurrent. The fault is controlled by hardware and is not configurable via the Fault Inputs
Variable name:	MotorOverspeed
Units:	Boolean, 0 or 1
HiDS Location:	Fault Inputs
Description:	Indicates if the measured/calculated RPM does exceeds the specified limit
Variable name:	MotorOverTemp
Units:	Boolean, 0 or 1
HiDS Location:	Fault Inputs
Description:	Indicates if either motors temperature exceeds the specified limit
Variable name:	IGBTOverTemperature
Units:	Boolean, 0 or 1
HiDS Location:	Fault Inputs
Description:	Indicates if either motors temperature exceeded the specified limit. See the Fault Inputs for details
Variable name:	MotorLossOfFeedback
Units:	Boolean, 0 or 1
HiDS Location:	Fault Inputs
Description:	Indicates whether there is a loss of feedback for either servo. See the Fault Inputs for details
Variable name:	I2T
Units:	Boolean, 0 or 1
HiDS Location:	Fault Inputs
Description:	Indicates if there is an I2T fault on either servo. I2T (read I-squared-T) is an estimation of the energy content in current transient conditions. Because a motor can often handle short current transients in excess of rated currents, this test can help protect against motor overheating or damage due to excess current transients. See the Fault Inputs for details
Variable name:	DSPOverTemperature
Units:	Boolean, 0 or 1
HiDS Location:	Fault Inputs

Description:	Indicates an over temperature condition of the internal DSP (Digital Signal Processor)
Variable name:	Cpu2Execution
Units:	Boolean, 0 or 1
HiDS Location:	Fault Inputs
Description:	Indicates if the 2nd CPU is not running if its expected to.
Variable name:	BridgeControlFault
Units:	Boolean, 0 or 1
HiDS Location:	Fault Inputs
Description:	Excessive power running through MOSFET/IGBT, motor will shutdown. Also known as a De-saturation fault.
Variable name:	ExternalFaultTrip
Units:	Boolean, 0 or 1
HiDS Location:	Fault Inputs
Description:	Indicates an external event caused a fault condition; often mapped to a serial/CAN-command interface timeout.
Variable name:	AccessoryIO
Units:	Boolean, 0 or 1
HiDS Location:	Fault Inputs
Description:	Excessive power running through, or an internal failure, of the internal Fan, Inrush, Regen, and external-5V switches.
Variable name:	MotorAHWOvercurrent
Units:	Boolean, 0 or 1
HiDS Location:	Fault Inputs
Description:	Indicates whether the measured current for servo A exceeds the value set by Ma.HardwareOvercurrent. The fault is controlled by hardware and is not configurable via the Fault Inputs
Variable name:	MotorAOverspeed
Units:	Boolean, 0 or 1
HiDS Location:	Fault Inputs
Description:	Indicates if the the measured/calculated RPM does exceeds the specified limit
Variable name:	MotorAOverTemp
Units:	Boolean, 0 or 1
HiDS Location:	Fault Inputs
Description:	Indicates if the motor-A temperature exceeds the specified limit
Variable name:	MalGBTOverTemperature
Units:	Boolean, 0 or 1
HiDS Location:	Fault Inputs
Description:	Indicates if the motor-A IGBT/MOSFET temperature exceeded the specified limit. See the Fault Inputs for details
Variable name:	MotorALossOfFeedback

Units:	Boolean, 0 or 1
HiDS Location:	Fault Inputs
Description:	Indicates whether there is a loss of feedback for servo A. See the Fault Inputs for details
Variable name:	MotorAI2T
Units:	Boolean, 0 or 1
HiDS Location:	Fault Inputs
Description:	Indicates if there is an I2T fault on servo A. I2T (read I-squared-T) is an estimation of the energy content in current transient conditions. Because a motor can often handle short current transients in excess of rated currents, this test can help protect against motor overheating or damage due to excess current transients. See the Fault Inputs for details
Variable name:	MotorABridgeFault
Units:	Boolean, 0 or 1
HiDS Location:	Fault Inputs
Description:	Excessive power running through MOSFET/IGBT, motor will shutdown. Also known as a De-saturation fault.
Variable name:	MaVelocityTracking
Units:	Boolean, 0 or 1
HiDS Location:	Fault Inputs
Description:	Indicates that the RPM-error (commanded - actual) exceeded the specified limit for Servo A.
Variable name:	MaPositionTracking
Units:	Boolean, 0 or 1
HiDS Location:	Fault Inputs
Description:	Indicates that the absolute motor (rotational) position exceeded the specified limit for Servo A.
Variable name:	MotorBHWOvercurrent
Units:	Boolean, 0 or 1
HiDS Location:	Fault Inputs
Description:	Indicates whether the measured current for servo B exceeds the value set by Mb.HardwareOvercurrent. The fault is controlled by hardware and is not configurable via the Fault Inputs
Variable name:	MotorBOverspeed
Units:	Boolean, 0 or 1
HiDS Location:	Fault Inputs
Description:	Indicates if the measured/calculated RPM does exceeds the specified limit
Variable name:	MotorBOverTemp
Units:	Boolean, 0 or 1
HiDS Location:	Fault Inputs
Description:	Indicates if the motor-B temperature exceeds the specified limit
Variable name:	MbIGBTOverTemperature
Units:	Boolean, 0 or 1
HiDS Location:	Fault Inputs
Description:	Indicates if the motor-A IGBT/MOSFET temperature exceeded the specified limit. See the Fault

Inputs for details

Variable name:	MotorBLossOfFeedback
Units:	Boolean, 0 or 1
HiDS Location:	Fault Inputs
Description:	Indicates whether there is a loss of feedback for servo B. See the Fault Inputs for details
Variable name:	MotorBI2T
Units:	Boolean, 0 or 1
HiDS Location:	Fault Inputs
Description:	Indicates if there is an I2T fault on servo B. I2T (read I-squared-T) is an estimation of the energy content in current transient conditions. Because a motor can often handle short current transients in excess of rated currents, this test can help protect against motor overheating or damage due to excess current transients. See the Fault Inputs for details
Variable name:	MotorBBridgeFault
Units:	Boolean, 0 or 1
HiDS Location:	Fault Inputs
Description:	Excessive power running through MOSFET/IGBT, motor will shutdown. Also known as a De-saturation fault.
Variable name:	MbVelocityTracking
Units:	Boolean, 0 or 1
HiDS Location:	Fault Inputs
Description:	Indicates that the RPM-error (commanded - actual) exceeded the specified limit for Servo B.
Variable name:	MbPositionTracking
Units:	Boolean, 0 or 1
HiDS Location:	Fault Inputs
Description:	Indicates that the absolute motor (rotational) position exceeded the specified limit for Servo B.
Variable name:	SystemFaultStateWarning
Units:	Boolean, 0 or 1
HiDS Location:	Fault Inputs
Description:	Indicates if any of the warning indicators for either Servo are active.
Variable name:	MotorAFaultStateWarning
Units:	Boolean, 0 or 1
HiDS Location:	Fault Inputs
Description:	Indicates if any warning indicators are active for Servo-A.
Variable name:	MotorBFaultStateWarning
Units:	Boolean, 0 or 1
HiDS Location:	Fault Inputs
Description:	Indicates if any warning indicators are active for Servo-A.
Variable name:	WarningsClearEverySeconds
Units:	Time in seconds
HiDS Location:	Fault Inputs

Description:	Sets the time interval at which warnings are cleared.
Variable name:	VbusOverVoltageWarning
Units:	Boolean, 0 or 1
HiDS Location:	Fault Inputs
Description:	The warning level was exceeded for the VbusOverVoltage fault.
Variable name:	VbusUnderVoltageWarning
Units:	Boolean, 0 or 1
HiDS Location:	Fault Inputs
Description:	The warning level was exceeded for the VbusUnderVoltage fault.
Variable name:	PrechargeUndervoltageWarning
Units:	Boolean, 0 or 1
HiDS Location:	Fault Inputs
Description:	The warning level was exceeded for the PrechargeUndervoltage fault.
Variable name:	VLogicUnderVoltageWarning
Units:	Boolean, 0 or 1
HiDS Location:	Fault Inputs
Description:	The warning level was exceeded for the VLogicUnderVoltage fault.
Variable name:	MotorHWOvercurrentWarning
Units:	Boolean, 0 or 1
HiDS Location:	Fault Inputs
Description:	The warning level was exceeded for the MotorHWOvercurrent fault.
Variable name:	MotorOverspeedWarning
Units:	Boolean, 0 or 1
HiDS Location:	Fault Inputs
Description:	The warning level was exceeded for the MotorOverspeed fault.
Variable name:	MotorOverTempWarning
Units:	Boolean, 0 or 1
HiDS Location:	Fault Inputs
Description:	The warning level was exceeded for the MotorOverTemp fault.
Variable name:	IGBTOverTemperatureWarning
Units:	Boolean, 0 or 1
HiDS Location:	Fault Inputs
Description:	The warning level was exceeded for the IGBTOverTemperature fault.
Variable name:	MotorLossOfFeedbackWarning
Units:	Boolean, 0 or 1
HiDS Location:	Fault Inputs
Description:	The warning level was exceeded for the MotorLossOfFeedback fault.
Variable name:	I2TWarning
Units:	Boolean, 0 or 1

HiDS Location:	Fault Inputs
Description:	The warning level was exceeded for the I2T fault.
Variable name:	DSPOverTemperatureWarning
Units:	Boolean, 0 or 1
HiDS Location:	Fault Inputs
Description:	The warning level was exceeded for the DSPOverTemp fault.
Variable name:	AccessoryIOWarning
Units:	Boolean, 0 or 1
HiDS Location:	Fault Inputs
Description:	The warning level was exceeded for the AccessoryIO.
Variable name:	Cpu2ExecutionWarning
Units:	Boolean, 0 or 1
HiDS Location:	Fault Inputs
Description:	The warning level was exceeded for the Cpu2Execution fault.
Variable name:	MotorAHWOvercurrentWarning
Units:	Boolean, 0 or 1
HiDS Location:	Fault Inputs
Description:	The warning level was exceeded for the MotorAHWOvercurrent fault.
Variable name:	MotorAOverspeedWarning
Units:	Boolean, 0 or 1
HiDS Location:	Fault Inputs
Description:	The warning level was exceeded for the MotorAOverspeed fault.
Variable name:	MotorAOverTempWarning
Units:	Boolean, 0 or 1
HiDS Location:	Fault Inputs
Description:	The warning level was exceeded for the MotorAOverTemp fault.
Variable name:	MaIGBTOverTemperatureWarning
Units:	Boolean, 0 or 1
HiDS Location:	Fault Inputs
Description:	The warning level was exceeded for the MaIGBTOverTemperature fault.
Variable name:	MotorALossOfFeedbackWarning
Units:	Boolean, 0 or 1
HiDS Location:	Fault Inputs
Description:	The warning level was exceeded for the MotorALossOfFeedback fault.
Variable name:	MaVelocityTrackingWarning
Units:	Boolean, 0 or 1
HiDS Location:	Fault Inputs
Description:	The warning level was exceeded for the MaVelocityTracking fault.
Variable name:	MaPositionTrackingWarning

Units:	Boolean, 0 or 1
HiDS Location:	Fault Inputs
Description:	The warning level was exceeded for the MaPositionTracking fault.
Variable name:	MotorBHWOVercurrentWarning
Units:	Boolean, 0 or 1
HiDS Location:	Fault Inputs
Description:	The warning level was exceeded for the MotorBHWOVercurrent fault.
Variable name:	MotorBOverspeedWarning
Units:	Boolean, 0 or 1
HiDS Location:	Fault Inputs
Description:	The warning level was exceeded for the MotorBOverspeed fault.
Variable name:	MotorBOverTempWarning
Units:	Boolean, 0 or 1
HiDS Location:	Fault Inputs
Description:	The warning level was exceeded for the MotorBOverTemp fault.
Variable name:	MbIGBTOverTemperatureWarning
Units:	Boolean, 0 or 1
HiDS Location:	Fault Inputs
Description:	The warning level was exceeded for the MbIGBTOverTemperature fault.
Variable name:	MotorBLossOfFeedbackWarning
Units:	Boolean, 0 or 1
HiDS Location:	Fault Inputs
Description:	The warning level was exceeded for the MotorBLossOfFeedback fault.
Variable name:	MbVelocityTrackingWarning
Units:	Boolean, 0 or 1
HiDS Location:	Fault Inputs
Description:	The warning level was exceeded for the MbVelocityTracking fault.
Variable name:	MbPositionTrackingWarning
Units:	Boolean, 0 or 1
HiDS Location:	Fault Inputs
Description:	The warning level was exceeded for the MbPositionTracking fault.
Variable name:	IOFault.MbBridgeFault
Units:	Boolean, 0 or 1
HiDS Location:	Fault Inputs
Description:	Shows the raw digital input from the IGBT/MOSFET fault/de-saturation line.

10.15FPGA (Hyperion Only)

Variable name:	FPGABuildDay/Hour/Minute/Year
HiDS location:	Shown on FPGA page in Advanced tab.

Description: Read Only. Shows the build date of the current FPGA "firmware".
Units/Bounds: Date.

10.16Hall

Variable name: HallX.Degrees
Units: Degrees, 0-360
HiDS Location: Hall page
Description: Shows the motor position in degrees, normalized to 0-360°

Variable name: HallX.Radians
Units: Radians, 0 to 2π
HiDS Location: Hall page
Description: Shows the motor position in radians, normalized to 0 to 2π .

Variable name: HallX.RadiansSummed
Units: Radians, unbounded
HiDS Location: Hall page
Description: Unless explicitly cleared, this variable shows the total accumulated motor position in radians, since power on. Note the sign of the total motor movements is considered in the summing. For example, from 0, a command of +10 radians, plus a command of +5 radians results in RadiansSummed = 5 (because the second movement was -5 radians).

Variable name: HallX.RadiansPerSecond
Units: Radians per seconds, unbounded
HiDS Location: Hall page
Description: Shows the unfiltered motor velocity in radians per second

Variable name: HallX.FilteredRadPerSec
Units: Radians per seconds, unbounded
HiDS Location: Hall page
Description: Shows the filtered motor velocity in radians per second

Variable name: HallX.RPM
Units: Radians per minute
HiDS Location: Hall page
Description: Shows the unfiltered motor velocity in revolutions per minute.

Variable name: HallX.FilteredRPM
Units: Radians per minute
HiDS Location: Hall page
Description: Shows the filtered motor velocity in revolutions per minute.

Variable name: HallX.U
Units: Boolean, 0 or 1
HiDS Location: Hall page
Description: Indicates the detected Hall U digital input.

Variable name: HallX.V
Units: Boolean, 0 or 1
HiDS Location: Hall page
Description: Indicates the detected Hall V digital input.

Variable name: HallX.W
Units: Boolean, 0 or 1
HiDS Location: Hall page
Description: Indicates the detected Hall W digital input.

10.17 Inrush

Not all Controllers support the Inrush feature. Refer to your Controller's specifications. If supported, the Inrush feature will usually be enabled by default, and the Inrush timing can be set on the Inrush page within the Advanced tab.

Variable name: Inrush.Enable
Units: Boolean, 0 or 1
HiDS Location: Inrush page
Description: Enables or disables the Inrush feature. If enabled, the motor-state machine will enable the precharge switch, then wait approximately 0.01seconds times Inrush.MinCount for the bus-voltage to charge before enabling the bus-switch.

Variable name: Inrush.MinCount
Units: 0.01 seconds delay per integer value, must be ≥ 0
HiDS Location: Inrush page
Description: If Inrush.enable = 1, the motor-state machine will enable the precharge-switch, then wait approximately 0.01seconds times Inrush.MinCount for the bus-voltage to charge before enabling the bus-switch.

Variable name: Inrush.Precharge
Units: Boolean, 0 or 1
HiDS Location: Inrush page
Description: 0 indicates the precharge switch is open; 1 indicates it is closed.

Variable name: Inrush.BusOn
Units: Boolean, 0 or 1
HiDS Location: Inrush page
Description: 0 indicates the bus switch is open; 1 indicates it is closed.

Variable name: Inrush.VbusOnThreshold
Units: Volts, unbounded
HiDS Location: Inrush page
Description: If Inrush.enable = 1, the motor-state machine will enable the precharge-switch, then wait approximately 0.01seconds times Inrush.MinCount for the bus-voltage to charge before enabling the bus-switch. After this delay, the Vbus measurement must be $>$ Inrush.VbusOnThreshold, otherwise a precharge-under-voltage fault is thrown.

10.18Limits

Many of the measurement limits can be configured in the [Limits] tab, but the Motor B limits are not shown there. To change the motor B limits, adjust the Motor B variables shown on the Limits page in the Advanced Tab.

Variable name: Ma.MaxCurrentCommand
Units: Amps, Mx.MaxCurrentCommand should be positive; Mx.MinCurrentCommand should be negative. The bounds are application specific, but should be less than the over-current value (Mx.HardwareOvercurrent).
HiDS Location: Limits page
Description: Used to bound the current command into the current-loop. If running in Torque mode, these limits simply restrict the current command. If running in Velocity or Position modes, if the output from the Velocity loop reaches these limits, then velocity (or position) loop integrals may build up, which may lead to an over-current fault condition.

Variable name: Ma.MinCurrentCommand
Units: Amps, Mx.MaxCurrentCommand should be positive; Mx.MinCurrentCommand should be negative. The bounds are application specific, but should be less than the over-current value (Mx.HardwareOvercurrent).
HiDS Location: Limits page
Description: Used to bound the current command into the current-loop. If running in Torque mode, these limits simply restrict the current command. If running in Velocity or Position modes, if the output from the Velocity loop reaches these limits, then velocity (or position) loop integrals may build up, which may lead to an over-current fault condition.

Variable name: Ma.HardwareOvercurrent
Units: Amps, the maximum current rating is defined by the product specification.
HiDS Location: Limits page
Description: Used to detect the over-current fault condition. The hardware overcurrent protects the controller from instantaneous phase-currents that could be caused by a phase-short to chassis or ground, but the over-current fault can also be tripped by setting the limit too low or control-loop instability. This limit cannot be configured over the maximum rating of the controller.

Note Hardware overcurrent can also be a common fault on the controller due to loop-instabilities, grounding problems, or hi-pot failures.

Secondary note: The controller measures current for a brief time (<1us) every 25us, which is what the HiDS scope shows. However the hardware-overcurrent reacts to nanosecond-duration events, so a low-inductance motor with large current-ripple can have much larger currents than show up in HiDS. Trouble-shooting over-current faults usually requires measuring the current with an external current probe with time-scales sub 1us.

Variable name: ConfigMax.MaCurrentCommand
Units: Amps
HiDS Location: Limits page
Description: Defines the absolute maximum current command for Mx (serves as max bound for Mx.MaxCurrentCommand and Mx.MinCurrentCommand)

Variable name:	ConfigMax.MbCurrentCommand
Units:	Amps
HiDS Location:	Limits page
Description:	Defines the absolute maximum current command for Mx (serves as max bound for Mx.MaxCurrentCommand and Mx.MincurrentCommand)
Variable name:	ConfigMax.MaHwOvercurrent
Units:	Amps
HiDS Location:	Limits page
Description:	Defines the absolute maximum limit threshold for a hardware overcurrent fault (serves as a max bound for Mx.HardwareOvercurrent)
Variable name:	ConfigMax.MbHwOvercurrent
Units:	Amps
HiDS Location:	Limits page
Description:	Defines the absolute maximum limit threshold for a hardware overcurrent fault (serves as a max bound for Mx.HardwareOvercurrent)
Variable name:	ConfigMax.DspTemp
Units:	Degrees Celsius
HiDS Location:	Limits page
Description:	Defines the absolute maximum limit threshold for a DSP over temperature fault.
Variable name:	ConfigMax.MalgbtTemp
Units:	Degrees Celsius
HiDS Location:	Limits page
Description:	Defines the absolute maximum limit threshold for an IGBT over temperature fault
Variable name:	ConfigMax.MblgbtTemp
Units:	Degrees Celsius
HiDS Location:	Limits page
Description:	Defines the absolute maximum limit threshold for an IGBT over temperature fault
Variable name:	ConfigMax.VbusMaxLimit
Units:	Volts
HiDS Location:	Limits page
Description:	Defines the absolute maximum limit threshold for a VBUS over voltage fault

10.19 LoopInputs

The LoopInputs page lists the various variables defining the input-source multiplexers for the Current, Velocity, and Position loop inputs.

Variable name:	IqCmdSrcSelect1
Units:	Integer, 0-7
HiDS Location:	LoopInputs page
Description:	This variable is typically updated via the input source drop-down menus under the [LoopX Gains] tab; it displays the first user selection for the torque loop command input multiplexer. The output of the two command input multiplexers (Mx.IqCmdSrc1 and Mx.IqCmdSrc2) are summed together

to produce Ma.IqCmd. The following options are available:

- 0=No input selected
- 1=Internal SinTest
- 2=IqUserCmd (HiDS RunPanel)
- 3=CAN Interface
- 4=RS-422 Interface
- 5=Analog Input via COS_MA line
- 6=Analog Input via SIN_MA line
- 7=Analog Input via CMD_MA line

Variable name: IqCmdSrcSelect2
Units: Integer, 0-7
HiDS Location: LoopInputs page
Description: This variable is typically updated via the input source drop-down menus under the [LoopX Gains] tab; it is the second user selection for the torque loop command input multiplexer. The output of the two command input multiplexers (Mx.IqCmdSrc1 and Mx.IqCmdSrc2) are summed together to produce Ma.IqCmd]

- 0=No input selected
- 1=Internal SinTest
- 2=IqUserCmd (HiDS RunPanel)
- 3=CAN Interface
- 4=RS-422 Interface
- 5=Analog Input via COS_MA line
- 6=Analog Input via SIN_MA line
- 7=Analog Input via CMD_MA line

Variable name: Mx.IqCmdSrc1
Units: Amps
HiDS Location: LoopInputs page
Description: One of the two outputs from the Servo-A/B torque loop command input multiplexer

Variable name: Mx.IqCmdSrc2
Units: Amps
HiDS Location: LoopInputs page
Description: One of the two outputs from the Servo-A/B torque loop command input multiplexer

Variable name: VelCmdSrcSelect1
Units: Integer, 0-7
HiDS Location: LoopInputs page
Description: This variable is typically updated via the input source drop-down menus under the [LoopX Gains] tab; it displays the first user selection for the velocity loop command input multiplexer. The output of the two command input multiplexers (MxVL.VelCmdSrc1 and MxVL.VelCmdSrc2) are summed together to produce MxVL.RPMCommand. The following options are available:

- 0=No input selected
- 1=Internal SinTest
- 2=RPMUserCmd (HiDS RunPanel)
- 3=CAN Interface
- 4=RS-422 Interface
- 5=Analog Input via COS_MA line

6=Analog Input via SIN_MA line

7=Analog Input via CMD_MA line

Variable name: VelCmdSrcSelect2
Units: Integer, 0-7
HiDS Location: LoopInputs page
Description: This variable is typically updated via the input source drop-down menus under the [LoopX Gains] tab; it displays the second user selection for the velocity loop command input multiplexer. The output of the two command input multiplexers (MxVL.VelCmdSrc1 and MxVL.VelCmdSrc2) are summed together to produce MxVL.RPMCommand. The following options are available:
0=No input selected
1=Internal SinTest
2=RPMUserCmd (HiDS RunPanel)
3=CAN Interface
4=RS-422 Interface
5=Analog Input via COS_MA line
6=Analog Input via SIN_MA line
7=Analog Input via CMD_MA line

Variable name: MxVL.VelCmdSrc1
Units: RPM, unbounded
HiDS Location: LoopInputs page
Description: One of the two outputs from the Servo-A velocity loop command input multiplexer

Variable name: MxVL.VelCmdSrc2
Units: RPM, unbounded
HiDS Location: LoopInputs page
Description: One of the two outputs from the Servo-A velocity loop command input multiplexer

Variable name: PosCmdSrcSelect1
Units: Integer, 0-7
HiDS Location: LoopInputs page
Description: This variable is typically updated via the input source drop-down menus under the [LoopX Gains] tab; it displays the first user selection for the position loop command input multiplexer. The output of the two command input multiplexers (MxPL.PosCmdSrc1 and MxPL.PosCmdSrc2) are summed together to produce MxPL.RadiansUserCommand. The following options are available:
0=No input selected
1=Internal SinTest
2=RadiansUserCmd (HiDS RunPanel)
3=CAN Interface
4=RS-422 Interface
5=Analog Input via COS_MA line
6=Analog Input via SIN_MA line
7=Analog Input via CMD_MA line

Variable name: PosCmdSrcSelect2
Units: Integer, 0-7
HiDS Location: LoopInputs page
Description: This variable is typically updated via the input source drop-down menus under the [LoopX Gains]

tab; it displays the second user selection for the position loop command input multiplexer. The output of the two command input multiplexers (MxPL.PosCmdSrc1 and MxPL.PosCmdSrc2) are summed together to produce MxPL.RadiansUserCommand. The following options are available:

0=No input selected

1=Internal SinTest

2=RadiansUserCmd (HiDS RunPanel)

3=CAN Interface

4=RS-422 Interface

5=Analog Input via COS_MA line

6=Analog Input via SIN_MA line

7=Analog Input via CMD_MA line

Variable name: MxPL.PosCmdSrc1

Units: Radians

HiDS Location: LoopInputs page

Description: One of the two outputs from the Servo-A position loop command input multiplexer

Variable name: MxPL.PosCmdSrc2

Units: Radians

HiDS Location: LoopInputs page

Description: One of the two outputs from the Servo-A position loop command input multiplexer

Variable name: SinOut.IqTestOutput

Units: Sinusoidal, 1 to -1

HiDS Location: LoopInputs page

Description: The product of SinOut.output and SinTest.iq_amplitude. If Internal SinTest is selected for the torque command input multiplexer, this will be the multiplexer output.

Variable name: SinOut.VelTestOutput

Units: Sinusoidal, 1 to -1

HiDS Location: LoopInputs page

Description: The product of SinOut.output and SinTest.VelocityCommandAmp. If Internal SinTest is selected for the torque command input multiplexer, this will be the multiplexer output.

Variable name: SinOut.PosTestOutput

Units: Sinusoidal, 1 to -1

HiDS Location: LoopInputs page

Description: The product of SinOut.output and SinTest.position_amp. If Internal SinTest is selected for the torque command input multiplexer, this will be the multiplexer output.

Variable name: ResolverXSinICmd

Units: Sinusoidal, 2.5 to -2.5

HiDS Location: LoopInputs page

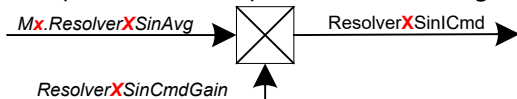
Description: The product of Mx.ResolverXSinAvg and ResolverXSinCmdGain. If SinA Analog command is selected from the input multiplexer, this will be the multiplexer output.

Variable name: ResolverXSinCmdGain

Units: Unitless multiplier

HiDS Location: LoopInputs page

Description: Specifies how much gain to apply to the Resolver SinA analog input.



Variable name: ResolverXCoslCmd

Units: Sinusoidal, 2.5 to -2.5

HiDS Location: LoopInputs page

Description: The product of Mx.ResolverXSinAvg and ResolverXSinCmdGain. If CosA Analog command is selected from the input multiplexer, this will be the multiplexer output.

Variable name: ResolverXCosCmdGain

Units: Sinusoidal, 2.5 to -2.5

HiDS Location: LoopInputs page

Description: Specifies how much gain to apply to the Resolver CosA analog input.

Variable name: Mx.lcmdInput

Units: Volts

HiDS Location: LoopInputs page

Description: Displays the measured voltage from the external Analog input pins. The raw voltage is scaled by Mx.lcmdInputScale

Variable name: Mx.lcmdInputScale

Units: Unitless multiplier

HiDS Location: LoopInputs page

Description: Used to scale the external analog input.

10.20 Manual Feedback

For a description of the use of Manual Feedback, refer to the Theory of Operations section within this document.

Variable name: ManualFeedbackConfigX.RampRpmPerSecond

Units: RPM per second, unbounded

HiDS Location: ManualFeedback page

Description: Used in conjunction with ManualFeedbackConfigX.RPM and ManualFeedbackConfigX.RampRpmStop, the RampRpmPerSecond configures the RPM acceleration from ManualFeedbackConfigX.RPM to ManualFeedbackConfigX.RampRpmStop. Note this value is auto-set to MxVL.AccelRPMPerSec if AutoSetManualFeedbackParameters is set to true (1). This allows the standard Run Panel to be used in Manual Feedback.

Variable name: ManualFeedbackConfigX.RampRpmStop

Units: RPM, unbounded

HiDS Location: ManualFeedback page

Description: Used in conjunction with ManualFeedbackConfigX. RampRpmPerSecond and ManualFeedbackConfigX.RampRpmStop. The RampRpmStop configures end velocity to accelerate to. Note this value is auto-set to MxVL.RPMUserCommand if AutoSetManualFeedbackParameters is set to true (1). This allows the standard Run Panel to be used in Manual Feedback.

Variable name: ManualFeedbackConfigX.AutoRampEnable

Units:	Boolean, 0 or 1
HiDS Location:	ManualFeedback page
Description:	Automatically enables or disables the acceleration from ManualFeedbackConfigX.RPM to ManualFeedbackConfigX.RampRpmStop.
Variable name:	ManualFeedbackConfigX.RampEnable
Units:	Boolean, 0 or 1
HiDS Location:	ManualFeedback page
Description:	Enables or disables the acceleration from ManualFeedbackConfigX.RPM to ManualFeedbackConfigX.RampRpmStop.
Variable name:	ManualFeedbackConfigX.RPM
Units:	RPM
HiDS Location:	ManualFeedback page
Description:	Displays the manual-feedback generated RPM
Variable name:	ManualFeedbackX.Degrees
Units:	Degrees, 0-360
HiDS Location:	ManualFeedback page
Description:	Shows the motor position in degrees, normalized to 0-360°
Variable name:	ManualFeedbackX.Radians
Units:	Radians, valid range is from 0 to 2π
HiDS Location:	ManualFeedback page
Description:	Shows the motor position in radians, normalized to 0 to 2π .
Variable name:	ManualFeedbackX.RadiansSummed
Units:	Radians, unbounded
HiDS Location:	ManualFeedback page
Description:	Unless explicitly cleared, this variable shows the total accumulated motor position in radians, since power on. Note the sign of the total motor movements is considered in the summing. For example, from 0, a command of +10 radians, plus a command of +5 radians results in RadiansSummed = 5 (because the second movement was -5 radians).
Variable name:	ManualFeedbackX.RadiansPerSecond
Units:	Radians per seconds, unbounded
HiDS Location:	ManualFeedback page
Description:	Shows the unfiltered motor velocity in radians per second
Variable name:	ManualFeedbackX.FilteredRadPerSec
Units:	Radians per seconds, unbounded
HiDS Location:	ManualFeedback page
Description:	Shows the filtered motor velocity in radians per second
Variable name:	ManualFeedbackX.Rpm
Units:	Radians per minute
HiDS Location:	ManualFeedback page
Description:	Shows the unfiltered motor velocity in revolutions per minute.

Variable name: ManualFeedbackX.FilteredRpm
Units: Radians per minute
HiDS Location: ManualFeedback page
Description: Shows the filtered motor velocity in revolutions per minute.

10.21 Motor AHSL, MotorBHSL

Variable name: MxIqInputFilter-Type
Units: Integer, 0-10
HiDS Location: MotorXHSL pages
Description: Configures the filter type. The following filter types are available:
0=Filter Disabled
1=Low Pass
2=High Pass
3=Band Pass
4=Notch
5=Low Shelf
6=High Shelf
7=Gain
8=Integral
9=Low Pass 1st Order
10=High Pass 1st Order

Variable name: MxIqInputFilter-GaindB
Units: dB
HiDS Location: MotorXHSL pages
Description: Filter gain in dB.

Variable name: MxIqInputFilter-CenterOrCorner
Units: Hz
HiDS Location: MotorXHSL pages
Description: The filters' center frequency

Variable name: MxIqInputFilter-Q
Units: Unitless
HiDS Location: MotorXHSL pages
Description: Dimensionless parameter that describes how under-damped an oscillator or resonator is,[1] and characterizes a resonator's bandwidth relative to its center frequency

Variable name: MxIqInputFilter-Max
Units: Hz
HiDS Location: MotorXHSL pages
Description: Maximum cut-off frequency

Variable name: MxIqInputFilter-SampleFreq
Units: Hz
HiDS Location: MotorXHSL pages
Description: Filters sampling frequency

Variable name: MxlqInputFilter-Update
Units: Boolean, 0 or 1
HiDS Location: MotorXHSL pages
Description: In filter configuration is modified the changes will not take effect until an Update (1) is performed.

Variable name: Mx.lqCmdFiltered
Units: Amps
HiDS Location: MotorXHSL pages
Description: If MxlqInputFilter is enabled, this is the filtered lqUserCommand

Variable name: Mx.lqUserCommand
Units: Amps
HiDS Location: MotorXHSL pages
Description: One of the input selections for the current loop input command multiplexers; if selected as the input source this will be the command into the current loop. This command is usually input via the HiDS RunPanel

Variable name: Mx.lqCANCommand
Units: Amps
HiDS Location: MotorXHSL pages
Description: One of the input selections for the current loop input command multiplexers; if selected as the input source this will be the command into the current loop. This command is received via an ESI CAN protocol.

Variable name: Mx.lqSerialCommand
Units: Amps
HiDS Location: MotorXHSL pages
Description: One of the input selections for the current loop input command multiplexers; if selected as the input source this will be the command into the current loop. This command is received via an ESI Serial protocol.

Variable name: Mx.DesiredCurrent
Units: Amps
HiDS Location: MotorXHSL pages
Description: When running velocity mode, the velocity loop output produces Mx.DesiredCurrent as an input to the current loop. This value is automatically summed in with the command input sources selected from the two multiplexers.

Variable name: Mx.lcmdAnalogInput
Units: Amps
HiDS Location: MotorXHSL pages
Description: One of the input selections for the current loop input command multiplexers; if selected as the input source this will be the command into the current loop. This command is received from an external analog voltage .

Variable name: Mx.IdUserCommand
Units: Volts
HiDS Location: MotorXHSL pages

Description: The (SinTest.output * SinTest.id_amplitude) result is summed with Mx.IdUserCommand to produce Mx.IdCmd

Variable name: Mx.Ia
Units: Amps
HiDS Location: MotorXHSL pages
Description: Measured motor phase-A current

Variable name: Mx.Ib
Units: Amps
HiDS Location: MotorXHSL pages
Description: Measured motor phase-B current

Variable name: Mx.Ic
Units: Amps
HiDS Location: MotorXHSL pages
Description: Measured motor phase-C current

Variable name: Mx.Ialpha
Units: Amps
HiDS Location: MotorXHSL pages
Description: Output of the Clark transform; using not used for debug/analysis.

Variable name: Mx.Ibeta
Units: Amps
HiDS Location: MotorXHSL pages
Description: Output of the Clark transform; using not used for debug/analysis.

Variable name: Mx.AngleSin
Units: Radians
HiDS Location: MotorXHSL pages
Description: The SIN component of the electrical angle

Variable name: Mx.AngleCos
Units: Amps
HiDS Location: MotorXHSL pages
Description: The COS component of the electrical angle

Variable name: Mx.ThirdHarmonic
Units: Volts
HiDS Location: MotorXHSL pages
Description: The Third Harmonic represents the minor voltage-adjustment necessary to center-align the 3 phase-voltages that should be 120 degrees out of phase.

Variable name: Mx.ElectricalAngle
Units: Degrees in Radians
HiDS Location: MotorXHSL pages
Description: Degree or the cycle of EMF induced in a single conductor

Variable name: Mx.Id

Units:	Amps
HiDS Location:	MotorXHSL pages
Description:	The output from the Park transform is the torque vector Mx.Iq and the flux vector Mx.Id, which are the measured/calculated vectors in the rotating frame to be used in compensating for error from the desired Mx.IqCmd and Mx.IdCmd
Variable name:	Mx.Iq
Units:	Amps
HiDS Location:	MotorXHSL pages
Description:	The output from the Park transform is the torque vector Mx.Iq and the flux vector Mx.Id, which are the measured/calculated vectors in the rotating frame to be used in compensating for error from the desired Mx.IqCmd and Mx.IdCmd
Variable name:	Mx.IqFiltered
Units:	Amps
HiDS Location:	MotorXHSL pages
Description:	The output of a low pass first order filter with the Mx.Iq vector as input
Variable name:	Mx.IqFilterCoeff
Units:	Float
HiDS Location:	MotorXHSL pages
Description:	First order coefficient for Mx.Iq filter
Variable name:	Mx.IqCmd
Units:	Amps
HiDS Location:	MotorXHSL pages
Description:	Mx.IqCmd is the torque input into the torque loop. It is the sum of the output of the velocity loop (Mx.DesiredCurrent) and the output of the loop command multiplexers (Mx.IqCmdFiltered).
Variable name:	Mx.LastIqCmd
Units:	Amps
HiDS Location:	MotorXHSL pages
Description:	Previous Mx.IqCmd
Variable name:	Mx.IdCmd
Units:	Amps
HiDS Location:	MotorXHSL pages
Description:	Mx.IqCmd is the flux input into the torque loop. It is the sum of Mx.IdUserCommand and SinTest.output; note that this value is typically zero.
Variable name:	Mx.LastIdCmd
Units:	Amps
HiDS Location:	MotorXHSL pages
Description:	Previous Mx.IdCmd
Variable name:	Mx.IqError
Units:	Amps
HiDS Location:	MotorXHSL pages
Description:	The error between the desired IQ command and the resultant IQ, and it is the goal to reduce this

IQ error to zero

Variable name: Mx.IdError
Units: Amps
HiDS Location: MotorXHSL pages
Description: The error between the desired ID command and the resultant ID, and it is the goal to reduce this ID error to zero

Variable name: Mx.IqPIDOutput
Units: Volts
HiDS Location: MotorXHSL pages
Description: This is the output of the IQ PI-compensator within the current-loop.

Variable name: Mx.IdPIDOutput
Units: Volts
HiDS Location: MotorXHSL pages
Description: This is the output of the ID PI-compensator within the current-loop.

Variable name: Mx.UdFeedForward
Units: Volts
HiDS Location: MotorXHSL pages
Description: Feed forward error compensation to help counter the effects of reactive losses (Mx.UdFeedForward) with increasing frequency. Of course this feed-forward error compensation relies on reasonably accurate configurations of the motor phase inductance MotorX.Inductance. Typically this feed-forward is enabled (1), but it can be disabled by setting Mx.UdFeedForwardEnable=0

Variable name: Mx.UdFeedForwardEnable
Units: Boolean, 0 or 1
HiDS Location: MotorXHSL pages
Description: Enables feed forward for reactive losses.

Variable name: Mx.Ud
Units: Volts
HiDS Location: MotorXHSL pages
Description: The error compensation of Id results in this intermediate voltage and input into the Inverse Park Transform

Variable name: Mx.IRFeedForward
Units: Volts
HiDS Location: MotorXHSL pages
Description: Feed-forward error compensation to help counter the effects of motor-back-EMF-voltage (Mx.EMFFeedForward) and the effects of IR increases (Mx.IRFeedForward). Of course these feed-forward error compensations rely on reasonably accurate configurations of the motor voltage constant MotorX.Ke and the motor phase resistance MotorX.Ohms. Typically this feed-forward is enabled (1), but it can be disabled by setting Mx.UqFeedForwardEnable = 0 (false).

Variable name: Mx.EMFFeedForward
Units: Volts

HiDS Location:	MotorXHSL pages
Description:	Feed-forward error compensation to help counter the effects of motor-back-EMF-voltage (Mx.EMFFeedForward) and the effects of IR increases (Mx.IRFeedForward). Of course these feed-forward error compensations rely on reasonably accurate configurations of the motor voltage constant MotorX.Ke and the motor phase resistance MotorX.Ohms. Typically this feed-forward is enabled (1), but it can be disabled by setting Mx.UqFeedForwardEnable = 0 (false).
Variable name:	Mx.UqFeedForward
Units:	Volts
HiDS Location:	MotorXHSL pages
Description:	Feed-forward error compensation to help counter the effects of motor-back-EMF-voltage (Mx.EMFFeedForward) and the effects of IR increases (Mx.IRFeedForward). Of course these feed-forward error compensations rely on reasonably accurate configurations of the motor voltage constant MotorX.Ke and the motor phase resistance MotorX.Ohms. Typically this feed-forward is enabled (1), but it can be disabled by setting Mx.UqFeedForwardEnable = 0 (false).
Variable name:	Mx.UqFeedForwardEnable
Units:	Boolean, 0 or 1
HiDS Location:	MotorXHSL pages
Description:	Enables (1) the EMF feed-forward error compensator.
Variable name:	Mx.Uq
Units:	Volts
HiDS Location:	MotorXHSL pages
Description:	The error compensation of Iq results in this intermediate voltage and input into the Inverse Park Transform
Variable name:	Mx.Ualpha
Units:	Volts
HiDS Location:	MotorXHSL pages
Description:	Ualpha and Ubeta are outputs from the Inverse-Park transform within the current-loop.
Variable name:	Mx.Ubeta
Units:	Volts
HiDS Location:	MotorXHSL pages
Description:	Ualpha and Ubeta are outputs from the Inverse-Park transform within the current-loop.
Variable name:	Mx.VoltageMagnitude
Units:	Volts L-N
HiDS Location:	MotorXHSL pages
Description:	Indicates the voltage vector ($\sqrt{Ualpha^2 + Ubeta^2}$) required by the motor at that moment in time.
Variable name:	Mx.VBusUtilization
Units:	Percent, value should be between 0 and 1
HiDS Location:	MotorXHSL pages
Description:	Can indicate what percent of the high-voltage is used in motor-control. Calculated as VoltageMagnitude / MaxCommandVoltage, and >99% indicates current-loop saturation (out of voltage).

Variable name:	Mx.PreSaturated
Units:	Boolean, 0 or 1
HiDS Location:	MotorXHSL pages
Description:	If 1, then (VoltageMagnitude / MaxCommandVoltage) is greater than 0.99, which indicates the current-loop is saturated (out of voltage).
Variable name:	Mx.Saturated
Units:	Boolean, 0 or 1
HiDS Location:	MotorXHSL pages
Description:	If 1, then (VoltageMagnitude / MaxCommandVoltage) is greater than 1, which indicates the current-loop is saturated (out of voltage).
Variable name:	Mx.IntegralHoldEnable
Units:	Boolean, 0 or 1
HiDS Location:	MotorXHSL pages
Description:	If 1, then (VoltageMagnitude / MaxCommandVoltage) is greater than 1, which indicates the current-loop is saturated (out of voltage). This will freeze the loop integral until the error changes sign (until less voltage is demanded).
Variable name:	Mx.IntegralHoldActive
Units:	Boolean, 0 or 1
HiDS Location:	MotorXHSL pages
Description:	This variable indicates the loop integral is being held constant; see IntegralHoldEnable.
Variable name:	Mx.Va
Units:	Volts
HiDS Location:	MotorXHSL pages
Description:	Voltage measured at phase-A
Variable name:	Mx.Vb
Units:	Volts
HiDS Location:	MotorXHSL pages
Description:	Voltage measured at Phase-B
Variable name:	Mx.Vc
Units:	Volts
HiDS Location:	MotorXHSL pages
Description:	Voltage measured at Phase-C
Variable name:	Mx.IqRampLimitPer_ms
Units:	Amps per millisecond
HiDS Location:	MotorXHSL pages
Description:	Sets the input slew-rate limiter for IqCmd.
Variable name:	Mx.IdRampLimitPer_ms
Units:	Amps per millisecond
HiDS Location:	MotorXHSL pages
Description:	Sets the input slew-rate limiter for IdCmd.

10.22 MotorParameters

Most of the relevant motor parameters can also be configured in the [Motor] tab.

Variable name:	MotorX.Inductance
Units:	Henries, positive only
HiDS Location:	MotorParameters page
Description:	This inductance is the inductance per phase, so if measured / supplied motor inductance is measured line-to-line, then the internal value is ½ the line-to-line value.
Variable name:	MotorX.Ohms
Units:	Ohms, positive only
HiDS Location:	MotorParameters page
Description:	The units of resistance shown on the [Motor] tab are Line to Neutral Ohms. This resistance is the resistance per phase, so if measured / supplied motor resistance is measured line-to-line, then the internal value is ½ the line-to-line value.
Variable name:	MotorX.Ke
Units:	VoltsPeak L-N per RPM, positive only
HiDS Location:	MotorParameters page
Description:	Also known as the voltage constant or back-EMF constant, the units of the Motor voltage constant, Ke, are Line-to-Neutral Volts-PEAK/RPM. The conversion from Line-to-Line: L-N V/RPM = L-L V/KRPM / (1000 * √3)
Variable name:	MotorX.PolePairs
Units:	Integer, positive only.
HiDS Location:	MotorParameters page
Description:	Pole pairs are the number of sets of three-way electromagnetic windings that a motor has. Note: this is the pole PAIRS (the actual number of motor poles divided by 2)
Variable name:	MotorX.FeedbackType
Units:	Integer, 0-7
HiDS Location:	MotorParameters page
Description:	This variable would normally be changed via the [Motor] tab feedback dropdown list. Selects the active feedback device for all controls loops, except the position loop if MotorX.SecondaryFeedback is enabled (1). The following feedback options are available: 0=Resolver 1=Quadrature Encoder 2=Manual 3=Sensorless 4=Hall 5=Serial Encoder 6=Sin/Cos Encoder 7=EnDat Encoder
Variable name:	MotorX.FeedbackPhase
Units:	Degrees, -360° to 360°
HiDS Location:	MotorParameters page
Description:	For proper vector control of a motor, the resolver shaft must be aligned with the motor shaft

(usually in manufacturing), or else a phase offset must be entered here. The FeedbackPhase is the electrical angle offset.

Variable name: MotorX.FeedbackDirection
Units: Signed multiplier, the valid values are -1 or 1
HiDS Location: MotorParameters page
Description: Should a motor phase wire be swapped or a resolver sine/cosine swap occur, this variable can compensate; the valid values are 1 (normal) or -1 (reversed).

Variable name: MotorX.ResolverSpeed
Units: Integer, positive values only
HiDS Location: MotorParameters page
Description: If the resolver indicates multiple rotations of the motor, during a single actual motor rotation, this value can be set to that multiple.

10.23 SecondaryFB

Variable name: MotorX.SecondaryFeedbackType
Units: Integer, 0-7
HiDS Location: SecondaryFB page
Description: This variable would normally be changed via the [Motor] tab secondary feedback dropdown list. Selects the active feedback device for the position loop of MotorX, the following options are available:
0=Resolver
1=Quadrature Encoder
2=Manual
3=Sensorless
4=Hall
5=Serial Encoder
6=Sin/Cos Encoder
7=EnDat Encoder

Variable name: MotorX.SecondaryFeedbackEnabled
Units: Boolean, 0 or 1
HiDS Location: SecondaryFB page
Description: Indicates whether the secondary/position-loop feedback is enabled for this motor.

Variable name: MotorX.SecondaryFeedbackRadians
Units: Radians
HiDS Location: SecondaryFB page
Description: The latest position, in radians, for the secondary feedback device

Variable name: MotorX.SecondaryFeedbackRadiansSummed
Units: Radians
HiDS Location: SecondaryFB page
Description: This variable shows the total accumulated motor position in radians, since power on.

10.24Position

Variable name:	MxPLInputFilter-Type
Units:	Integer, 0-10
HiDS Location:	Position page
Description:	Configures the filter type. The following filter types are available: 0=Filter Disabled 1=Low Pass 2=High Pass 3=Band Pass 4=Notch 5=Low Shelf 6=High Shelf 7=Gain 8=Integral 9=Low Pass 1st Order 10=High Pass 1st Order
Variable name:	MxPLInputFilter-GaindB
Units:	dB
HiDS Location:	Position page
Description:	Filter gain in dB.
Variable name:	MxPLInputFilter-CenterOrCorner
Units:	Hz
HiDS Location:	Position page
Description:	The filters' center frequency
Variable name:	MxPLInputFilter-Q
Units:	Unitless
HiDS Location:	Position page
Description:	Dimensionless parameter that describes how under-damped an oscillator or resonator is,[1] and characterizes a resonator's bandwidth relative to its center frequency
Variable name:	MxPLInputFilter-Max
Units:	Hz
HiDS Location:	Position page
Description:	Maximum cut-off frequency
Variable name:	MxPLInputFilter-SampleFreq
Units:	Hz
HiDS Location:	Position page
Description:	Filters sampling frequency
Variable name:	MxPLInputFilter-Update
Units:	Boolean, 0 or 1
HiDS Location:	Position page
Description:	In filter configuration is modified the changes will not take effect until an Update (1) is performed.

Variable name:	MxPL.PosCmdFiltered
Units:	Radians
HiDS Location:	Position page
Description:	Filtered radians command, filter used is MxPLInputFilter
Variable name:	MotorXPositionEnable
Units:	Boolean, 0 or 1
HiDS Location:	Position page
Description:	Enables the position loop for Motor-A. This variable would normally be changed via the [Loop Gains] tab Control Mode dropdown list.
Variable name:	MxPL.MinRadiansCommand
Units:	Radians
HiDS Location:	Position page
Description:	MxPL.MinRadiansCommand limits the lower value of the input to the position loop.
Variable name:	MxPL.MaxRadiansCommand
Units:	Radians
HiDS Location:	Position page
Description:	MxPL.MaxRadiansCommand limits the upper value of the input to the position loop.
Variable name:	MxPL.AbsolutePosition
Units:	Radians
HiDS Location:	Position page
Description:	The absolute position (in radians) reported back from the feedback device. For resolver and encoder feedbacks, the MxPL.AbsolutePosition equals the MxPL.GearRatio times the ResolverXRadiansSummed or EncoderX.RadiansSummed respectively.
Variable name:	MxPL.SetPositionToZero
Units:	Boolean, 0 or 1
HiDS Location:	Position page
Description:	Resets the current feedback RadiansSummed value to zero, which sets the position-loop active/measured position to zero.
Variable name:	MxPL.RadiansUserCommand
Units:	Radians
HiDS Location:	Position page
Description:	One of the command input selections for the position loop input command multiplexers; if selected as the input source this will be the command into the current loop. This command is usually input via the HiDS RunPanel.
Variable name:	MxPL.RadiansCANCommand
Units:	Radians
HiDS Location:	Position page
Description:	One of the command input selections for the position loop input command multiplexers; if selected as the input source this will be the command into the current loop. This command is usually received via the CAN bus using ESI CAN protocol

Variable name:	MxPL.RadiansSerialCommand
Units:	Radians
HiDS Location:	Position page
Description:	One of the command input selections for the position loop input command multiplexers; if selected as the input source this will be the command into the current loop. This command is usually received via the serial bus using ESI Serial protocol
Variable name:	MxPL.RadiansCommand
Units:	Radians
HiDS Location:	Position page
Description:	The final position input into the position loop; it is the sum of the output of two position loop input multiplexers.
Variable name:	MxPL.AccelLimitedRadiansCommand
Units:	Radians, unbounded
HiDS Location:	Position page
Description:	For a step-response input to MxPL.RadiansUserCommand, the MxPL.AccelLimitedRadiansCommand variable is the resulting positional action is limited by positional acceleration value setting variable MxPL.AccelRadiansPerSec.
Variable name:	MxPL.DesiredPositionChange
Units:	Radians
HiDS Location:	Position page
Description:	The difference, in radians, between the commanded position and the absolute position
Variable name:	MxPL.DesiredRPM
Units:	Revolutions per Minute
HiDS Location:	Position page
Description:	The output of the position loop, and one of the inputs summed into the velocity loop
Variable name:	MxPL.PositionDirection
Units:	Signed multiplier, the valid values are -1 or 1.
HiDS Location:	Position page
Description:	MxPL.PositionDirection can swap the motor's rotational direction when operated in the position mode. Note when set to -1, this variable only swaps the sign of the incoming position command and the reported position; the raw feedback speeds (i.e. ResolverX.Rpm) are unchanged by this variable.
Variable name:	MxPL.AccelRadiansPerSec
Units:	Radians per Second
HiDS Location:	Position page
Description:	For a step-response input to MxPL.RadiansUserCommand, the MxPL.AccelRadiansPerSec positional acceleration variable limits the resulting positional action, which is indicated by MxPL.AccelLimitedRadiansCommand.
Variable name:	MxPL.IntegralHoldEnable
Units:	Boolean, 0 or 1
HiDS Location:	Position page
Description:	If 1, then loop-output is greater than the maximum RPM setting, which indicates the loop is

saturated. This will freeze the loop integral until the error changes sign (until less RPM is demanded).

Variable name: MxPL.IntegralHoldActive
Units: Boolean, 0 or 1
HiDS Location: Position page
Description: This variable indicates the loop integral is being held constant; see IntegralHoldEnable.

Variable name: MxPL.PcmdInputScale
Units: Radians per volt
HiDS Location: Position page
Description: MxPL.PcmdInputScale sets the scale used to multiply the MxPL.PcmdInput to produce a position input to the position loop based on the analog voltage input provided. When Mx.LcmdInput is unused, this value is set to zero. This value is summed with MxPL.RadiansUserCommand to produce MxPL.RadiansCommand.

Variable name: MxPL.PcmdInput
Units: Radians
HiDS Location: Position page
Description: MxPL.PcmdInputScale sets the scale used to multiply the MxPL.PcmdInput to produce a position input to the position loop based on the analog voltage input provided. When Mx.LcmdInput is unused, this value is set to zero. This value is summed with MxPL.RadiansUserCommand to produce MxPL.RadiansCommand.

Variable name: MxPL.RadiansError
Units: Radians
HiDS Location: Position page
Description: MxPL.RadiansError indicates the positional error. $MxPL.RadiansError = MxPL.AccelLimitedRadiansCommand - MxPL.AbsolutePosition$.

Variable name: MxPL.GearRatio
Units: Integer, positive only
HiDS Location: Position page
Description: Specifies the gear ratio of the motor system, if one exists. The units of GearRatio is motor-revolutions / post-gearbox revolutions.

10.25 Power

The Controller captures real-time phase voltage and current measurements in order to estimate the real, reactive, and apparent components of the power applied.

Variable name: MxPower.PhaseAIRms
Units: Amps-RMS, bounded only by the current-limiter of the controller
HiDS Location: Power page
Description: The Phase A current

Variable name: MxPower.PhaseAVRms
Units: Volts-RMS, bounded only by the voltage limiter of the controller
HiDS Location: Power page

Description:	The Phase A voltage
Variable name:	MxPower.PhaseAReal
Units:	Watts, bounded by MxVa and Mx.Ia
HiDS Location:	Power page
Description:	The Phase A real-component of power is the average of the product of Mx.Va * Mx.Ia.
Variable name:	MxPower.PhaseAApparent
Units:	Watts, bounded by MxPower.PhaseAIRMS
HiDS Location:	Power page
Description:	The Phase A apparent power is the product of MxPower.PhaseAIRms * MxPower.PhaseAVRms in Watts.
Variable name:	MxPower.PhaseAReactive
Units:	Watts, bounded by MxPower.PhaseAApparent and MxPower.PhaseAReal.
HiDS Location:	Power page
Description:	The Phase A reactive power is calculated indirectly as the square-root of (MxPower.PhaseAApparent ² - MxPower.PhaseAReal ²).
Variable name:	MxPower.Real
Units:	Watts, bounded by MxPower.Real
HiDS Location:	Power page
Description:	It is assumed each phase-power is equal, so $MxPower.Real = 3 * MxPower.PhaseAReal$.
Variable name:	MxPower.Apparent
Units:	Watts, bounded by MxPower.Apparent
HiDS Location:	Power page
Description:	It is assumed each phase-power is equal, so $MxPower.Apparent = 3 * MxPower.PhaseAApparent$.
Variable name:	MxPower.Reactive
Units:	Watts, bounded by MxPower.Reactive
HiDS Location:	Power page
Description:	It is assumed each phase-power is equal, so $MxPower.Reactive = 3 * MxPower.PhaseAReactive$.
Variable name:	MxPower.PFDegrees
Units:	Degrees, 0-360°
HiDS Location:	Power page
Description:	$MxPower.PFDegrees = \arccos(MxPower.RF)$ in degrees.
Variable name:	MxPower.PF
Units:	Unit-less, bounded by MxPower.Real and MxPower.Apparent.
HiDS Location:	Power page
Description:	$MxPower.PF = MxPower.Real / MxPower.Apparent$.

10.26 Regen

Not all Controllers support the Regen feature. Refer to your Controller's specifications. If supported, the Regen feature will usually be enabled by default, and the Regen voltage threshold can be set with the HiDS Limits Tab.

Variable name:	Regen.Enable
Units:	Boolean, 0 or 1
HiDS Location:	Regen page
Description:	Enables or disables the Regen feature.
Variable name:	Regen.State
Units:	Boolean, 0 or 1
HiDS Location:	Regen page
Description:	Indicates whether the regen switch is active (1) or not (0)
Variable name:	Regen.OnThreshold
Units:	Voltage, bound by voltage limitation of the controller
HiDS Location:	Regen page
Description:	Sets the voltage limit, which when exceeded, the Regen switch is enabled, and energy will be dissipated by the Regen resistor.
Variable name:	Regen.MaxOn_ms
Units:	Time in milliseconds
HiDS Location:	Regen page
Description:	In order to protect the Regen resistor, limit the Regen on (and off) time to 'MaxOn_ms' milliseconds.

10.27 Resolver

Variable name:	ResolverX.Degrees
Units:	Degrees, 0 to 360°
HiDS Location:	Resolver page
Description:	Shows the motor position in degrees, normalized to 0 to 360°.
Variable name:	ResolverX.Radians
Units:	Radians, 0 to 2 π .
HiDS Location:	Resolver page
Description:	Shows the motor position in radians, normalized to 0 to 2 π .
Variable name:	ResolverX.RadiansSummed
Units:	Radians, unbounded
HiDS Location:	Resolver page
Description:	Unless explicitly cleared, this variable shows the total accumulated motor position in radians, since power on. Note the sign of the total motor movements is considered in the summing. For example, from 0, a command of +10 radians, plus a command of +5 radians results in RadiansSummed = 5 (because the second movement was -5 radians).
Variable name:	ResolverX.RadiansPerSecond
Units:	Radians per Second
HiDS Location:	Resolver page
Description:	Shows the unfiltered motor velocity in Radians per Second
Variable name:	ResolverX.FilteredRadPerSec

Units:	Radians per Second
HiDS Location:	Resolver page
Description:	Shows the filtered motor velocity in Radians per second
Variable name:	ResolverX.Rpm
Units:	Revolutions per minute
HiDS Location:	Resolver page
Description:	Shows the unfiltered motor velocity in revolutions per minute.
Variable name:	ResolverX.FilteredRpm
Units:	Revolutions per minute
HiDS Location:	Resolver page
Description:	Shows the filtered motor velocity in revolutions per minute.
Variable name:	ResolverX.Timestamp
Units:	Time in microseconds
HiDS Location:	Resolver page
Description:	Shows the time of the last measured rotor position.
Variable name:	ResolverX.HSPRadians
Units:	Radians
HiDS Location:	Resolver page
Description:	Shows the filtered motor velocity in Radians per second
Variable name:	ResolverAFeedbackRms
Units:	Volts
HiDS Location:	Resolver page
Description:	The square-root of the sine(squared) and cosine(squared) inputs. Above the threshold indicates a good resolver input signal.
Variable name:	ResolverASinRms
Units:	Volts
HiDS Location:	Resolver page
Description:	The RMS of the resolver sine input.
Variable name:	ResolverACosRms
Units:	Volts
HiDS Location:	Resolver page
Description:	The RMS of the resolver cosine input.
Variable name:	ResolverATrackingObs.Enable
Units:	Boolean, 0 or 1
HiDS Location:	Resolver page
Description:	Enables the angle tracking observer which calculates both the shaft angle and speed.
Variable name:	ResolverATrackingObs.NaturalFreqHz
Units:	Hz
HiDS Location:	Resolver page
Description:	The natural frequency of the tracking observer.

Variable name:	ResolverATrackingObs.DampingFactor
Units:	Float, 0-1
HiDS Location:	Resolver page
Description:	The tracking observer damping factor.
Variable name:	ResolverATrackingObs.AngleIn
Units:	Radians
HiDS Location:	Resolver page
Description:	Raw resolver shaft angle value before tracking observer (input into tracking observer algorithm).
Variable name:	ResolverATrackingObs.TrackingK1
Units:	Unitless
HiDS Location:	Resolver page
Description:	The angle tracking observer is a second order transfer function, where TrackingK1 and TrackingK2 are the two coefficients.
Variable name:	ResolverATrackingObs.TrackingK2
Units:	Unitless
HiDS Location:	Resolver page
Description:	The angle tracking observer is a second order transfer function, where TrackingK1 and TrackingK2 are the two coefficients.
Variable name:	ResolverATrackingObs.AngleOut
Units:	Radians
HiDS Location:	Resolver page
Description:	The angular output of the tracking observer, corresponding to the shaft angle.
Variable name:	ResolverX.FeedbackRms
Units:	Voltage, RMS, typically between 0 and 2.5V
HiDS Location:	Resolver page
Description:	The ResolverXFeedbackRms is the square-root of the Resolver sine voltage squared plus the Resolver cosine voltage squared. This value is useful to determine if the resolver inputs are all properly connected.
Variable name:	ResolverX.SinRms
Units:	Voltage, RMS, typically between 0 and 1V.
HiDS Location:	Resolver page
Description:	ResolverXSinRms is raw resolver sine RMS voltage.
Variable name:	ResolverX.CosRms
Units:	Voltage, RMS, typically between 0 and 1V.
HiDS Location:	Resolver page
Description:	ResolverXCosRms is raw resolver cosine RMS voltage.
Variable name:	ResolverX.SinCosEncoderDcOffset
Units:	Volts
HiDS Location:	Resolver page
Description:	DC Offset subtracted from raw measurement.

10.28 Sensorless

Variable name: SensorlessX.Degrees
Units: Degrees, 0 to 360°
HiDS Location: SensorlessA and SensorlessB pages
Description: Shows the motor position in degrees, normalized to 0 to 360°.

Variable name: SensorlessX.Radians
Units: Radians, 0 to 2π .
HiDS Location: SensorlessA and SensorlessB pages
Description: Shows the motor position in radians, normalized to 0 to 2π .

Variable name: SensorlessX.RadiansPerSecond
Units: Radians per second
HiDS Location: SensorlessA and SensorlessB pages
Description: Shows the unfiltered motor velocity in Radians per Second

Variable name: SensorlessX.FilteredRadPerSec
Units: Radians per second
HiDS Location: SensorlessA and SensorlessB pages
Description: Shows the filtered motor velocity in Radians per second

Variable name: SensorlessX.RPM
Units: Revolutions per minute
HiDS Location: SensorlessA and SensorlessB pages
Description: Shows the unfiltered motor velocity in revolutions per minute

Variable name: SensorlessX.FilteredRPM
Units: Revolutions per minute
HiDS Location: SensorlessA and SensorlessB pages
Description: Shows the filtered motor velocity in revolutions per minute.

Variable name: AutoSetSensorlessXParameters
Units: Boolean, 0 or 1
HiDS Location: SensorlessA and SensorlessB pages
Description: Automatically sets the sensorless tuning parameters based on the configured Min/MaxCurrentCommand and Overspeed.Limit.

Variable name: MxSmo.StartupCurrent
Units: Amps
HiDS Location: SensorlessA and SensorlessB pages
Description: Similar to starting a motor in manual feedback, this current is often 25% to 50% of the motor's maximum rated continuous current.

Variable name: MxSmo.MinOpenLoopIq
Units: Amps
HiDS Location: SensorlessA and SensorlessB pages
Description: When the back-EMF voltage is large enough to run sensorless in closed-loop, this variable sets

the typical `IqCmd` required to run at the `MxSmo.ClosedLoopRPMThreshold` RPM.

Variable name:	<code>MxSmo.OpenLoopRPMThreshold</code>
Units:	RPM
HiDS Location:	SensorlessA and SensorlessB pages
Description:	The <code>MxSmo.ClosedLoopRPMThreshold</code> variable sets the actual closed-loop RPM value. This variable is typically set to ~80% of <code>MxSmo.ClosedLoopRPMThreshold</code> , and the “blended region” is used to linearly scale the <code>IqCmd</code> from <code>MxSmo.StartupCurrent</code> to <code>MxSmo.MinOpenLoopIq</code> .
Variable name:	<code>MxSmo.ClosedLoopRPMThreshold</code>
Units:	RPM
HiDS Location:	SensorlessA and SensorlessB pages
Description:	The <code>MxSmo.ClosedLoopRPMThreshold</code> variable sets the actual closed-loop RPM value. This RPM value must generate sufficient back-EMF voltage to allow the sensorless algorithm to properly estimate the motor’s electrical angle.
Variable name:	<code>MxSmo.OpenLoopRPMPerSec</code>
Units:	RPM-per-second
HiDS Location:	SensorlessA and SensorlessB pages
Description:	This variable sets the velocity acceleration in the open-loop region, which depends on the motor’s inertia.
Variable name:	<code>MxSmo.RPMToSwitchToMaxAccel</code>
Units:	RPM
HiDS Location:	SensorlessA and SensorlessB pages
Description:	Typically, this value is set to the same as <code>MxSmo.ClosedLoopRPMThreshold</code> ; however sometimes a higher speed is required before sufficient back-EMF exists to increase acceleration.
Variable name:	<code>MxSmo.AccelRPMPerSec</code>
Units:	RPM-per-second
HiDS Location:	SensorlessA and SensorlessB pages
Description:	This variable sets the velocity acceleration in the closed-loop region, which depends on the motor’s inertia.
Variable name:	<code>MxSmo.StartIqRampPer_ms</code>
Units:	Amps-per-millisecond
HiDS Location:	SensorlessA and SensorlessB pages
Description:	This variable is calculated as follow: $\text{MxSmo.StartIqRampPer_ms} = \text{MxSmo.StartupCurrent} / \text{MxSmo.StartIqRampDelay_ms}$. This delay allows motors with large inertia to stabilize before starting the open-loop operation.
Variable name:	<code>MxSmo.StartIqRampDelay_ms</code>
Units:	Milliseconds
HiDS Location:	SensorlessA and SensorlessB pages
Description:	This variable and <code>StartupCurrent</code> set the <code>StartIqRampPer_ms</code> as follows: $\text{MxSmo.StartIqRampPer_ms} = \text{MxSmo.StartupCurrent} / \text{MxSmo.StartIqRampDelay_ms}$. This delay allows motors with large inertia to stabilize before starting the open-loop operation.
Variable name:	<code>MxSmo.AngleIn</code>

Units:	Radians 0-2PI
HiDS Location:	SensorlessA and SensorlessB pages
Description:	This is the post-low-pass-filtered estimated rotor-angle including the PhaseCorrectionValue.
Variable name:	MxSmo.AngleOut
Units:	Radians 0-2PI
HiDS Location:	SensorlessA and SensorlessB pages
Description:	This is the post-tarcking-observer/final-estimated rotor-angle.
Variable name:	MxSmo.StartupMode
Units:	Integer 0, 1, or 2
HiDS Location:	SensorlessA and SensorlessB pages
Description:	0 indicates the calculated sensorless RPM is in the open-loop range, 2 indicates closed-loop RPM, and 1 is in-between open-loop and closed-loop.
Variable name:	MxSmo.WorkingVelocity
Units:	RPM
HiDS Location:	SensorlessA and SensorlessB pages
Description:	The Working velocity is either the measured SensorlessX.FilteredRPM or calculated open-loop RPM.
Variable name:	MxSmo.Theta
Units:	Radians, 0 to 2PI
HiDS Location:	SensorlessA and SensorlessB pages
Description:	Theta is the normalized (0 to 2PI) final estimate of the electrical rotor position.

10.29System

Variable name:	Various
Summary:	Internal ESI Controller debug variables.
HiDS location:	Shown on System page in Advanced tab.
Description:	Read Only. Typically only ESI-internal debug variables

10.30Serial

Variable name:	CommandModeX
Units:	Integer, 0-2
HiDS Location:	Serial page
Description:	The ESI Motion RS422/CAN protocol can be used for either Current, Velocity, or Position control. Refer to ESI Document 100121, RS422 Command Protocol or to ESI Document 100211, CAN Command Protocol for details. The allowable commands are: 0=Torque 1=Velocity 2=Position
Variable name:	CommandCurrentScaleX
Units:	Unitless multiplier
HiDS Location:	Serial page
Description:	Defines the scale factor that is applied to received serial/can torque command,



This Document does not contain Technical Data or Technology as defined in the ITAR Part 120.10 or EAR Part 772.

Mx.IqSerialCommand/Mx.IqCANCommand

Variable name:	CommandVelocityScaleX
Units:	Unitless multiplier
HiDS Location:	Serial page
Description:	Defines the scale factor that is applied to received serial/can velocity command, MxVL.RPMSerialCommand/MxVL.RPMCanCommand
Variable name:	CommandPositionScaleX
Units:	Unitless multiplier
HiDS Location:	Serial page
Description:	Defines the scale factor that is applied to received serial position command, MxPL.RadiansSerialCommand/MxPL.RadiansCANCommand
Variable name:	SerialFeedbackEnable
Units:	Boolean, 0 or 1
HiDS Location:	Serial page
Description:	If the Controller supports an external RS422 command interface, this variable may be useful to debug that interface.
Variable name:	SerialBaud
Units:	Integer, the supported values are 300, 1200, 2400, 4800, 9600, 19.2K, 38.4K, 57.6K, 115.2K, 230.4K, 460.8K, 921.6K, 1Meg, and 1.2Meg baud.
HiDS Location:	Serial page
Description:	If the Controller supports an external RS422 command interface, this variable sets the baud rate used by the Controller.
Variable name:	SerialCommandRate_ms
Units:	Time in milliseconds
HiDS Location:	Serial page
Description:	For Controllers using RS422 communication for control, this variable sets RS422-(received)command timeout to report an error. A value of 0 disables the timeout check. If non-zero, and if RS422-packets are not recieved in this many milliseconds, a fault occurs and motor-control stops.
Variable name:	SerialFeedbackRate_ms
Units:	Time in milliseconds
HiDS Location:	Serial page
Description:	If the Controller supports an external RS422 command interface, this variable configures how many milliseconds between packets.
Variable name:	SerialChecksumErrorCount
Units:	Integer
HiDS Location:	Serial page
Description:	If the Controller supports an external RS422 command interface, this variable is incremented for each packet received with an invalid CRC or checksum (depending on the specific protocol).
Variable name:	SerialChecksumPassCount
Units:	Integer

HiDS Location:	Serial page
Description:	If the Controller supports an external RS422 command interface, this variable is incremented for each packet received with an valid CRC or checksum (depending on the specific protocol).
Variable name:	SerialCharactersReceived
Units:	Integer
HiDS Location:	Serial page
Description:	If the Controller supports an external RS422 command interface, this variable is incremented for each byte received on the RS422 interface.
Variable name:	SerialNumReportedErrors
Units:	Integer
HiDS Location:	Serial page
Description:	If the Controller supports an external RS422 command interface, this variable is incremented for error reported by internal serial transceiver. The transceiver error flag indicates that one of the error flags in the receiver status register is set. This flag is a logical OR of the break detect, framing error, overrun, and parity error enable flags.

10.31 Serial Encoder

The Serial Encoder page shows the state and configuration variables related to the 2 BISS Serial Encoder interfaces. The Serial Encoder variables are shown as follows:

Variable name:	SerialEncoderX.Degrees
Units:	Degrees, 0 to 360°
HiDS Location:	SerialEncoder page
Description:	Shows the motor position in degrees, normalized to 0 to 360°.
Variable name:	SerialEncoderX.Radians
Units:	Radians, 0 to 2 π .
HiDS Location:	SerialEncoder page
Description:	Shows the motor position in radians, normalized to 0 to 2 π .
Variable name:	SerialEncoderX.RadiansSummed
Units:	Radians, unbounded
HiDS Location:	SerialEncoder page
Description:	Unless explicitly cleared, this variable shows the total accumulated motor position in radians, since power on. Note the sign of the total motor movements is considered in the summing. For example, from 0, a command of +10 radians, plus a command of +5 radians results in RadiansSummed = 5 (because the second movement was -5 radians).
Variable name:	SerialEncoderX.RadiansPerSecond
Units:	Radians per Second
HiDS Location:	SerialEncoder page
Description:	Shows the unfiltered motor velocity in Radians per Second
Variable name:	SerialEncoderX.FilteredRadPerSec
Units:	Radians per Second
HiDS Location:	SerialEncoder page

Description:	Shows the filtered motor velocity in Radians per second
Variable name:	SerialEncoderX.Rpm
Units:	Revolutions per minute
HiDS Location:	SerialEncoder page
Description:	Shows the unfiltered motor velocity in revolutions per minute.
Variable name:	SerialEncoderX.FilteredRpm
Units:	Revolutions per minute
HiDS Location:	SerialEncoder page
Description:	Shows the filtered motor velocity in revolutions per minute.
Variable name:	SerialEncoderX.Timestamp
Units:	Time in microseconds
HiDS Location:	SerialEncoder page
Description:	Shows the time of the last measured rotor position.
Variable name:	SerialEncoderXData.PositionRaw
Units:	Integer (Counts)
HiDS Location:	SerialEncoder page
Description:	This is the un-modified 32bit position value received from the Serial Encoder.
Variable name:	SerialEncoderXData.PositionScaled
Units:	Radians
HiDS Location:	SerialEncoder page
Description:	This is the 32bit PositionRaw value multiplied by SerialEncoderConfig.ResolutionInRadians.
Variable name:	SerialEncoderXData.Error
Units:	Integer
HiDS Location:	SerialEncoder page
Description:	This is the un-modified 1-bit Error value received from the Serial Encoder.
Variable name:	SerialEncoderXData.Warning
Units:	Integer
HiDS Location:	SerialEncoder page
Description:	This is the un-modified 1-bit warning value received from the Serial Encoder.
Variable name:	SerialEncoderXData.CRCReceived
Units:	Integer
HiDS Location:	SerialEncoder page
Description:	This is the un-modified 6-bit CRC value received from the Serial Encoder.
Variable name:	SerialEncoderXData.CRCCalculated
Units:	Integer
HiDS Location:	SerialEncoder page
Description:	This is the 6-bit CRC value calculated from the received packet from the Serial Encoder.
Variable name:	SerialEncoderXData.NumErrorPackets
Units:	Integer

HiDS Location:	SerialEncoder page
Description:	This variable is incremented every time SerialEncoderData.Error occurs.
Variable name:	SerialEncoderXConfig.NumPositionBits
Units:	Integer
HiDS Location:	SerialEncoder page
Description:	This configures the number of position bits supplied by the BISS Serial Encoder. Note only 16, 18, 26, and 32-bit Serial Encoders are supported.
Variable name:	SerialEncoderXConfig.ResolutionInRadians
Units:	Integer
HiDS Location:	SerialEncoder page
Description:	This configures the scaler to multiply the PositionRaw value to convert to actual Radians.
Variable name:	SerialEncoderXConfig.VerifyCRC
Units:	Boolean, 0 or 1
HiDS Location:	SerialEncoder page
Description:	Indicates whether CRC passed (1) or failed (0)
Variable name:	SerialEncoderXConfig.ClockFrequencyMhz
Units:	MHz
HiDS Location:	SerialEncoder page
Description:	This configures the frequency of the clock to use to clock the data out of the Serial Encoder BISS Interface. Only 2, 4, and 8Mhz are supported. To change this configuration, one must change the value, save it into non-volatile (Flash) memory, and power-cycle the Controller.
Variable name:	SerialEncoderXConfig.CustomBissEnable
Units:	Boolean, 0 or 1
HiDS Location:	SerialEncoder page
Description:	If disabled (0), the legacy 26bit and 32bit configurations are used with hard-coded field lengths. If enabled (1), the PositionStartBit, NumPositionBits, and CrcStartBit set the encoder-parsing behavior.
Variable name:	SerialEncoderXConfig.PositionStartBit
Units:	Positive Integer (including 0)
HiDS Location:	SerialEncoder page
Description:	Sets the first bit position of the BISS measurement packet. This is often 0, but some encoders include stuff-bits or a multturn (absolute) position field in front of the required rotor-position field.
Variable name:	SerialEncoderXConfig.CrcStartBit
Units:	Positive Integer (including 0)
HiDS Location:	SerialEncoder page
Description:	Sets the first bit position of the BISS CRC field. This is often (PositionStartBit+ NumPositionBits)+2 (as there is an error-bit and warning-bit between the position field and CRC), but some encoders include stuff-bits after the required rotor-position field.
Variable name:	SerialEncoderXData.PositionWord0-3
Units:	Integer

HiDS Location: SerialEncoder page
Description: The BISS data starts 2 bits after the start-bit is received; these 4 16-bit values represent the 64 bits clocked in after the start bit.

10.32TestPoint

Variable name: **Various**
Summary: Internal ESI Controller debug variables.
HiDS location: Shown on TestPoint page in Advanced tab.
Description: Read Only. These are ESI-internal debug variables used to show the various states of the digital and analog test points.

10.33Temperature

Often all of the relevant temperature values can be viewed within the HiDS Run Panel.

Variable name: DSPTemp
Units: Degrees Celsius
HiDS Location: Temperature page
Description: Shows the current temperature of the Digital Signal processor.

Variable name: Mx.MotorTemp
Units: Degrees Celsius
HiDS Location: Temperature page
Description: Motor temperature in degrees celsius. The equation used to convert the measured voltage (from the varying motor-thermistor resistance) to degrees C is user-adjustable via the MxMotorTempX0Coeff (and X1, X2, X3) variables.

Variable name: MaIGBTTemp
Units: Degrees celsius
HiDS Location: Temperature page
Description: Servo-A IGBT (or MOSFET) internal-temperature reading.

Variable name: MbIGBTTemp
Units: Degrees celsius
HiDS Location: Temperature page
Description: Servo-B IGBT (or MOSFET) internal-temperature reading.

Variable name: MxMotorTempX0Coeff
Units: Float, unbounded
HiDS Location: Temperature page
Description: Coefficient in 3rd order polynomial to map thermistor resistance to a voltage measured which results in a temperature in degrees celsius

Variable name: MxMotorTempX1Coeff
Units: Float, unbounded
HiDS Location: Temperature page
Description: Coefficient in 3rd order polynomial to map thermistor resistance to a voltage measured which results in a temperature in degrees celsius

Variable name: MxMotorTempX2Coeff
Units: Float, unbounded
HiDS Location: Temperature page
Description: Coefficient in 3rd order polynomial to map thermistor resistance to a voltage measured which results in a temperature in degrees celsius

Variable name: MxMotorTempX3Coeff
Units: Float, unbounded
HiDS Location: Temperature page
Description: Coefficient in 3rd order polynomial to map thermistor resistance to a voltage measured which results in a temperature in degrees celsius

10.34Utility

Sintest is a flexible sine-wave / square-wave generator whose output can be bound to the inputs of the current, velocity, or position loop. Using an automated sine input is useful for basic system bring up as well as dynamic system response testing.

Variable name: SecondsSinceReset
Units: Time in seconds
HiDS Location: Utility page
Description: The number of seconds that have elapsed since the controller has been reset or powered up

Variable name: SecondsSinceEnabledMa
Units: Time in seconds
HiDS Location: Utility page
Description: Displays the time elapsed since Motor A was enabled

Variable name: SecondsSinceEnabledMb
Units: Time in seconds
HiDS Location: Utility page
Description: Displays the time elapsed since Motor A was enabled

Variable name: HeartBeatOutput
Units: Square wave, 0 to 1
HiDS Location: Utility page
Description: Provides a HeartBeatFreq Hz square wave that can be mapped to a digital output pins. This can be a good way to verify a servo is operational by only applying logic power.

Variable name: HeartBeatFreq
Units: Hertz
HiDS Location: Utility page
Description: Sets the frequency of the HeartBeatOutput square wave.

Variable name: SinTest.output
Units: Sinusoidal, -1 to 1
HiDS Location: Utility page

Description:	SinTest.output is the resulting sine output, at the frequency defined by SinTest.frequency, and SinTest.hz_per_second (if non-zero).
Variable name:	SinTest.angle
Units:	Radians, 0 to 2π
HiDS Location:	Utility page
Description:	SinTest.angle is the resulting sine output angle, as it rotates at the frequency defined by SinTest.frequency, and SinTest.hz_per_second (if non-zero).
Variable name:	SinTest.frequency
Units:	Hertz, 0 to 40KHz, depending on controller architecture.
HiDS Location:	Utility page
Description:	SinTest.frequency sets the frequency output of the SinTest function.
Variable name:	SinTest.hz_per_second
Units:	Hertz per second, 0 to ~200, depending on controller architecture.
HiDS Location:	Utility page
Description:	Should a swept-sine input be required (for frequency sweeps), SinTest. hz_per_second sets the hertz-per-second of the sweep. 0 disables the sweep (constant frequency).
Variable name:	SinTest.seconds_per_decade
Units:	Seconds per frequency-decade, unbounded
HiDS Location:	Utility page
Description:	For large frequency sweeps, this variable sets the number of seconds per frequency decade.
Variable name:	SinTest.min_hz
Units:	Hertz, 0 to 40KHz, depending on controller architecture
HiDS Location:	Utility page
Description:	For swept-sine operations, SinTest.min_Hz sets the frequency to roll-over to once SinTest.max_Hz is reached.
Variable name:	SinTest.max_hz
Units:	Hertz, 0 to 40KHz, depending on controller architecture
HiDS Location:	Utility page
Description:	For swept-sine operations, SinTest.max_Hz sets the maximum frequency to roll-over to SinTest.min_Hz.
Variable name:	SinTest.update_time
Units:	Seconds per update, typically 10-50ms, depending on controller architecture.
HiDS Location:	Utility page
Description:	SinTest. update_time is the sine-test update rate at which the angle and output are calculated.
Variable name:	SinTest.OutputIsSquarewave
Units:	Boolean, 0 or 1
HiDS Location:	Utility page
Description:	If true, the sine-test output is a square wave.
Variable name:	SinTest.ZeroOnEnable
Units:	Boolean, 0 or 1

HiDS Location:	Utility page
Description:	If true, zero's the sin-test output so that it starts from a known location when a servo is enabled
Variable name:	SinTest.CycleLimit
Units:	Integer
HiDS Location:	Utility page
Description:	Allows user to select how many cycles of SinTest.output to produce, once the number of cycles has been reached the output remains zero until the motor's are re-enabled.
Variable name:	SinTest.iq_amplitude
Units:	Amps, bounded by Mx.MinCurrentCommand and Mx.MaxCurrentCommand
HiDS Location:	Utility page
Description:	SinTest.iq_amplitude sets the scale used to multiply the SinTest.output to produce the IQ current input to the torque loop. When the SinTest is unused, this value is set to zero. The (SinTest.output * SinTest.iq_amplitude) result is summed with Mx.IqUserCommand, Mx.DesiredCurrent, and (Mx.lcmdInput * Mx.lcmdInputScale) to produce Mx.IqCmd.
Variable name:	SinTest.id_amplitude
Units:	Amps, bounded by Mx.MinCurrentCommand and Mx.MaxCurrentCommand.
HiDS Location:	Utility page
Description:	SinTest.id_amplitude sets the scale used to multiply the SinTest.output to produce the ID current input to the torque loop. When the SinTest is unused, this value is set to zero. The (SinTest.output * SinTest.id_amplitude) result is summed with Mx.IdUserCommand to produce Mx.IdCmd.
Variable name:	SinTest.VelocityCommandAmp
Units:	RPM, unbounded
HiDS Location:	Utility page
Description:	SinTest.VelocityCommandAmp sets the scale used to multiply the SinTest.output to produce the velocity input to the velocity loop. When the SinTest is unused, this value is set to zero. The (SinTest.output * SinTest.velocity_amp) result is summed with MxVL.RPMUserCommand, MxPL.DesiredRPM, and (Mx.lcmdInput * MxVL.VcmdInputScale) to produce MxVL.RPMCommand. This is useful to verify the system's closed-loop response.
Variable name:	SinTest.VelocityErrorAmp
Units:	RPM, unbounded
HiDS Location:	Utility page
Description:	SinTest.VelocityErrorAmp sets the scale used to multiply the SinTest.output to produce the velocity input summed with RPMError within the velocity loop. This is useful to verify the system's open-loop response.
Variable name:	SinTest.position_amp
Units:	Radians, unbounded
HiDS Location:	Utility page
Description:	SinTest.position_amp sets the scale used to multiply the SinTest.output to produce the position input to the position loop. When the SinTest is unused, this value is set to zero. The (SinTest.output * SinTest.position_amp) result is summed with MxPL.RadiansUserCommand, (Mx.lcmdInput * MxPL.PcmdInputScale) to produce MxPL.RadiansCommand.
Variable name:	SinTest.EnablePositionFilter

Units: Boolean, 0 or 1
 HiDS Location: Utility page
 Description: SinTest.EnablePositionFilter enables a 1Hz filter on the SinTest.position_amp value. In position-loop based systems, a step-change in the position can be undesired. Enabling this filter softens changes in the SinTest.position_amp variable.

Variable name: NullVariable
 Units: Unitless
 HiDS Location: Utility page
 Description: This variable has no internal function in the Controller, but it can be used to set a user-configurable Digital-input to "Off" or "Unused".

10.35 Velocity Loop

Variable name: MxVL.FeedbackRpmPeak
 Units: RPM, unbounded
 HiDS Location: Velocity Loop page
 Description: Stores the highest RPM observed for the Motor

Variable name: MxVLInputFilter-Type
 Units: Integer, 0-10
 HiDS Location: Velocity Loop page
 Description: Configures the filter type. The following filter types are available:
 0=Filter Disabled
 1=Low Pass
 2=High Pass
 3=Band Pass
 4=Notch
 5=Low Shelf
 6=High Shelf
 7=Gain
 8=Integral
 9=Low Pass 1st Order
 10=High Pass 1st Order

Variable name: MxVLInputFilter-GaindB
 Units: dB
 HiDS Location: Velocity Loop page
 Description: Filter gain in dB.

Variable name: MxVLInputFilter-CenterOrCorner
 Units: Hz
 HiDS Location: Velocity Loop page
 Description: The filters' center frequency

Variable name: MxVLInputFilter-Q
 Units: Hz
 HiDS Location: Velocity Loop page
 Description: Dimensionless parameter that describes how under-damped an oscillator or resonator is,[1] and

characterizes a resonator's bandwidth relative to its center frequency

Variable name:	MxVLInputFilter-Max
Units:	Hz
HiDS Location:	Velocity Loop page
Description:	Maximum cut-off frequency
Variable name:	MxVLInputFilter-SampleFreq
Units:	Hz
HiDS Location:	Velocity Loop page
Description:	Filters sampling frequency
Variable name:	MxVLInputFilter-Update
Units:	Boolean, 0 or 1
HiDS Location:	Velocity Loop page
Description:	In filter configuration is modified the changes will not take effect until an Update (1) is performed.
Variable name:	MxVL.VelCmdFiltered
Units:	RPM
HiDS Location:	Velocity Loop page
Description:	Filtered velocity command, filter used is MxVLInputFilter
Variable name:	MotorXVelocityEnable
Units:	Boolean, 0 or 1
HiDS Location:	Velocity Loop page
Description:	Enables the velocity control loop. This variable would normally be changed via the [Loop Gains] tab Control Mode dropdown list.
Variable name:	MxVL.MinRPMCommand
Units:	RPM, unbounded
HiDS Location:	Velocity Loop page
Description:	MxPL.MinRPMCommand limits the lower value of the input to the velocity loop.
Variable name:	MxVL.MaxRPMCommand
Units:	RPM, unbounded
HiDS Location:	Velocity Loop page
Description:	MxVL.MaxRPMCommand limits the upper value of the input to the velocity loop.
Variable name:	MxVL.MaxElectricalFreq
Units:	Hertz, unbounded
HiDS Location:	Velocity Loop page
Description:	Indicates the maximum motor electrical frequency supported by this controller. The electrical frequency = (MotorRPM / 60) * PolePairs.
Variable name:	MxVL.RotationalDirection
Units:	Signed multiplier, the valid values are -1 or 1.
HiDS Location:	Velocity Loop page
Description:	MxVL.RotationalDirection can swap the motor's rotational direction when operated in the velocity mode. Note when set to -1, this variable only swaps the sign of the incoming velocity command

and the reported velocity; the raw feedback speeds (i.e. ResolverX.Rpm) are unchanged by this variable. For sensorless feedback, the preferred method to change rotational directions is leave this variable as 1 and set MotorX.FeedbackDirection to -1, as this method changes all sensorless intermediate variable signs (i.e. SensorlessX.RPM).

Variable name: MxVL.RPMUserCommand

Units: RPM, unbounded

HiDS Location: Velocity Loop page

Description: One of the command input selections for the velocity loop input command multiplexers; if selected as the input source this will be the command into the velocity loop. This command is usually input via the HiDS RunPanel.

Variable name: MxVL.RPMCANCommand

Units: RPM, unbounded

HiDS Location: Velocity Loop page

Description: One of the command input selections for the velocity loop input command multiplexers; if selected as the input source this will be the command into the velocity loop. This command is usually received via the CAN bus using ESI CAN protocol

Variable name: MxVL.RPMSerialCommand

Units: RPM, unbounded

HiDS Location: Velocity Loop page

Description: One of the command input selections for the velocity loop input command multiplexers; if selected as the input source this will be the command into the velocity loop. This command is usually received via the CAN bus using ESI Serial protocol

Variable name: MxVL.RPMCommand

Units: RPM, unbounded

HiDS Location: Velocity Loop page

Description: MxVL.RPMCommand is the final velocity input into the velocity loop, and is the sum of MxVL.RPMUserCommand, MxPL.DesiredRPM (the output from the position loop), and $(MxVL.VcmdInput * MxPL.PcmdInputScale)$.

Variable name: MxVL.AccelLimitedRPMCommand

Units: RPM, unbounded

HiDS Location: Velocity Loop page

Description: For a step-response input to MxVL.RPMCommand, the MxVL.AccelLimitedRPMCommand variable is the resulting velocity action that is limited by the velocity acceleration value setting variable MxPL.AccelRPMPerSec.

Variable name: MxVL.DesiredVelocityChange

Units: RPM, unbounded

HiDS Location: Velocity Loop page

Description: The difference between the previous command (MxVL.RPMCommand) and the new command (MxVL.AccelLimitedRPMCommand)

Variable name: MxVL.AccelRPMPerSec

Units: RPM per second, positive only

HiDS Location: Velocity Loop page

Description:	For a step-response input to MxVL.RPMCommand, the MxVL.AccelRPMPerSec velocity acceleration variable limits the resulting velocity command, which is indicated by MxVL.AccelLimitedRPMCommand.
Variable name:	MxVL.IntegralHoldEnable
Units:	Boolean, 0 or 1
HiDS Location:	Velocity Loop page
Description:	If 1, then loop-output is greater than the maximum current setting, which indicates the loop is saturated. This will freeze the loop integral until the error changes sign (until less current is demanded).
Variable name:	MxVL.IntegralHoldActive
Units:	Boolean, 0 or 1
HiDS Location:	Velocity Loop page
Description:	This variable indicates the loop integral is being held constant; see IntegralHoldEnable.
Variable name:	MxVL.VcmdInputScale
Units:	RPM per volt, unbounded
HiDS Location:	Velocity Loop page
Description:	MxVL.VcmdInputScale sets the scale used to multiply the MxVL.VcmdInput to produce a velocity input to the velocity loop based on the analog voltage input provided. When MxVL.VcmdInput is unused, this value is set to zero. This value is summed with MxVL.RPMUserCommand and MxPL.DesiredRPM to produce MxVL.RPMCommand.
Variable name:	MxVL.VcmdInput
Units:	Volts, unbounded
HiDS Location:	Velocity Loop page
Description:	MxVL.VcmdInputScale sets the scale used to multiply the MxVL.Vcmdinput to produce a velocity input to the velocity loop based on the analog voltage input provided. When MxVL.VcmdInput is unused, this value is set to zero. This value is summed with MxVL.RPMUserCommand and MxPL.DesiredRPM to produce MxVL.RPMCommand.
Variable name:	MxVL.VerrorInputScale
Units:	RPM per volt, unbounded
HiDS Location:	Velocity Loop page
Description:	Sets the scale used to multiply the MxVL.VerrorInput to produce a velocity input to the velocity loop for an open-loop response, based on the analog voltage input provided. When MxVL.VerrorInput is unused, this value is set to zero. This value is summed with MxVL.RPMUserCommand and MxPL.DesiredRPM to produce MxVL.RPMErrorPlusOffset.
Variable name:	MxVL.RPMError
Units:	RPM, unbounded
HiDS Location:	Velocity Loop page
Description:	MxVL.RPMError indicates the velocity error. $MxVL.RPMError = MxVL.AccelLimitedRPMCommand - \langle FeedbackType \rangle .RPM$.
Variable name:	MxVL.RPMErrorPlusOffset
Units:	RPM, unbounded
HiDS Location:	Velocity Loop page

Description:	Indicates the velocity error but also sums in the open loop-response analog inputs
Variable name:	MxVL.RPMErrIntegral
Units:	RPM
HiDS Location:	Velocity Loop page
Description:	Indicates the integral of the loop.
Variable name:	MxVL.FrictionFeedForward
Units:	Amps per RPM
HiDS Location:	Velocity Loop page
Description:	For motor-systems that may have varying friction or inertia that is dependent on motor-speed, MxVL.FrictionFeedForward is a friction-based feed-forward to the velocity-loop. This feed-forward produces a current, MxVL.FrictionFeedForwardCurrent, that is added into Mx.DesiredCurrent. $\text{MxVL.FrictionFeedForwardCurrent} = \text{MxVL.AccelLimitedRPMCommand} * \text{MxVL.FrictionFeedForward}.$
Variable name:	MxVL.FrictionFeedForwardCurrent
Units:	Amps per RPM
HiDS Location:	Velocity Loop page
Description:	For motor-systems that may have varying friction or inertia that is dependent on motor-speed, MxVL.FrictionFeedForward is a friction-based feed-forward to the velocity-loop. This feed-forward produces a current, MxVL.FrictionFeedForwardCurrent, that is added into Mx.DesiredCurrent. $\text{MxVL.FrictionFeedForwardCurrent} = \text{MxVL.AccelLimitedRPMCommand} * \text{MxVL.FrictionFeedForward}.$
Variable name:	MxVL.StepFeedForward
Units:	Amps, unbounded
HiDS Location:	Velocity Loop page
Description:	For motor-systems that may have an initial inertia or startup friction to overcome, MxVL.StepFeedForward is an inertia-based feed-forward to the velocity-loop. This feed-forward produces a current, {MxVL.StepFeedForward * signof(MxVL.AccelLimitedRPMCommand)} that is added into Mx.DesiredCurrent.

10.36 Hardware Test

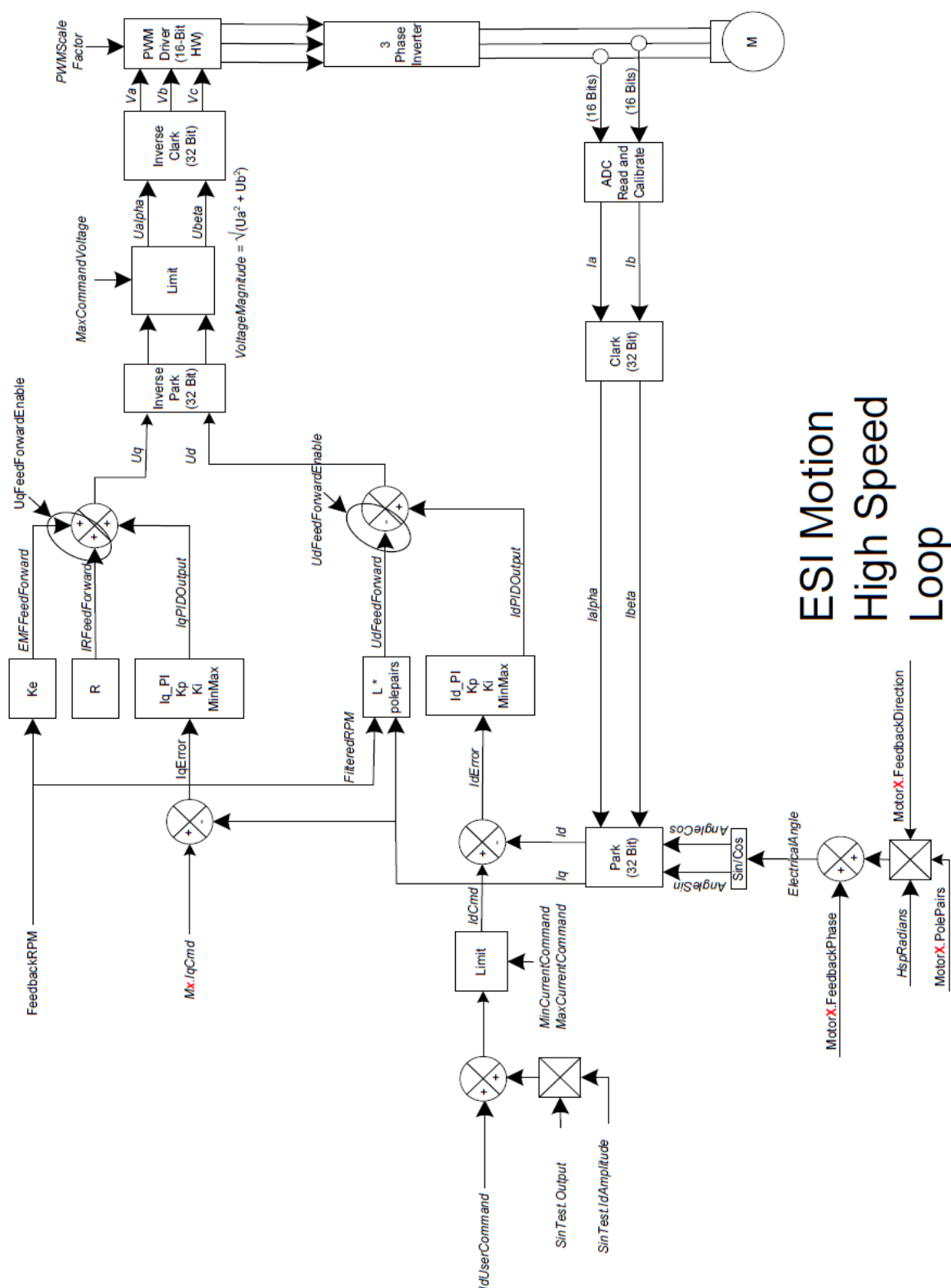
Variable name:	Various
Summary:	Internal ESI Controller debug variables.
HiDS location:	Shown on Hardware Test page in Advanced tab.
Description:	Read Only. These are ESI-internal debug variables used to test various internal sub-systems of the Controller.

10.37 User Defined

Variable name:	Various
Summary:	Displays all variables added to the User Defined page.
HiDS location:	Shown on Hardware Test page in Advanced tab.
Description:	Read Only. This page displays all variables added to the User Defined page. The User Defined page is described above.

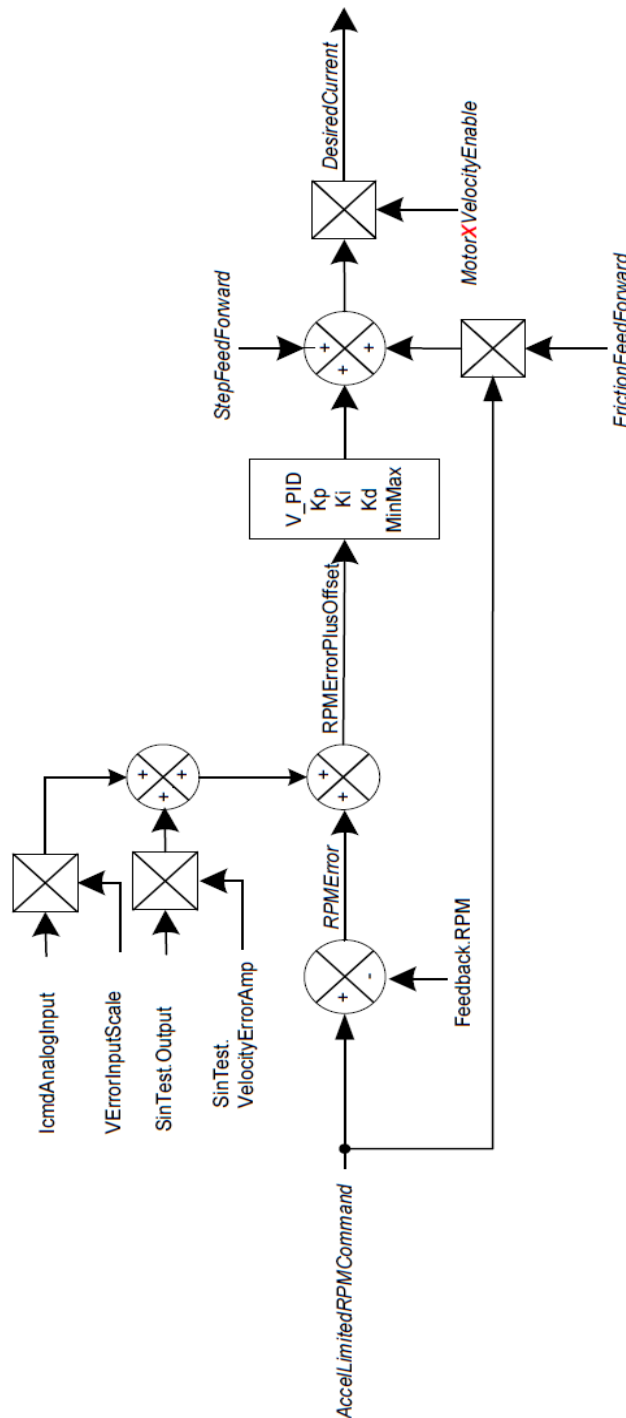
11 Appendix B: The ESI Motion Current Loop

Note some of the variable names are abbreviated in the large drawing in order to fit on one page.



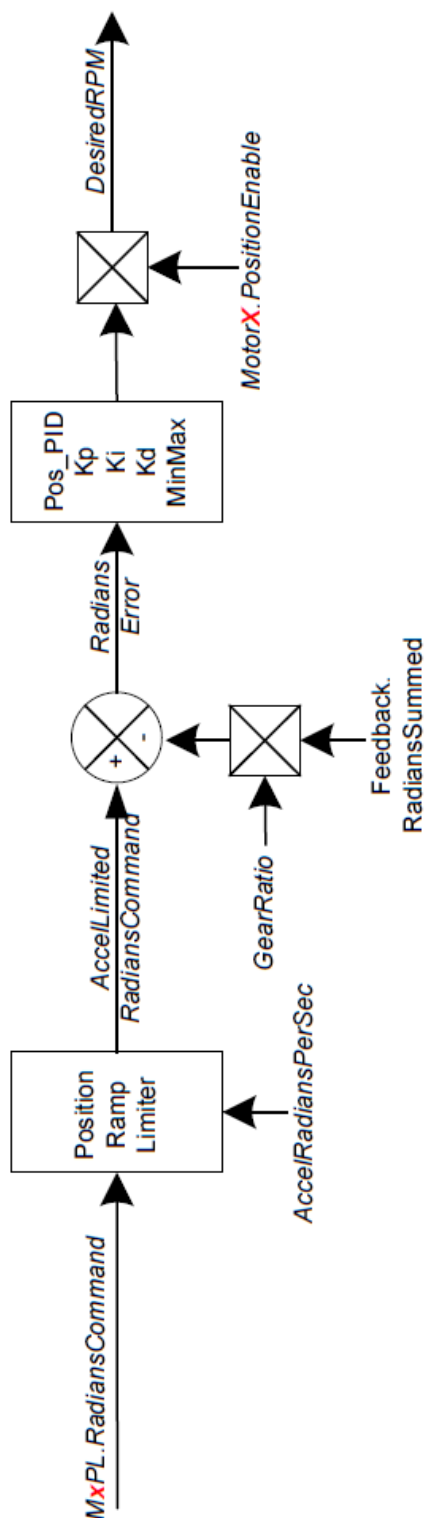
12 Appendix C: The ESI Motion Velocity Loop

Note some of the variable names are abbreviated in the large drawing in order to fit on one page.



ESI Motion Velocity Loop

13 Appendix D: The ESI Motion Position Loop



ESI Motion Position Loop