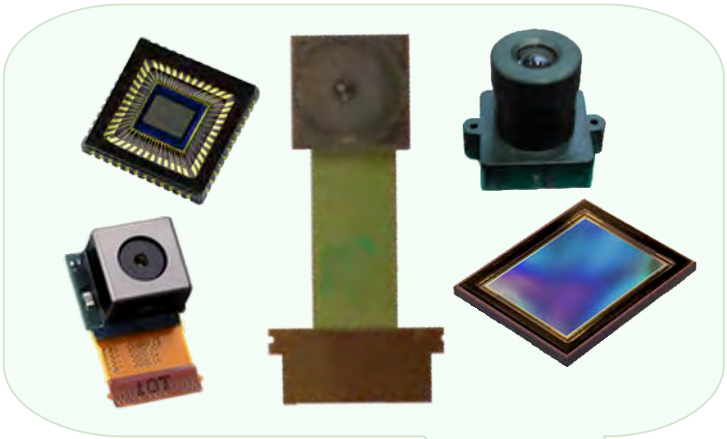
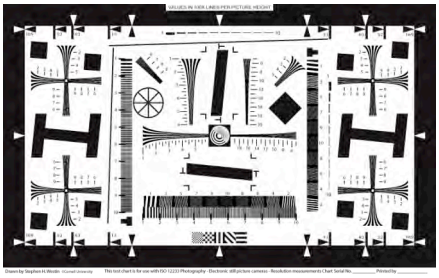
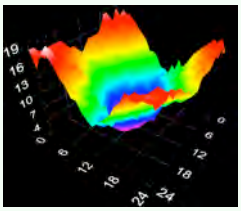
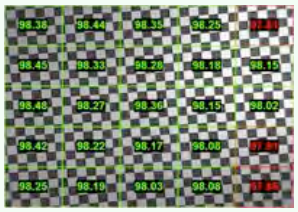
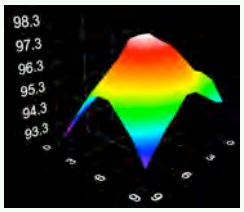
ISL-3200™ Digital Imaging Sensor Interface and Test Solution

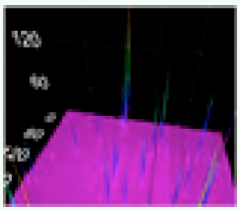












www.jovasolutions.com
(415) 348.1400
fax (415) 348.1414

Table of Contents

1. INTRODUCTION

ISL™ is a cost-effective combination of sensor interface, test electronics, and application software that provides complete communications, image capture, and characterization testing, of a variety of image sensors.

The ISL™ line of image sensor instrumentation (including the ISL-1600 and ISL-3200) is fully compatible with the LabVIEW™ development environment and the TestStand™ Test Sequencer and Test Management Platform from National Instruments.

The SL Test and Automation Suite includes a complete LabVIEW driver package as well as TestStand custom step types and pre-written TestStand sequence files for complete control and automation of the ISL instrument as well as control and automation of advanced image analysis routines that are part of the ISL Advanced Analysis package.

The ISL Test and Automation Suite provides complete control over the instrumentation and control hardware, including power supply control, master clock control, sensor I2C or SPI communications, and digital image capture. Also included is the ISL Script Engine, which can execute simple text files containing a sequence of individual commands. This enables automation of complicated setup sequences or repetitive tasks. The same script files can be generated and/or used with the ISL Application Software.

This guide assumes an in depth knowledge of both the LabVIEW™ and TestStand™ product from National Instruments. More information on these products can be found at http://www.ni.com/.

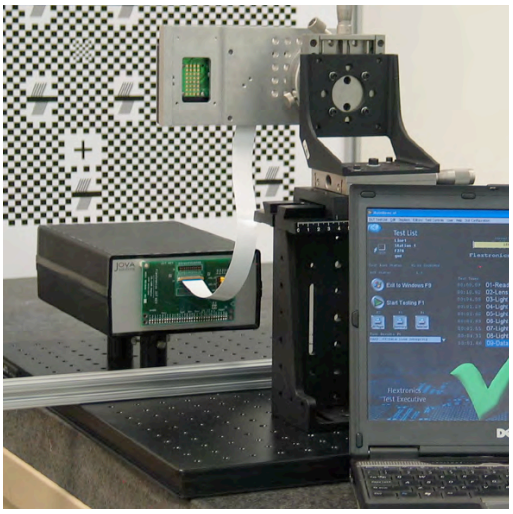
2. CONTACT INFORMATION

2.1 SALES AND SUPPORT

Jova Solutions, Inc.
965 Mission Street, Suite 600
San Francisco, CA 94103

415-348-1400 Office
415-348-1414 Fax
415-348-1408 Technical Support
800-755-1400 Toll Free Sales

http://www.imagesensorlab.com
info@imagesensorlab.com



3. REFERENCE DOCUMENTS

Description	*Doc. No*	*Company/Author*	*Rev. Date*
ISL-3200 Product Specification	210-0004-08	Jova Solutions	07/14/2009
ISL-3200 Basic User Manual	210-0001-08	Jova Solutions	07/14/2009
ISL-3200 Advanced Analysis Guide	210-0002-08	Jova Solutions	07/14/2009
ISL-3200 Quick Start Guide	210-0008-02	Jova Solutions	07/14/2009
ISL-3200 Test and Automation Suite Guide	210-0003-05	Jova Solutions	07/14/2009

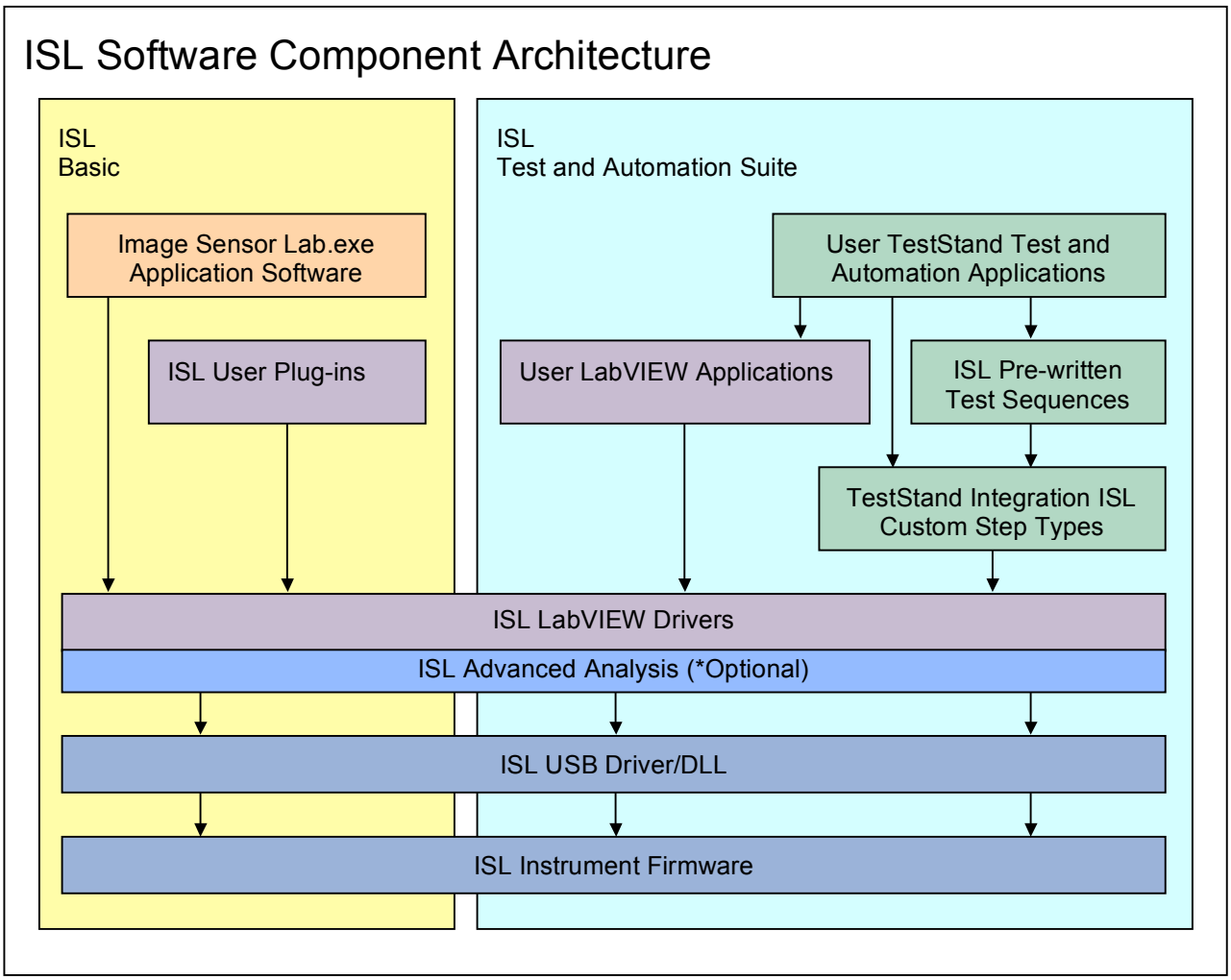
Here are the documents related to the ISL-3200, which you may find useful:

4. GENERAL CONCEPTS

ISL consists of application software running on a host computer, which communicates with the ISL image sensor interface electronics box. Users typically connect image sensors to the I/O connector on the ISL hardware with a custom adapter board and use the application software to control the control signals and instruments within the ISL instrument to communicate with and acquire images from the image sensor.

The ISL Test and Automation Suite provides a development platform for building customized test and automation applications that utilize the ISL instrument and the pre-written control and analysis routines normally access through the ISL application software.

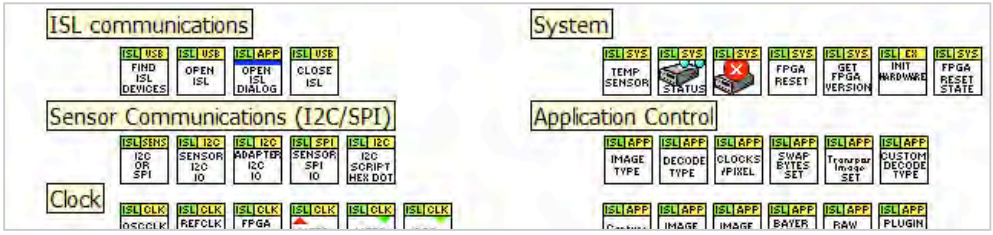
The ISL software layers and components are shown in the diagram below.



As shown in the diagram above, all communications and control over the ISL instrument go through a common USB driver and DLL.

Next comes the final software foundation layer, the LabVIEW Drivers. The LabVIEW drivers provide complete control over the ISL instrument, and the full functionality of the ISL instrument and USB driver/DLL are exposed through these LabVIEW drivers. All end user applications and test and automation software share this common driver software.

ISL Advanced Analysis routines are also contained in this layer of the software and are also shared with all higher-level software layers. The Advanced Analysis routine APIs are available in all installations of the software and development tools, but will not execute without error unless the ISL instrument attached to the computer has been licensed and keyed for the Advanced Analysis package.



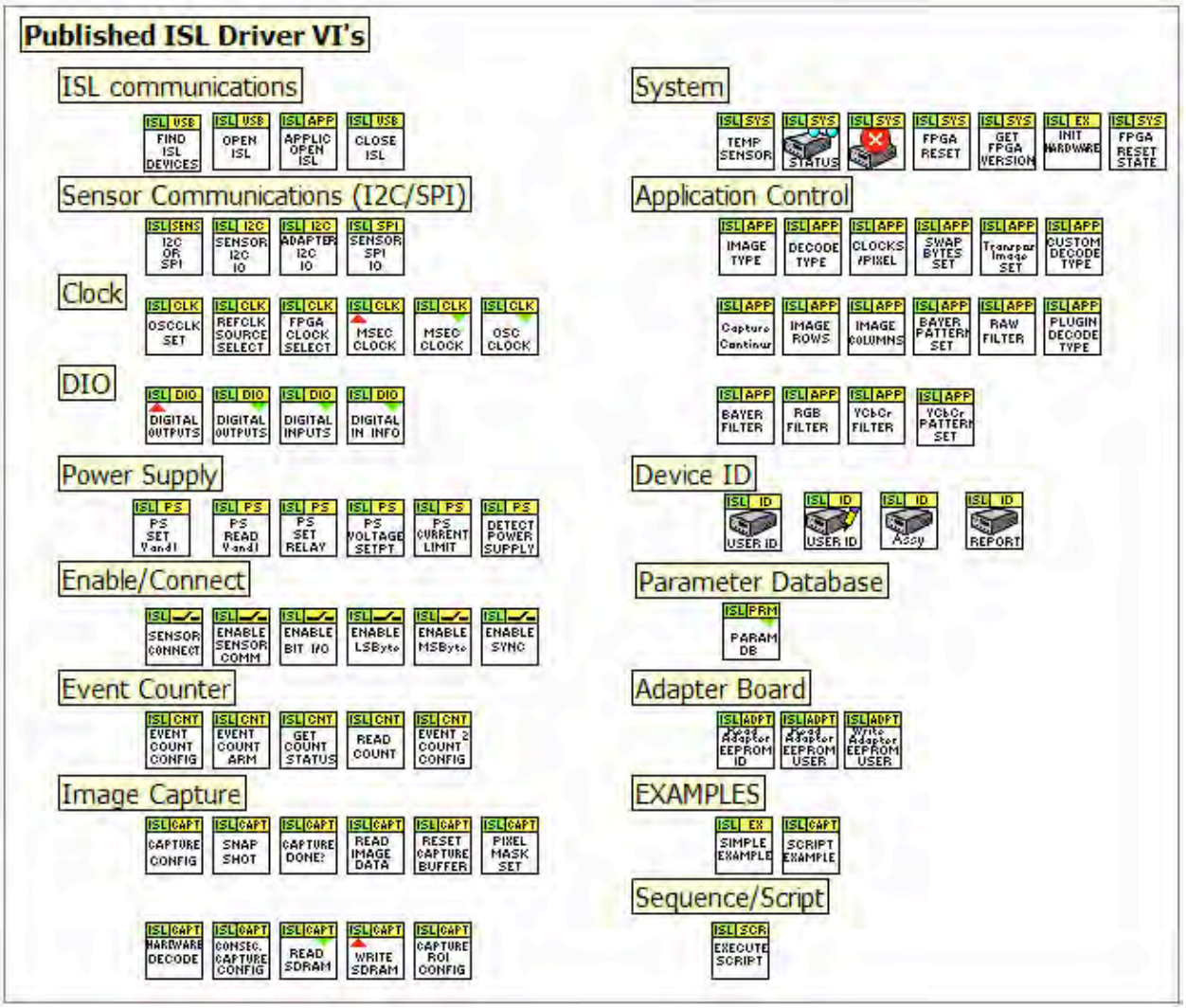
The LabVIEW driver layer is fully exposed to users and all of the VIs contained on the VI Tree can be used in custom LabVIEW™ or TestStand™ software applications.

Although users can call the LabVIEW Driver VIs directly from TestStand, the ISL Driver VI functionality has been more tightly integrated into TestStand™ and implemented as “custom step types”. Instead of hand linking the VIs into TestStand™, users have all the same functionality available as custom step types and easily inserted into user sequences directly from the TestStand™ menus.

5. GETTING STARTED WITH ISL TEST AND AUTOMATION SUITE

5.1 USING ISL IN THE LABVIEW DEVELOPMENT ENVIRONMENT

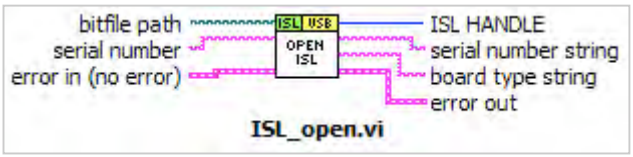
After the ISL Test and Automation Suite is installed, a folder of LabVIEW Driver VIs is created inside the ISL working folder. Opening the diagram of the ISL_Tree.vi provides direct access to all the published ISL VIs and these VIs can easily be copied and pasted into any user VI diagrams. The block diagram of the ISL_Tree.vi is shown below and breaks the VIs into functional groups corresponding to the various subsystems within the ISL instrumentation hardware.



5.1.1 ISL Communications

A connection must be established with the ISL instrumentation hardware before issuing any subsequent commands or requests.

The ISL_open.vi is used for opening a connection to the ISL hardware. This VI creates an ISL Handle reference that is a required input of almost every other VI in the VI Tree.



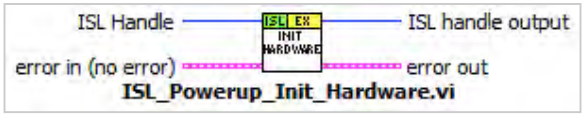
When opening a connection to the hardware, an FPGA bit file is also downloaded. The serial number input to this VI is optional and if left unwired or empty, the first ISL device found will be opened and an ISL Handle reference returned.

Every LabVIEW application that needs access to the ISL hardware should start with this VI and conclude with a call to the ISL_close.vi

Once the ISL_open.vi has been called successfully and a valid ISL Handle acquired, communications with the ISL Hardware can take place.

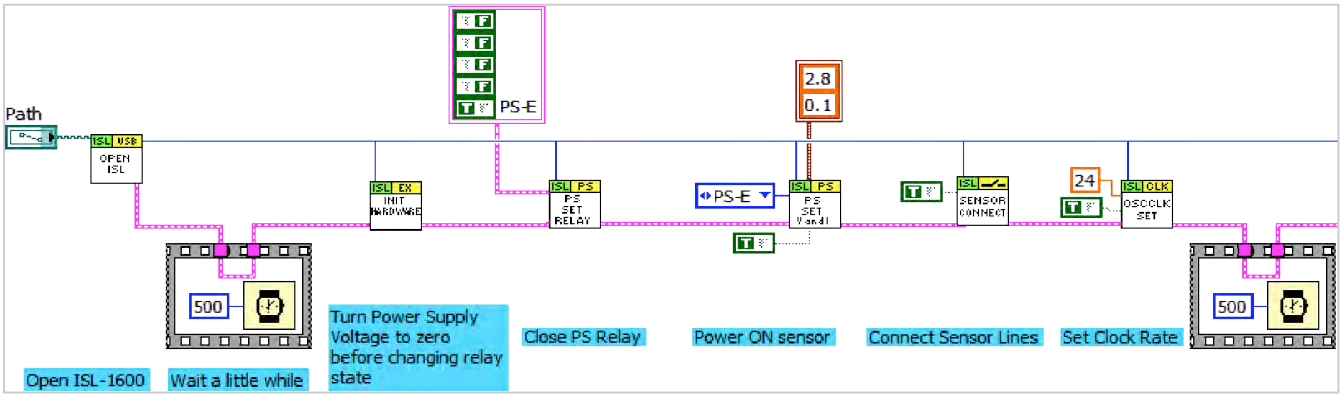
5.1.2 Initializing the ISL Hardware

The ISL instrumentation hardware should be initialized at the beginning of any application or whenever the power to the ISL hardware is first applied. The ISL_Powerup_Init_Hardware.vi must be called, which contains a collection of calls to put the instrument in a proper and safe startup state.



This VI first resets the FPGA and then sets all power supplies to zero voltage output and opens the mechanical relay connection to the image sensor. The on-board oscillator is also turned off and disconnected from the image sensor. All of the FET switches are opened to completely disconnect the image sensor from the ISL hardware electronics. This VI also sets the default image capture settings as well as configures some lower level hardware settings that are required for proper operation.

5.1.3 LabVIEW Code Example



The LabVIEW example code above depicts a typical LabVIEW application startup sequence. The sequence starts with the mandatory calls to ISL_Open.vi and ISL_Powerup_Init_Hardware.vi. After these initial calls are complete, any of the other ISL Tree VIs can be called. In this example, power supply E is then enabled and the voltage setpoint is set to 2.8 V with a 0.1A current limit. After that, the sensor signal FETs are closed to connect the image sensor to the ISL instrument electronics. To complete this example, the on-board oscillator is then set to 24 MHz.

5.1.4 Building Custom Applications Using LabVIEW

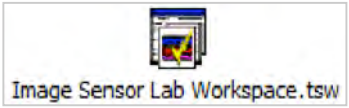
Using the programming concepts discussed above, you can create custom LabVIEW applications that fully utilize the ISL instrument and ISL Advanced Analysis routines. Complete descriptions of each ISL_Tree.vi VI are contained in Chapter 6 of this document.

5.2 USING ISL IN THE TESTSTAND DEVELOPMENT ENVIRONMENT

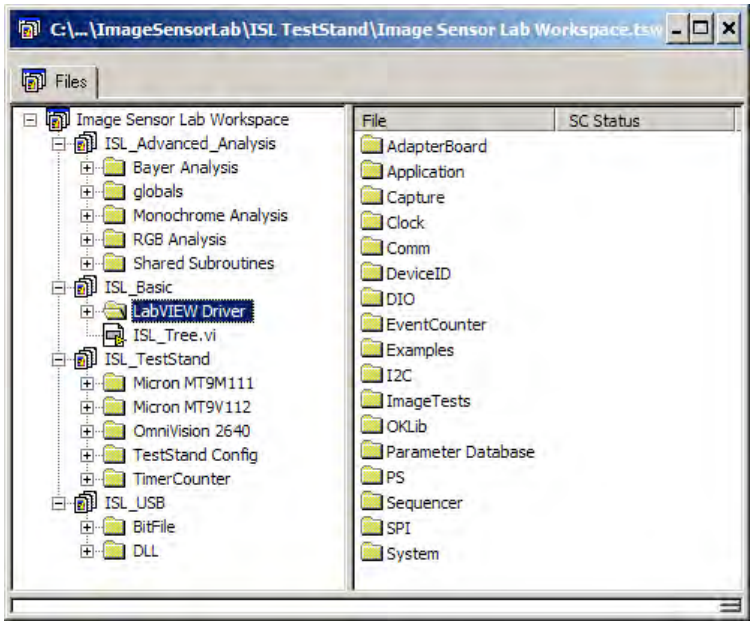
After installation of the ISL Test and Automation Suite, an ISL TestStand folder is created in the ISL working directory. All of the TestStand specific tools and pre-written sequence files are located in this directory; TestStand developers can also call VIs located in other directories within the ISL working directory. All of the VIs contained in the LabVIEW Drivers directory can be called from TestStand, but it is easier to use the built-in custom step types that are accessed directly from the ISL TestStand menus.

5.2.1 ISL Workspace and Project Files

The ISL Test and Automation Suite contains a TestStand Workspace file and sub project files that organize most of the VIs, sequence files, and other support files that make up the Test and Automation Suite.



The ISL Workspace is shown at the right. The Workspace is made up of four sub-projects. The lowest level is the ISL_USB project, which contains the ISL instrument FPGA bit file and communications DLL used by all higher-level software.



It is best to always keep the Images Sensor Lab Workspace file open so that TestStand can quickly find all of the components it may need when working with ISL.

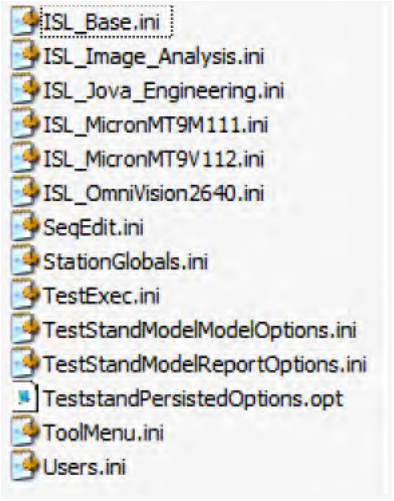
The ISL_Basic project contains the LabVIEW Driver VIs and the ISL_Tree.vi. These LabVIEW Drivers are used by both the ISL_TestStand project VIs and sequences as well as the ISL_Advanced_Analysis project VIs and sequences.

The Workspace and Project files are useful for locating where the resources are on disk for the various ISL components and when deploying TestStand test sequences to other Test Stations or computers. These files contain only disk file pointers and do not contain any additional information about how those VIs and resource files might be linked in to TestStand. The custom step type mapping to various VIs, sequences, and other resource files is covered later in section 5.2.3, ISL Type Palettes.

5.2.2 TestStand Configuration Files

The ISL Test and Automation Suite contains a TestStand configuration directory that can be used to customize the TestStand development environment for use with ISL hardware and software. These configuration files are located at ***Program Files\ImageSensorLab\ISL TestStand\TestStand Config***

The configuration .ini files that start with ISL_ are the type palette files that map the TestStand custom step types to the underlying LabVIEW Driver VI's. The .ini type palette files contain additional mappings between the custom step type and other VIs or pre-written sequence files.



TestStand can be configured to use this directory for all configuration files, or the type palette files can copied from this directory into the existing default TestStand configuration directory.

To configure TestStand to use the ISL configuration directory follow these steps:

A. From the TestStand Configure menu choose Station Options…
B. At the bottom of the preferences tab is a Configuration Directory entry space. Enter the path (***\ImageSensorLab\ISL TestStand\TestStand Config***)
C. Press the OK button to accept the change.

It may be preferable to simply copy the ISL type palette files (the .ini files beginning with ISL_) to the existing TestStand configuration file directory.

After modifying the TestStand configuration exit and restart TestStand to activate the configuration change.

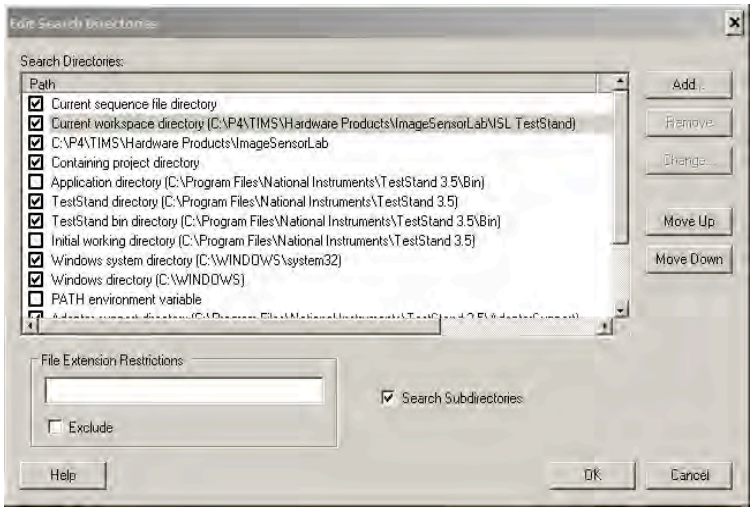
5.2.3 TestStand Search Directory Configuration

The TestStand search directories must be properly configured so that the ISL sequence files can properly find their resources before execution. To accommodate installation file structure variations it is advised that the ISL Workspace file is included in the search directories. To configure TestStand to search the ISL Workspace contents, follow these steps:

A. From the TestStand Configure menu choose Search Directories…
B. Press the Add… button.
C. Navigate to and enter the path ***\ImageSensorLab\ISL TestStand\Image Sensor Lab Workspace.tsw.***
D. Press the OK button to accept the change.

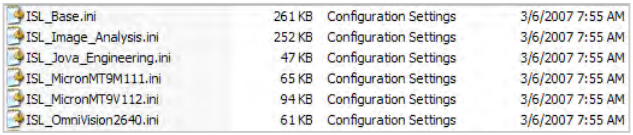
Check the box associated with the newly created search directory, and drag it toward the top of the search directories list, as shown in the screen shot below.

Also ensure that the Search Subdirectories checkbox is checked as shown in the screen shot, when the ISL Workspace.tsw file is selected in the list of search directories.

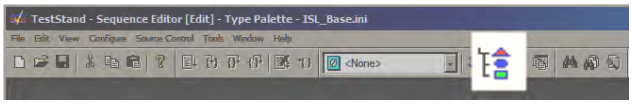


5.2.4 ISL Type Palettes for TestStand

The ISL_Basic.ini type palette file contains all the custom step types associated with the LabVIEW Driver Tree.vis. These custom step types provide complete control over all the subsystems of the ISL instrument, including power supplies, programmable oscillator, digital I/O, and I2C sensor communications.



The ISL_Image_Analysis.ini type palette contains the custom step types associated with the ISL Advanced Analysis routines. The ISL hardware must be licensed and keyed for Advanced Analysis or the Test Step will error and fail without returning results.

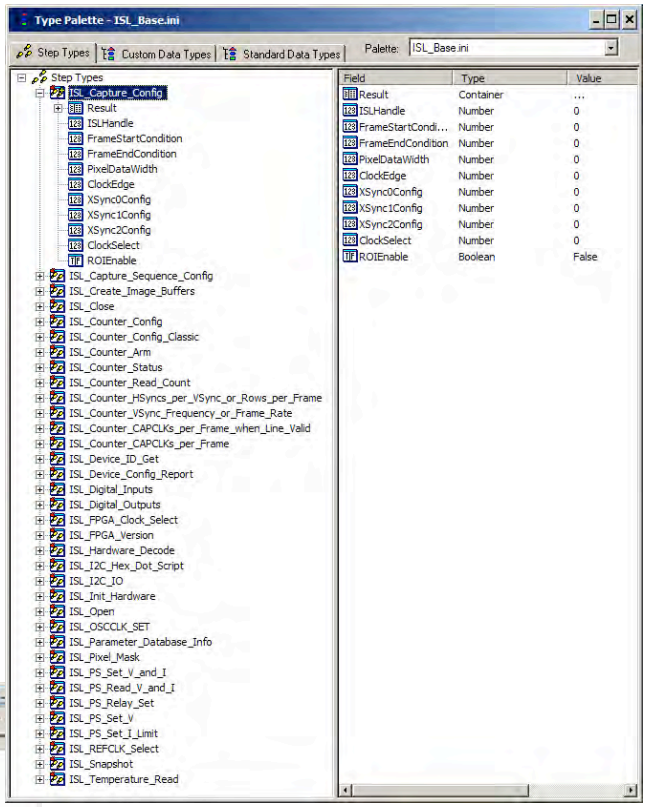


Several example type palettes for image sensor specific step types are also provided. These example type palettes are very useful and can often be slightly modified to support other image sensor types. Also, many new users of TestStand find that starting from working examples is a good way to learn and a quick way to get up and running.

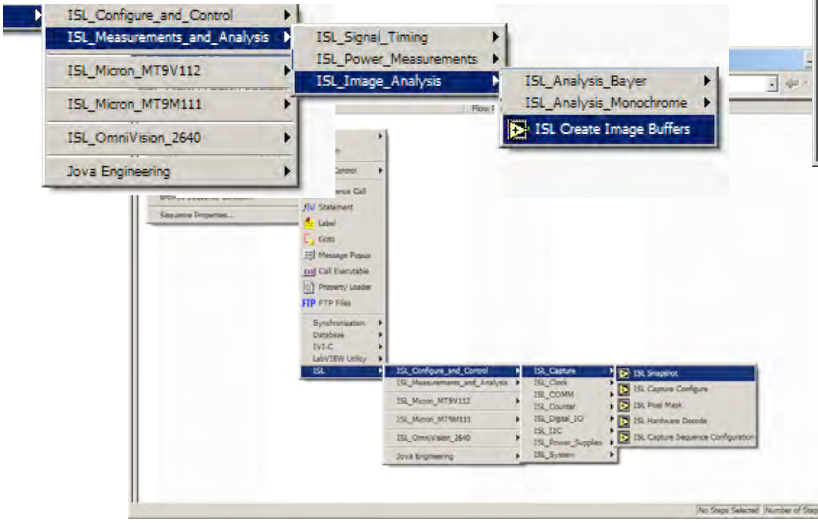
To view the type palette files from the TestStand main sequence window click on the “Type Palette” icon in the toolbar below the main menu.

The window shown to the right is displayed showing type palette information.

The drop-list in the upper right of this window selects which type of palette is currently being viewed in the window below. In this case the ISL_Base.ini type file contents are being displayed. The tree structure on the left expands to show each step types parameters. The ISL_Capture_Config step type is show expanded and all the individual capture configuration parameters are shown indented below.



When the ISLb custom step types are used in your applications, these same parameters are exposed as programmable inputs and outputs to the ISL routines.



5.2.5 Accessing ISL Step Types in TestStand

Once TestStand is properly configured to use the ISL type palette files and custom defined step types, most functions are easily accessed directly from the TestStand menus.

Right click in the Sequence workspace and choose the top item Insert Step. In the bottom section resides the ISL submenu. The top submenu is the ISL_Configure_and_Control. This submenu contains all the associated step types for all the ISL LabVIEW Driver VIs that control all the subsystems of the ISL instrument.

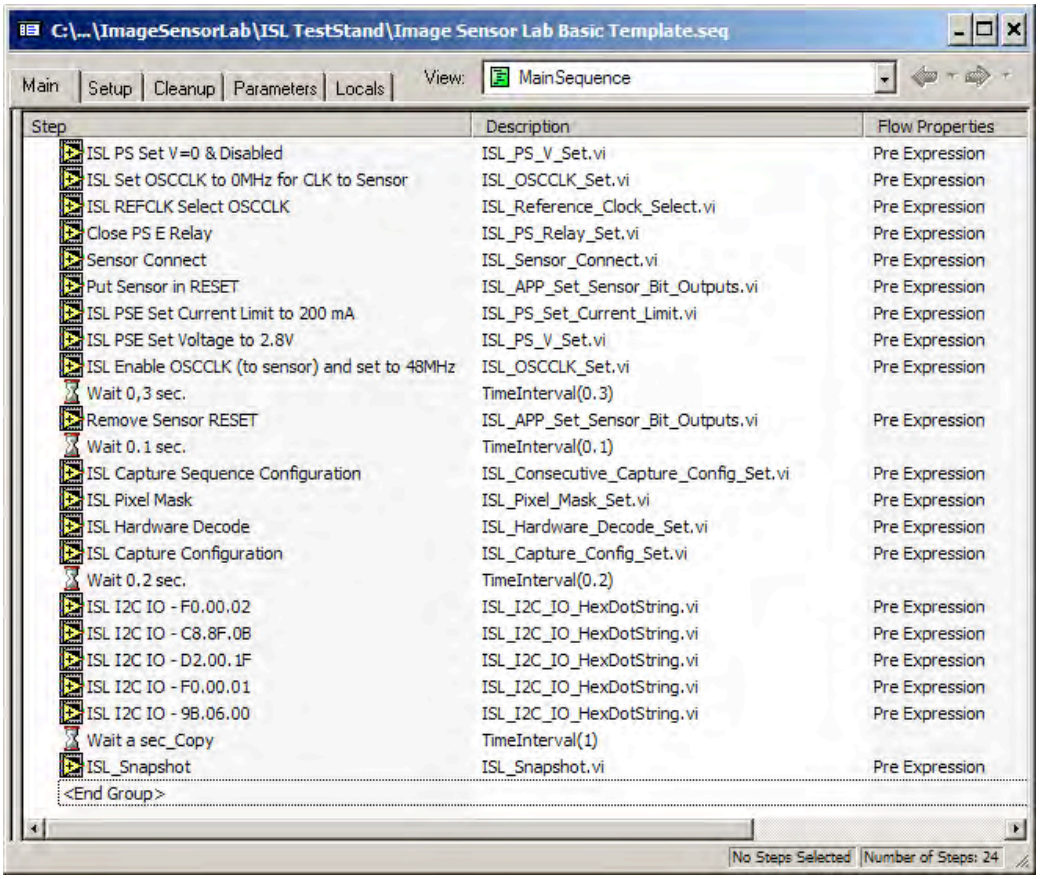
The next major submenu under ISL is the ISL_Measurements_and_Analysis submenu, which contains both pre-written test and measurement sequence file test steps and LabVIEW custom step types. The ISL Advanced Analysis routines are also accessed from this submenu.

Several submenus for some specific image sensors are also provided. These image sensor specific custom step types can serve as examples and templates for other image sensors.

5.2.6 ISL TestStand Template Sequence File

The ISL Test and Automation Suite includes a sequence file template that contains all the recommended steps to properly open, configure, and operate the ISL instrument. The template file is located at ***\ImageSensorLab\ISL TestStand\Image Sensor Lab Basic Template.seq***.

The Setup tab contains all of the steps necessary to properly startup the ISL instrument, regardless of which image sensor you may have connected. The ISL Device ID, ISL Device Configuration Report, and ISL FPGA Version test steps are optional and serve as examples of device configuration calls.



The first step of any top-level sequence should be ISL Open, one of the provided custom test step types. This opens a connection to the ISL instrument, downloads a bit file, and passes a reference to the open device out to a sequence parameter so that the connection can be shared with subsequent test steps.

The next mandatory test step is the ISL Initialize Hardware (ISL_Powerup_Init_Hardware.vi) which initializes all the subsystems within the ISL instrument and prepares it for user initiated commands.

The Main tab of the template sequence file contains steps associated with bringing up a typical image sensor.

Refer to the sensor specification for the connected sensor as to the required start up sequence. Usually you want to start with the power supplies off and disconnected from your image sensor. The master clock to the sensor should also be disabled and set to 0. Leaving a master clock running and connected to your sensor can give you unexpected results and generally will interfere with proper startup of the sensor.

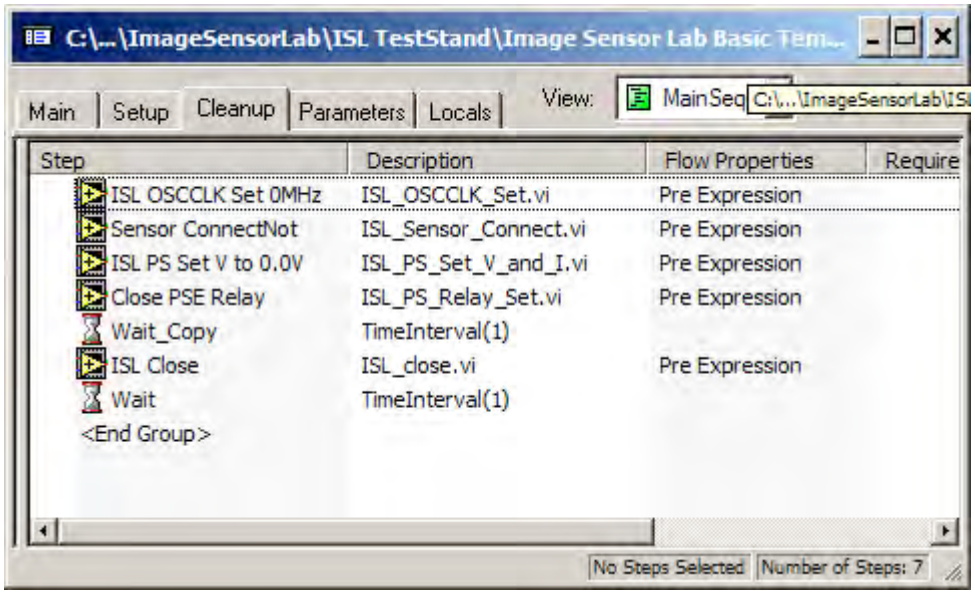
The power supply relay is then closed to connect the image sensor and power. The Sensor Connect step closes the FET switches connecting the image sensor clock, data, and sync lines to the ISL instrument. A digital output line of the ISL instrument is then toggled to hold the image sensor in RESET mode. The power supplies are then adjusted to 2.8V setpoint and 200mA current limit. The master clock is then set to 48 MHz and enabled. The digital output line connected to the sensor RESET line is then toggled back to ENABLE the image sensor. At this point the sensor should be clocking out image data. The next several steps in the template sequence set up the image capture configuration of the ISL instrument.

Next are a few steps that communicate with the image sensor via I2C and configure the image sensor for the desired operating more and image output format.

The final step commands a single frame image capture.

The Cleanup tab contains the mandatory steps to take at the end of a test cycle, before disconnecting the adapter board, removing an image sensor, or unplugging the AC power cord connected to the ISL instrument.

The master clock to the sensor should always be turned off and the FET switches that connect the sensor to the ISL instrumentation should be opened. The power supplies should be turned off and the relays opened, isolating the sensor. Finally the connection to the ISL instrument is closed, completing the shutdown sequence.

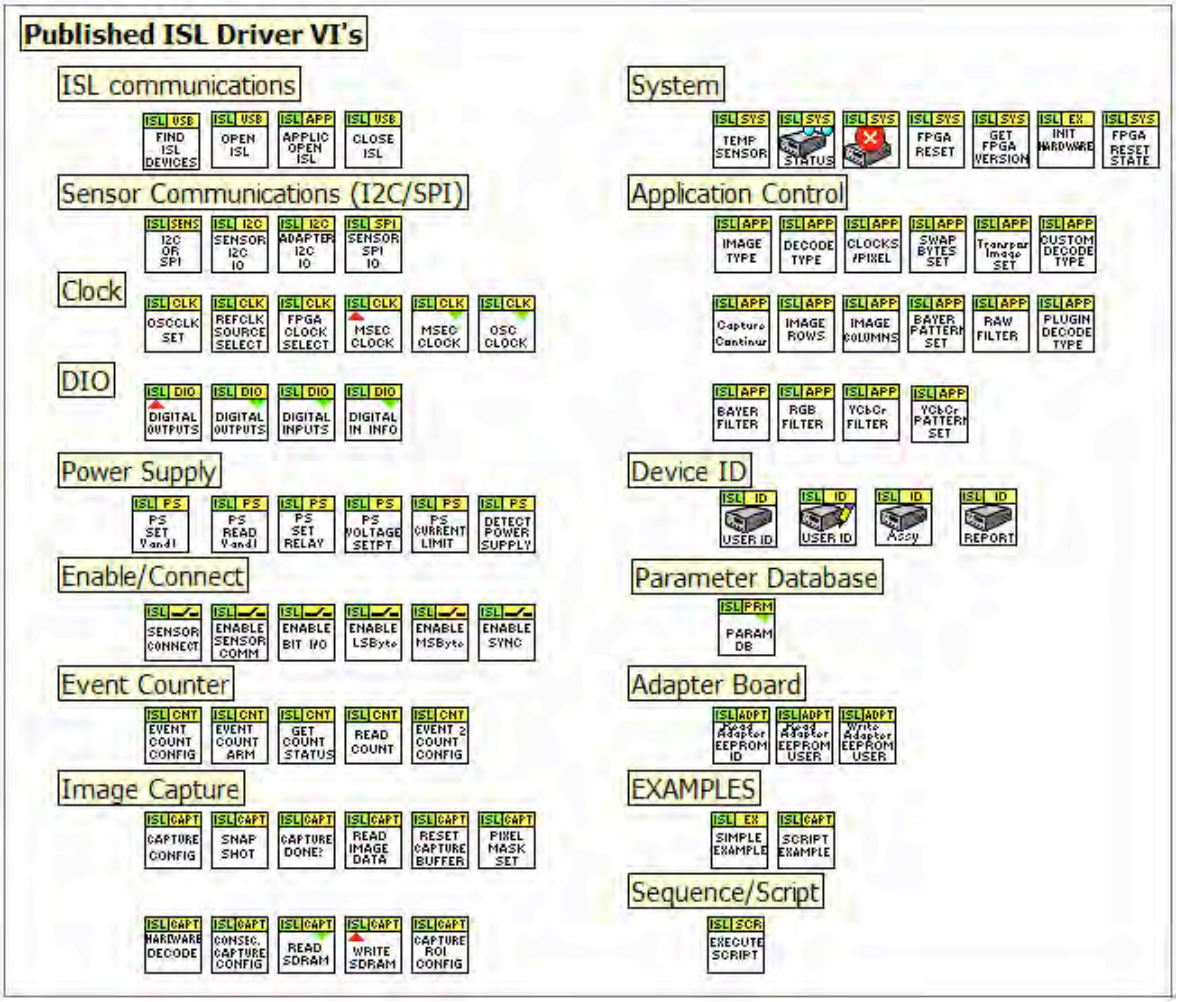


6. ISL LABVIEW INSTRUMENT DRIVERS

The ISL LabVIEW Driver package consists of a collection of pre-written LabVIEW VIs that control the various functions of the ISL instrumentation hardware.

The ISL_Tree.vi contained at the root level of the ISL LabVIEW Driver directory contains a copy of all the VIs that make up the LabVIEW driver package. The block diagram of the ISL_Tree.vi is shown below and breaks the VIs into functional groups corresponding to the various subsystems within the ISL instrumentation hardware.

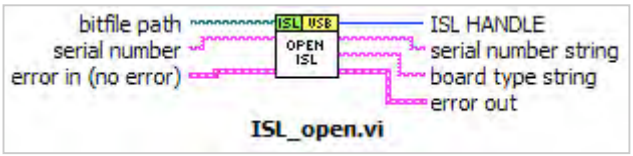
The LabVIEW drivers are discussed in much greater detail in the ISL Drivers Guide. The ISL LabVIEW drivers are used not only when developing LabVIEW test and automation software, but are also used in the TestStand development platform. Every VI in the LabVIEW driver tree has a corresponding TestStand custom step type. The TestStand custom step types are actually direct conduits to the LabVIEW driver VIs. Therefore, regardless of whether you develop in LabVIEW or TestStand, the ISL LabVIEW Driver Guide is the reference place for details on any of test and automation VIs or custom step types.



6.1 ISL COMMUNICATIONS

A connection must be established with the ISL instrumentation hardware before issuing any subsequent commands or requests.

The ISL_open.vi is used for opening a connection to the ISL hardware. This VI creates an ISL Handle reference that is a required input of almost every other VI in the VI Tree.



When opening a connection to the hardware, an FPGA bitfile is also downloaded. The serial number input to this VI is optional and if left unwired or empty, the first ISL device found will be opened and an ISL Handle reference returned.

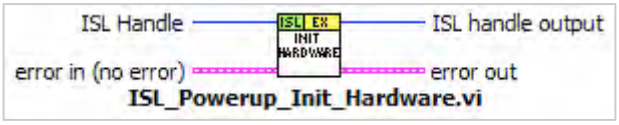
Every LabVIEW application that needs access to the ISL hardware should start with this VI and conclude with a call to the ISL_close.vi.

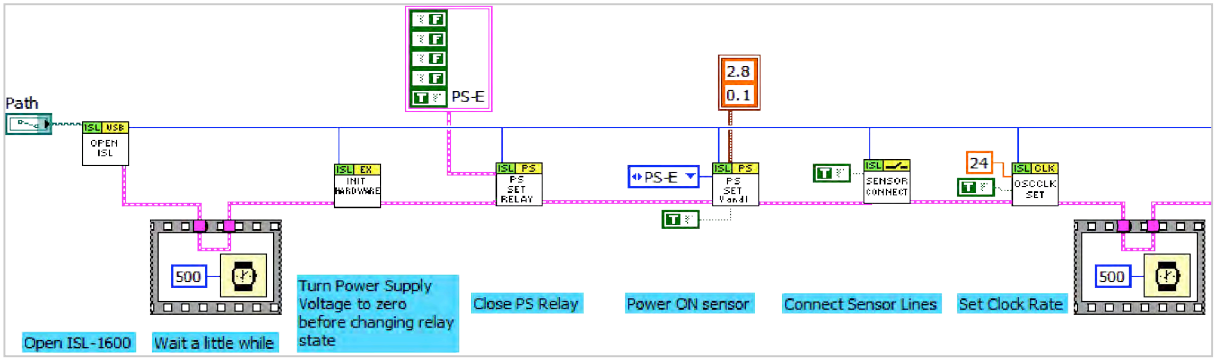
Once the ISL_open.vi has been called successfully and a valid ISL Handle has been acquired, communications with the ISL hardware can take place.

6.2 INITIALIZING THE ISL HARDWARE

The ISL instrumentation hardware should be initialized at the beginning of any application or whenever the power to the ISL hardware is first applied. The ISL_Powerup_Init_Hardware.vi must be called, which contains a collection of calls to put the instrument in a proper and safe startup state

This VI first resets the FPGA and then sets all power supplies to zero voltage output and opens the mechanical relay connection to the image sensor. The on-board oscillator is also turned off and disconnected from the image sensor. All of the FET switches are opened to completely disconnect the image sensor from the ISL hardware electronics. This VI also sets the default image capture settings as well as configures some lower level hardware settings that are required for proper operation.





6.3 LABVIEW CODE EXAMPLE

The LabVIEW example code shown above depicts a typical LabVIEW application startup sequence. The sequence starts with the mandatory calls to ISL_Open.vi and ISL_Powerup_Init_Hardware.vi. After these initial calls are complete, any of the other ISL Tree VIs can be called. In this example, power supply E is then enabled and the voltage setpoint is set to 2.8 V with a 0.1A current limit. After that, the sensor signal FETs are closed to connect the image sensor to the ISL instrument. To complete this example, the on-board oscillator is then set to 24 MHz.

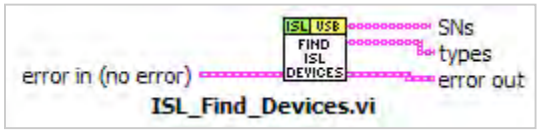
The Driver VIs are broken up into functional areas and are fully documented in the sections that follow.

6.4 LabVIEW Driver VI Function Reference

6.4.1 ISL Communications VIs

6.4.1.1 ISL_Find_Devices.vi

This VI can be called to determine the serial numbers and device types of any connected Image Sensor Lab USB device. This VI is typically called at the very beginning of an application program and does not require a valid ISL Handle before calling it.



Controls and Indicators

SNs A string array containing the serial numbers of the detected USB ISL devices.

string serial number string

types A string array containing the FPGA device types associated with the corresponding serial number from the **SNs** output.

string device type string

6.4.1.2 ISL_Open.vi

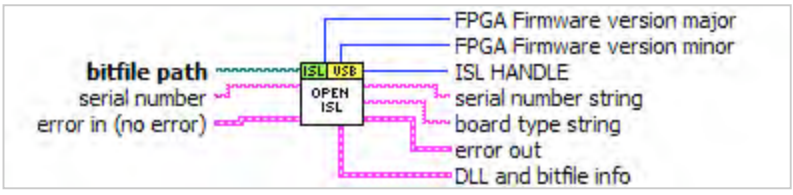
A connection must be established with the ISL instrumentation before issuing any subsequent commands or requests.

The ISL_open.vi is used for opening a connection to the ISL hardware. This VI creates an ISL Handle reference (ISL HANDLE) that is a required input of almost every other VI in the VI Tree.

When opening a connection to the hardware, an FPGA bit file is also downloaded. The bit file path control is a required input. The serial number input to this VI is optional and if left unwired or empty, the first ISL device found will be opened and an ISL Handle reference returned.

Every LabVIEW application that needs access to the ISL hardware should start with this VI and conclude with a call to the ISL_close.vi

Once the ISL_open.vi has been called successfully and a valid ISL Handle acquired, communications with the ISL hardware can take place.



Controls and Indicators

serial number The serial number input specifies the serial number of the ISL device to open. This input is not mandatory and the first detected ISL device is opened if this input is empty or not connected. Use this input if multiple ISL devices are connected to the same host computer.

bitfile path (REQUIRED) The absolute path to the ISL bit file used by the FPGA. This file path should point to the isl_3010.bit or isl_3010X2.bit file located in the root directory of the ISL Drivers.

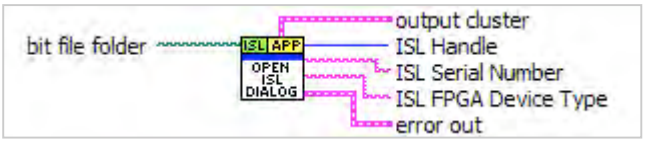
ISL HANDLE A reference for the opened device used by many of the VIs in the ISL Driver package.

Continued

- **board type string** The board type string output contains the FPGA type of the opened device.
- **serial number string** The serial number of the opened device.
- **FPGA Firmware version major** The major version number of the FPGA Firmware.
- **FPGA Firmware version minor** The minor version number of the FPGA Firmware.
- **DLL and bit file info** Summary information about the actual bit file and DLL used during the Open function.
  - **FPGA bit file path** The Path to the bit file actually downloaded during the Open function.
  - **bit file bytes** The size of the bit file in bytes.
  - **bit file LRC** The LRC checksum for the bit file used during the Open function.
  - **DLL Version Date** The DLL Version date embedded within the DLL file.
  - **DLL Version Time** The DLL Version time embedded within the DLL file.
  - **FPGA Version** The FPGA Firmware Version String.

6.4.1.3 Open_ISL_Device_for_App.vi

This VI launches a dialog box displaying a list of all the connected ISL devices. You can select one of the devices from this list then press the OK button to complete the device opening process. This VI outputs a valid ISL Handle reference to the newly opened device, and also outputs the devices type and serial number. An additional output cluster reports details about the FPGA bit file and DLL currently being used.

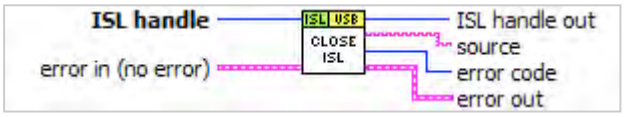


Controls and Indicators

- **bit file folder** The absolute path of the directory containing the ISL bit file. This is typically the root directory of the ISL LabVIEW drivers. The default bit file is named isl_3010.bit or isl_3010_2X.bit.
- **ISL Handle** A reference to the newly opened device selected by the user of this dialog window.
- **ISL Serial Number** The serial number of the device selected by the user of this dialog window.
- **ISL FPGA Device Type** The FPGA device type of the newly opened device selected by the user of this dialog window.
- **DLL and bitfile info** Summary information about the actual bit file and DLL used during the Open function.
  - **FPGA bit file path** Absolute path of the bit file used.
  - **bit file bytes** The size of the bit file in bytes.
  - **bit file LRC** The calculated LRC of the bit file.
  - **DLL Version Date** The DLL version date of the okTIMS.dll.
  - **DLL Version Time** The DLL version time of the okTIMS.dll
  - **FPGA Version** The version string of the downloaded bit file.

6.4.1.4 ISL_Close.vi

This VI closes the open reference (ISL_HANDLE) to the ISL hardware. The ISL_close.vi should be called at the end of every application when the ISL Handle is no longer needed.



Controls and Indicators

ISL handle (REQUIRED) The reference used to communicate with a device. Use ISL_open.vi to obtain a valid ISL Handle.

ISL handle out A copy of the ISL Handle input.

error code A copy of the error code contained in the standard error out cluster.

source The **source** string describes the origin of the error or warning and is a copy of the standard error out cluster source element.

6.4.2 ISL Sensor Communications VIs

6.4.2.1 ISL_Sensor_Comm_Type.vi

This VI must be called before attempting any sensor communications. This VI is used to specify the sensor communications type - I2C or SPI. By default the sensor communications type is set to I2C.

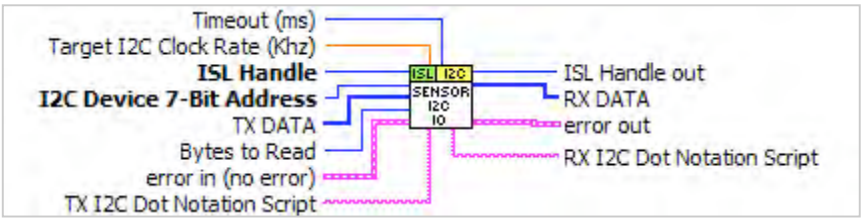
Controls and Indicators

ISL Handle (REQUIRED) The reference used to communicate with a device. Use ISL_open.vi to obtain a valid ISL Handle.

Sensor Comm Type Enumerated type – Used to specify the image sensor communications method. 0 = I2C, 1 = SPI.

6.4.2.2 ISL_Sensor_I2C_IO.vi

This VI is used to send and receive bytes to and from the sensor using I2C communications. Either a byte array (the TX DATA input) or a hex dot notation script (TX I2C Dot Notation Script input) is used, but not both. The byte array TX DATA input is only used if the TX I2C Dot Notation Script input is empty or unwired.



Controls and Indicators

ISL Handle (REQUIRED) The reference used to communicate with a device. Use ISL_open.vi to obtain a valid ISL Handle.

I2C Device 7-Bit Address (REQUIRED) The I2C Address of the image sensor.

TX DATA Byte array to be written to the image sensor. When writing a sensor register value this byte array contains the register address and value.

Bytes to Read When writing to an image sensor register set this value to 0. When reading an image sensor register value set this input to the width in bytes of the sensor register.

Target I2C Clock Rate (KHz) Clock rate used for I2C communications with the image sensor. This control is set to 100 KHz by default, but can be set to any desired I2C clock rate up to 10 MHz.

Timeout (ms) Timeout used during I2C communications with the image sensor. If a response is not received from the image sensor within this timeout setting, an error is generated.

TX I2C Dot Notation Script This control contains the data, in hex dot script notation, that is written to the image sensor.

Leave this control empty if using the alternate TX Data byte array input.

The hex dot notation script is a simple text string with two hex characters representing each byte and a period separating each byte. (example: F0.00.01)

RX DATA Byte array - received from the image sensor.

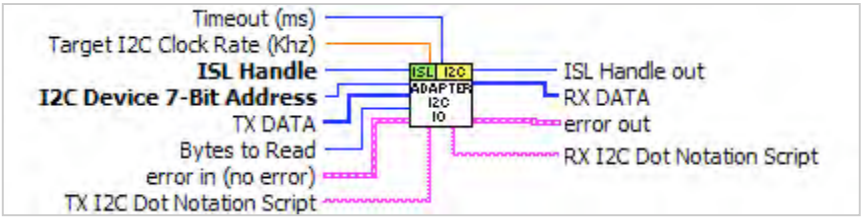
ISL Handle out ISL Device Handle

RX I2C Dot Notation Script This indicator contains the sensor I2C response, in hex dot script notation.

The hex dot notation script is a simple text string with two hex characters representing each byte and a period separating each byte. (example: F0.00.01)

6.4.2.3 ISL_Adapter_I2C_IO.vi

This VI is used to send and receive bytes to and from an ISL adapter board I2C device using I2C communications. Either a byte array (the TX DATA input) or a hex dot notation script (TX I2C Dot Notation Script input) is used, but not both. The byte array TX DATA input is only used if the TX I2C Dot Notation Script input is empty or unwired.



Controls and Indicators

ISL Handle The reference used to communicate with a device. Use ISL_open.vi to obtain a valid ISL Handle.

I2C Device 7-Bit Address - Address of the I2C slave device.

TX DATA Byte array to be written to the I2C slave device.

Bytes to Read When writing to an image sensor register set this value to 0. When reading an image sensor register value set this input to the width in bytes of the sensor register.

Target I2C Clock Rate (KHz) Clock rate used for I2C communications with the image sensor. This control is set to 100 KHz by default but can be set to any desired I2C clock rate up to 10 MHz.

Timeout (ms) Timeout used during I2C communications with the image sensor. If a response is not received from the image sensor within this timeout setting, an error is generated.

TX I2C Dot Notation Script This control contains the data, in hex dot script notation, that is written to the image sensor.

Leave this control empty if using the alternate TX Data byte array input.

The hex dot notation script is a simple text string with two hex characters representing each byte and a period separating each byte. (example: F0.00.01)

RX DATA Byte array received from the image sensor.

ISL Handle out ISL Device Handle

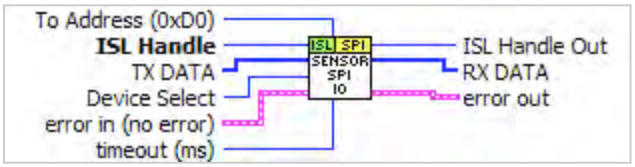
RX I2C Dot Notation Script This indicator contains the sensor I2C response, in hex dot script notation.

The hex dot notation script is a simple text string with two hex characters representing each byte and a period separating each byte. (example: F0.00.01)

6.4.2.4 ISL_Sensor_SPI_IO.vi

This VI is used to send and receive bytes to and from an image sensor or device using SPI communications. Either a byte array (the TX DATA input) or a hex dot notation script (TX I2C Dot Notation Script input) is used, but not both. The byte array TX DATA input is only used if the TX I2C Dot Notation Script input is empty or unwired. This VI performs the chip select operation, a data I/O operations up to 255 bytes, then a chip deselect operation.

ISL_Sensor_SPI_IO.vi



Controls and Indicators

TX DATA Data byte array to send during SPI communications transactions.

timeout (ms) timeout specifies the amount of time (in milliseconds) that the LabVIEW driver will wait after each byte is received from a ISL unit.

If the next byte is not received within the specified timeout, the IO function will return an error and any byte received up to that point.

ISL Handle The ISL Reference used to communicate with a device.

To Address (0xD0) To Address specifies the TIMS processor address.

The default value of 0xD0 refers to the main processor on any TIMS unit.

Device Select

RX DATA Data byte array received during the SPI communications transaction.

ISL Handle Out The TIMS Reference used to communicate with a device.

6.4.3 ISL Clock VIs

6.4.3.1 ISL_OSCCLK_Set.vi

This VI is used to control the on-board programmable oscillator. The programmable oscillator is often used as a Master Clock and routed to the clock input of the image sensor. Both the target clock frequency (in MHz) and an enable (on/off) input are required. The actual on-board programmable oscillator set points are somewhat granular and because of this the actual output setpoint frequency is output.

ISL_OSCCLK_Set.vi

Controls and Indicators

ISL Handle The TIMS Reference used to communicate with a device.

Target Freq (MHz) This required input sets the on-board oscillator frequency setpoint (in MHz).

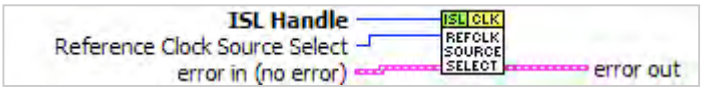
On/Off This required input enables or disables the on-board oscillator output.

Output Freq (mhz) The actual frequency setpoint (in MHz) of the on-board oscillator.

6.4.3.2 ISL_Reference_Clock_Select.vi

This VI is used to select which internal clock will be used as the Master Reference Clock and routed to the image sensor. The default clock selection is the on-board oscillator clock. An optional FPGA based clock can be selected, or the Master Reference Clock can be disabled and turned off.

ISL_Reference_Clock_Select.vi



Controls and Indicators

ISL Handle The ISL Reference used to communicate with a device. Use ISL_open.vi to obtain a valid ISL Handle.

Reference Clock Source Select Reference Clock Source Select controls which internal clock source is used as the Master Reference Clock and routed to the image sensor.

0 = On-board Oscillator

1 = FPGA base clock

2 = OFF and Disabled

6.4.3.3 ISL_Reference_Clock_Select.vi

This VI is used to select from the available FPGA based clock sources.

0 = Oscillator Clock Doubling.

Note: Use the ISL_Reference_Clock_Select.vi to choose between the on-board oscillator and the FPGA based clock, for use as the Master Reference Clock routed to the image sensor.

ISL_FPGA_Clock_Select.vi

Controls and Indicators

ISL Handle The ISL Reference used to communicate with a device.

FPGA Clock Select FPGA Clock Select is used to select from the available FPGA based clock sources.

Note: Use the ISL_Reference_Clock_Select.vi to choose between the on-board oscillator and the FPGA based clock, for use as the Master Reference Clock routed to the image sensor.

6.4.3.4 ISL_OSC_Clock_Get.vi

Returns the independently measured on-board oscillator frequency (in MHz).

ISL_OSC_Clock_Get.vi

Controls and Indicators

ISL Handle The ISL Reference used to communicate with a device.

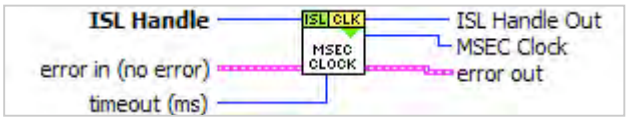
ISL Handle Output The ISL Reference used to communicate with a device.

OSC Clock Frequency The independently measured on-board oscillator frequency (in MHz).

6.4.3.5 ISL_MSEC_Clock_Get.vi

Returns the value of the on-board millisecond timer within the ISL device.

ISL_MSEC_Clock_Get.vi



Controls and Indicators

timeout (ms) timeout specifies the amount of time (in milliseconds) that the LabVIEW driver will wait after each byte is received from a ISL unit.

If the next byte is not received within the specified timeout, the IO function will return an error and any byte received up to that point.

ISL Handle The ISL Reference used to communicate with a device.

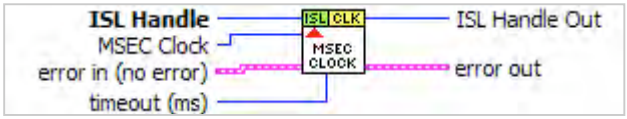
ISL Handle Out The ISL Reference used to communicate with a device.

MSEC Clock The value of the on-board millisecond timer with the ISL device.

6.4.3.6 ISL_MSEC_Clock_Set.vi

Sets the value of the on-board millisecond timer within the ISL device.

ISL_MSEC_Clock_Set.vi



Controls and Indicators

timeout (ms) timeout specifies the amount of time (in milliseconds) that the LabVIEW driver will wait after each byte is received from an ISL unit.

If the next byte is not received within the specified timeout, the IO function will return an error and any byte received up to that point.

ISL Handle The ISL Reference used to communicate with a device.

MSEC Clock The value written to the on-board millisecond timer as the current value. The millisecond time will begin counting up immediately from this new value.

ISL Handle Out The ISL Reference used to communicate with a device.

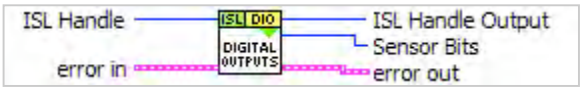
6.4.4 ISL Digital IO VIs

6.4.4.1 ISL_Get_Sensor_Bit_Output.vi

This VI returns the current state of the digital output control lines of the ISL device.

ISL_Get_Sensor_Bit_Output.vi

Connector Pane



Controls and Indicators

ISL Handle The ISL Reference used to communicate with a device.

Sensor Bits ISL device digital output control bit values. Each bit of this value corresponds to the output state of the associated ISL digital output line.

ISL Handle Output The ISL Reference used to communicate with a device.

6.4.4.2 ISL_Get_Sensor_Bit_Inputs.vi

This VI returns the current state of the digital input lines of the ISL device. In addition to the current state of each input line, a "change detector" reports on any input bits that have changed state since the last read.

ISL_Get_Sensor_Bit_Intputs.vi

Controls and Indicators

ISL Handle The ISL Reference used to communicate with a device.

Sensor Bit Inputs A cluster containing the current value of each bit and an indication of a transition of any bit since the last read.

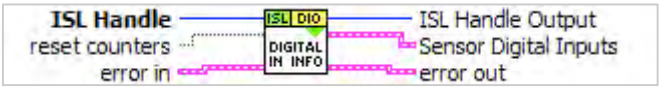
- **Val** The current value of the digital input bits
- **State** The current latched value of the "Q" output of the first column of flip-flops.
- **changes** Indicates the changes that have occurred since the last read.

ISL Handle Output The ISL Reference used to communicate with a device.

6.4.4.3 ISL_Get_Inptut_Bit_Info.vi

This VI returns the current state of the digital input lines of the ISL device. In addition to the current state of each input line, a change detector reports on each bit input. This higher-level utility VI calls the ISL_Get_Sensor_Bit_Inputs.vi.

ISL_Get_Input_Bit_Info.vi



Controls and Indicators

- **ISL Handle** The ISL Reference used to communicate with a device.
- **reset counters** Resets the accumulated count of transitions on each bit input.
- **ISL Handle Output** The ISL Reference used to communicate with a device.
- **Sensor Digital Inputs** ISL Digital Input State and Transition Information.
  - **ISL Digital Intputs** ISL Digital Input State and Transition Information.
    - **Bin-0** The current state of ISL digital input bit 0
    - **Bin-1** The current state of ISL digital input bit 1
    - **Bin-2** The current state of ISL digital input bit 2
    - **Bin-3** The current state of ISL digital input bit 3.
    - **Bin0 counter** ISL digital input bit 0 change detection counter.
    - **Bin1 counter** ISL digital input bit 1 change detection counter.
    - **Bin2 counter** ISL digital input bit 2 change detection counter.
    - **Bin3 counter** ISL digital input bit 3 change detection counter.

6.4.4.4 ISL_App_Set_Sensor_Bit_Outputs.vi

This VI sets the state of the ISL digital output control lines.

ISL_APP_Set_Sensor_Bit_Outputs.vi

Controls and Indicators

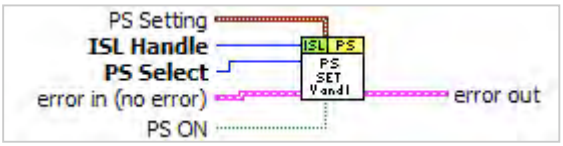
- **ISL Handle** The ISL Reference used to communicate with a device.
- **Bout Config** Cluster containing the individual ISL digital output line controls.
  - **Bout-0** ISL Digital Output Line 0 control
  - **Bout-1** ISL Digital Output Line 1 control
  - **Bout-2** ISL Digital Output Line 2 control
  - **Bout-3** ISL Digital Output Line 3 control

6.4.5 ISL Power Supply VIs

6.4.5.1 ISL_PS_Set_V_and_I.vi

This VI sets the voltage output setpoint and current limit value for a specified power supply within the ISL device.

ISL_PS_Set_V_and_I.vi



Controls and Indicators

ISL Handle The ISL Reference used to communicate with a device.

PS Setting Cluster containing the specified power supplies voltage output setpoint and current limit setting.

Voltage Output The specified power supplies voltage output setpoint.

Current Limit The specified power supplies current limit setting.

PS Select Power Supply Selector - selects which power supply within the ISL device to program.

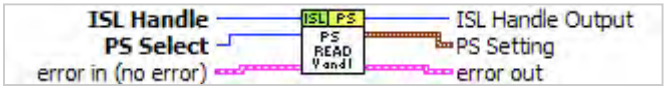
0 = PSA 1 = PSB 2 = PSC 3 = PSD 4 = PSE

PS ON Power Supply Enable - True enables the specified power supply. False disables the specified power-supply and sets the output to 0V.

6.4.5.2 ISL_PS_Read_V_and_I.vi

This VI measures the voltage output and current draw from the specified power supply.

ISL_PS_Read_V_and_I.vi



Controls and Indicators

PS Select Power Supply Selector - selects which power supply within the ISL device to program.

0 = PSA 1 = PSB 2 = PSC 3 = PSD 4 = PSE

ISL Handle The ISL Reference used to communicate with a device.

PS Setting Cluster containing the specified power supplies measured output voltage and measure current draw.

Voltage Output The specified power supplies measured output voltage.

Current Limit The specified power supplies measure current draw.

ISL Handle Output The ISL Reference used to communicate with a device.

6.4.5.3 ISL_PS_Relay_Set.vi

This VI sets the state of the relays the connect/isolate the power supplies to/from the image sensor.

ISL_PS_Relay_Set.vi

Controls and Indicators

ISL Handle The ISL Reference used to communicate with a device.

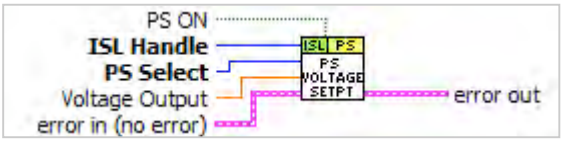
PS Output Relay Cluster containing the individual power supply relay controls.

- **PS-A** Power Supply A (PSA) relay state control.
- **PS-B** Power Supply B (PSB) relay state control.
- **PS-C** Power Supply C (PSC) relay state control.
- **PS-D** Power Supply D (PSD) relay state control.
- **PS-E** Power Supply E (PSE) relay state control.

6.4.5.4 ISL_PS_V_Set.vi

This VI sets the specified power supply voltage output setpoint.

ISL_PS_V_Set.vi



Controls and Indicators

ISL Handle The TIMS Reference used to communicate with a device.

PS Select Power Supply Selector - selects which power supply within the ISL device to program.

0 = PSA 1 = PSB 2 = PSC 3 = PSD 4 = PSE

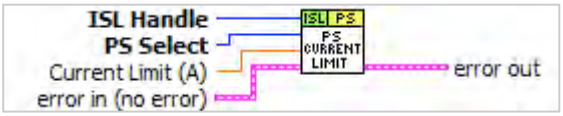
PS ON Power Supply Enable - True enables the specified power supply.
False disables the specified power supply and sets the output to 0V.

Voltage Output The specified power supplies voltage output setpoint.

6.4.5.5 ISL_PS_I_Set.vi

This VI sets the specified power supplies current limit setting.

ISL_PS_Set_Current_Limit.vi



Controls and Indicators

ISL Handle The ISL Reference used to communicate with a device.

PS Select Power Supply Selector - selects which power supply within the ISL device to program.

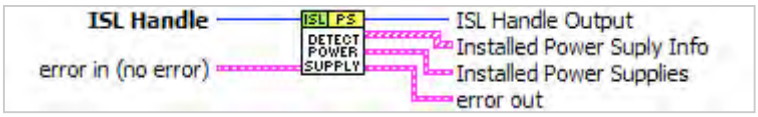
0 = PSA 1 = PSB 2 = PSC 3 = PSD 4 = PSE

Current Limit (A) The Current Limit (A) control sets the current limit of the specified power supply.

6.4.5.6 ISL_PS_Detect.vi

This VI detects power supply modules plugged into the ISL device.

ISL_PS_Detect.vi



Controls and Indicators

ISL Handle The ISL Reference used to communicate with a device.

ISL Handle Output The ISL Reference used to communicate with a device.

Installed Power Supply Info Array of clusters of detected power supply information. Each detected power supply in the connected ISL device returns a cluster of information including type, location, serial number, etc.

Detected Power Supply Info Cluster

Installed

power supply location Power supply location within the ISL device.

power supply type Power supply type number.

0 = ISL Type 1 supply (0 - 4 V 200mA)
1 = ISL Type 2 supply (0 - 4V 200mA High Res.)
2 = ISL Type 3 supply (5V Utility Supply)

Model ID Detected power supply model ID information

Model No.

Description 1

PCB Assy ID Detected power supply PCB assembly ID information

Part No.

Serial No.

Continued

Misc Info

Calibration Time Stamp Timestamp of the last calibration of this ISL power supply.

Installed Power Supplies Boolean cluster representing installed status of each power supply location.

- **PSA**
- **PSB**
- **PSC**
- **PSD**
- **PSE**

6.4.6 ISL Sensor Connections

6.4.6.1 ISL_Sensor_Connect.vi

This VI closes or opens the FET switch connection between the sensor communications, digital IO, data, and sync lines and the ISL device. Set this input to F (Open) before removing or installing an image sensor, or to isolate the image sensor from any of the electronic components of the ISL device.

ISL_Sensor_Connect.vi

Controls and Indicators

ISL Handle The ISL Reference used to communicate with a device.

Connect (F) Controls the FET Switch connection between the image sensor communications, digital IO, data, and sync lines and the ISL device electronics.

6.4.6.2 ISL_Enable_Sensor_Comm.vi

This VI closes or opens the FET switch connection between the sensor communications (I2C/SPI) lines and the ISL device.

The ISL_Sensor_Connect.vi should normally be used for this function, as it controls the FET switch connections to most of the sensor signals in one call.

ISL_Enable_Sensor_Comm.vi

Controls and Indicators

ISL Handle The ISL Reference used to communicate with a device.

Enable Sensor Comm Closes or opens the FET switch connection between the sensor communications (I2C/SPI) lines and the ISL device.

6.4.6.3 ISL_Enable_BitIO.vi

This VI closes or opens the FET switch connection between the sensor lines and the ISL device digital IO lines.

The ISL_Sensor_Connect.vi should normally be used for this function, as it controls the FET switch connections to most of the sensor signals in one call.

ISL_Enable_BitIO.vi

Controls and Indicators

ISL Handle The ISL Reference used to communicate with a device.

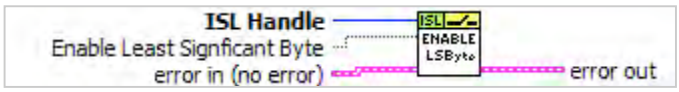
Enable BitIO Closes or opens the FET switch connection between the sensor and the ISL device digital IO lines.

6.4.6.4 ISL_Enable_LSByte.vi

This VI closes or opens the FET switch connection between the sensor and the ISL device lower 8 data lines.

The ISL_Sensor_Connect.vi should normally be used for this function, as it controls the FET switch connections to most of the sensor signals in one call.

ISL_Enable_LSByte.vi



Controls and Indicators

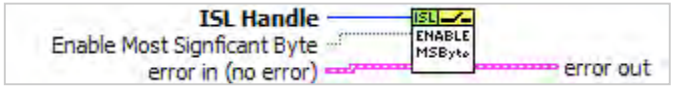
ISL Handle The ISL Reference used to communicate with a device.

Enable Least Significant Byte This VI closes or opens the FET switch connection between the sensor and the ISL device lower 8 data lines.

6.4.6.5 ISL_Enable_MSByte.vi

This VI closes or opens the FET switch connection between the sensor and the ISL device upper 8 data lines.

The ISL_Sensor_Connect.vi should normally be used for this function, as it controls the FET switch connections to most of the sensor signals in one call.



ISL_Enable_MSByte.vi

Controls and Indicators

ISL Handle The ISL Reference used to communicate with a device.

Enable Most Significant Byte This VI closes or opens the FET switch connection between the sensor and the ISL device upper 8 data lines.

6.4.6.6 ISL_Enable_Sync.vi

This VI closes or opens the FET switch connection between the sensor and the ISL device sync data lines.

The ISL_Sensor_Connect.vi should normally be used for this function, as it controls the FET switch connections to most of the sensor signals in one call.

ISL_Enable_Sync.vi

Controls and Indicators

ISL Handle The ISL Reference used to communicate with a device.

Enable Sync This VI closes or opens the FET switch connection between the sensor and the ISL device sync lines

6.4.7 ISL Event Counter

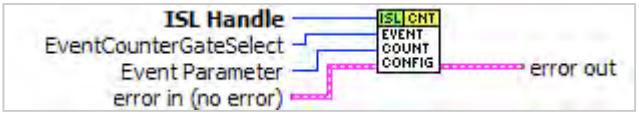
6.4.7.1 ISL_Event_Counter_Config.vi

This VI configures the ISL event counter. The ISL event counter is FPGA logic that can measure timing information from any of the signals connected to the ISL device.

NOTE: This is a legacy function! This VI has been replaced by the extended timer/counter functionality contained in the ISL_Event_Counter2.vi.

ISL_Event_Counter_Config.vi

Connector Pane



Controls and Indicators

ISL Handle The ISL Reference used to communicate with a device.

EventCounterGateSelect EventCounterGateSelect is the timer/counter gate control and measurements can be taken over a Frame Window or a one second time window.

0 = Frame Window
1 = 1 sec. timer

Event Parameter Event Parameter selects which image sensor connection is counted.

0= Data[0]	7= Data[7]	13= Data[13]	19= XSYNC2
1= Data[1]	8= Data[8]	14= Data[14]	20= Din0
2= Data[2]	9= Data[9]	15= Data[15]	21= Din1
3= Data[3]	10= Data[10]	16= VSYNC	22= Din2
4= Data[4]	11= Data[11]	17= XSYNC0	23= Oscillator Clock
5= Data[5]	12= Data[12]	18= XSYNC1	24= Capture Clock
6= Data[6]			

6.4.7.2 ISL_Event_Counter_Arm.vi

This VI arms the ISL timer/counter function and starts the measurement process.

ISL_Event_Counter_Arm.vi

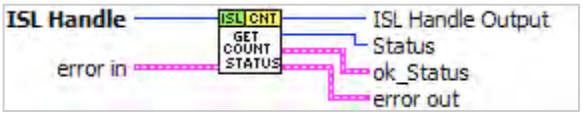
Controls and Indicators

U32 **ISL Handle** The ISL Reference used to communicate with a device.

6.4.7.3 ISL_Get_Counter_and_Capture_Status.vi

This VI returns the status of the FPGA event counter and capture functions.

ISL_Get_Counter_and_Capture_Status.vi



Controls and Indicators

U32 **ISL Handle** The ISL Reference used to communicate with a device.

I32 **Status** Event counter and capture status

Bit values:

0 - Capture BUSY
1 - Capture DONE
2 - Event Counter ARMED
3 - Event Counter ACTIVE
4 - Event Counter DONE

U32 **ISL Handle Output** The ISL Reference used to communicate with a device.

ok_Status Event counter and capture status

TF **Capture BUSY**

TF **Capture DONE**

TF **Counter ARMED**

TF **Counter ACTIVE**

TF **Counter DONE**

6.4.7.4 ISL_Get_Sensor_Event_Count.vi

This VI reads the timer counter results.

Be sure to call the ISL_Get_Coutner_and_Capture_Status.vi and ensure that it returns the status bit for Event Counter DONE set, before calling this vi.

ISL_Get_Sensor_Event_Count.vi

Controls and Indicators

ISL Handle The ISL Reference used to communicate with a device.

Count The Event Counter measured count

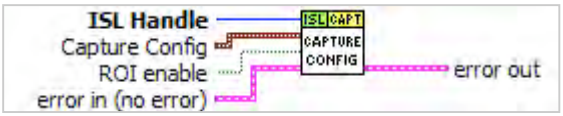
ISL Handle Output The ISL Reference used to communicate with a device.

6.4.8 ISL Image Capture

6.4.8.1 ISL_Capture_Config.vi

This VI writes the image capture configuration parameters to the ISL device. The clocking and synchronization signal edge and polarity settings can be specified here.

ISL_Capture_Config_Set.vi



Controls and Indicators

ISL Handle The TIMS Reference used to communicate with a device.

Capture Config The clocking and synchronization signal edge and polarity settings used for image capture can be specified here.

Pixel Data Store Config The Pixel Data Store Config control is used to specify the data width of the image capture.

0 = 16-bit word (capture all 16-bits)

1 = 8-bit byte (capture only the lower 8-bits)

2 = 8-bit byte (capture only the upper 8-bits)

3 = 16-bit word (word values are continuous count test pattern)

FRAME Start Condition The Frame Start Condition control is used to specify which edge of the VSYNC signal is to be used to define the Frame Start condition.

0 = VSYNC rising edge

1 = VSYNC falling edge

FRAME End Condition The Frame End Condition control is used to specify which edge of the VSYNC signal is to be used to define the Frame End condition.

0 = VSYNC rising edge

1 = VSYNC falling edge

Continued

XSYNC0 Config XSYNC0 is an optional synchronization signal input that is usually tied to the

Line or Row Valid signal from the image sensor.

0 = disabled

1 = enabled active high

2 = enabled active low

XSYNC1 Config XSYNC1 is an optional synchronization signal input used for image capture.

0 = disabled

1 = enabled active high

2 = enabled active low

XSYNC2 Config XSYNC2 is an optional synchronization signal input used for image capture.

0 = disabled

1 = enabled active high

2 = enabled active low

Clock Select Clock Select is used to specify the source of the Capture Clock (CAPCLK).

0 = PIXCLK - The PIXCLK coming from the image sensor is used for image capture.

1 = REFCLK - The Master Reference clock going from the ISL to the image sensor is also used for image capture.

Clock Edge

ROI enable The ROI Enable bit is use to enable/disable the ROI capture function.

T = Capture only the image data within the defined ROI

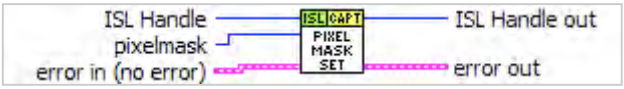
F = Capture ALL image data

6.4.8.2 ISL_Pixel_Mask_Set.vi

This VI is used to set the ISL input pixel bit mask.

A logical “AND” is performed between the 16-bit mask and each captured word, so that specific bits of the 16-bit input can be enabled or disabled. This is especially useful if some of the higher level input bits are not used and left unwired. This function can be used to force those upper unused bits to zero.

ISL_Pixel_Mask_Set.vi



Controls and Indicators

ISL Handle The ISL Reference used to communicate with a device.

pixelmask The pixelmask is a 16-bit value that is ANDed with the 16-bit image data input of the ISL device. Set the bits of this mask to zero to force that bit in the captured data to be zero. Set the bits of this mask to one to enable that bit in the captured data.

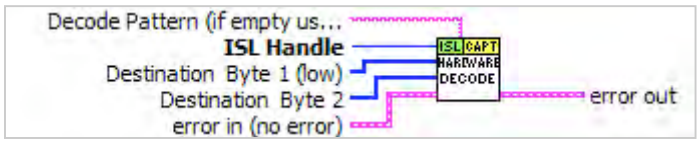
ISL Handle out The ISL Reference used to communicate with a device.

6.4.8.3 ISL_Hardware_Decode.vi

This VI is used to specify a hardware bit input remapping of the image data lines coming from the image sensor.

Either the Decode Pattern string input or the alternate Destination Byte clusters controls can be used.

ISL_Hardware_Decode_Set.vi



Controls and Indicators

ISL Handle The ISL Reference used to communicate with a device.

Destination Byte 1 (low) The Destination Byte 1 (low) cluster control is only used if the Decode Pattern string control is left empty.

This cluster control allows a user to select the source bit that should be mapped to a specific destination byte and bit. Each Destination Byte bit position has a drop-list containing each source bit.

bit setting

Destination Byte 2 The Destination Byte 2 (high) cluster control is only used if the Decode Pattern string control is left empty.

This cluster control allows a user to select the source bit that should be mapped to a specific destination byte and bit. Each Destination Byte bit position has a drop-list containing each source bit.

bit setting

Decode Pattern (if empty use other controls) Decode Pattern is a String input that controls the hardware remapping of the image data input lines to the ISL device.

Either this control is used, or if left empty, the alternate Destination Byte cluster controls are used.

The Decode Patter string must contain 16 hex characters, each character representing that particular bits position in the final data word.

For instance, for a straight through mapping, where each input bit is mapped to the same position in the final image data word as its position in the actual wiring, the Decode Pattern String should be set to:

FEDCBA9876543210

The character on the far right represents the signal connected to Bit 0 on the ISL connector. As shown in the example above, we have set that character in the Decode Pattern String to 0, implying that we want that data line to be mapped to bit 0 in the final destination word.

A Decode Pattern of:

0123456789ABCDEF

would reverse the order of the data lines connected to the ISL.

6.4.8.4 ISL_Consecutive_Capture_Config_Set.vi

This VI configures the number of consecutive images to capture for each commanded capture or snapshot. The number of images commanded must fit in the amount of on-board memory available in the ISL device.

ISL_Consecutive_Capture_Config_Set.vi

Controls and Indicators

ISL Handle The ISL Reference used to communicate with a device.

number of images The number of consecutive images to capture for each commanded capture or snapshot. The number of images commanded must fit in the amount of on-board memory available in the ISL device.

6.4.8.5 ISL_Capture_ROI_Config.vi

This VI is used to specify a Region of Interest (ROI) that is used during the image capture process. Only the pixel values within the ROI are captured into on-board memory for download to the host computer.

ISL_Capture_ROI_Config.vi

Controls and Indicators

ISL Handle ISL Device Handle

ROI coordinates The ROI coordinates of the capture region (Left, Top, Right, Bottom).

Numeric

ISL Handle Output ISL Device Handle

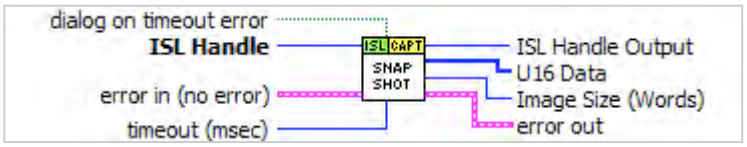
FPGA ROI description

Numeric

6.4.8.6 ISL_Snapshot.vi

This is an "all in one" capture that first resets the capture buffer to ensure a fresh image, waits until the CaptureComplete flag is set, and then downloads the image data.

ISL_Snapshot.vi



Controls and Indicators

ISL Handle ISL Device Handle

timeout (msec) Timeout used during image capture.

dialog on timeout error This input allows an optional error dialog to appear when a capture timeout occurs.

U16 Data Image data word array.

U16 Data

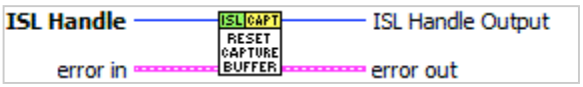
ISL Handle Output ISL Device Handle

Image Size (Words) TIMS Device Handle

6.4.8.7 _Reset_Capture_Buffer.vi

This VI resets the image capture buffer and forces the start of a new frame capture.

ISL_Reset_Capture_Buffer.vi



Controls and Indicators

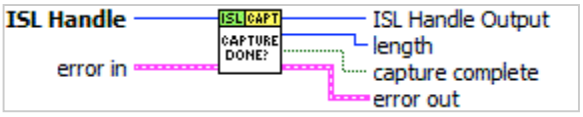
ISL Handle The ISL Reference used to communicate with a device.

ISL Handle Output ISL Device Handle

6.4.8.8 ISL_Capture_Complete.vi

This VI returns that status of the current image frame capture. When a capture is complete this VI will return the length of the capture (in 16-bit words) and a Boolean capture complete status bit.

ISL_Capture_Complete.vi



Controls and Indicators

ISL Handle The ISL Reference used to communicate with a device.

ISL Handle Output ISIL Device Handle

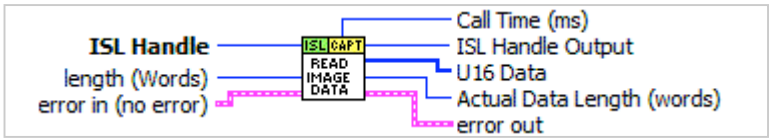
length The number of words captured in the current image capture.

capture complete Capture complete status bit.

6.4.8.9 ISL_Read_Image_Data.vi

This VI is used to read the last captured image data from the ISL devices on-board memory. The "length" output from the "ISL_Capture_Complete.vi" is typically wired to the length (in 16-bit words) input of this VI.

ISL_Read_Image_Data.vi



Controls and Indicators

length (Words) The number of words to read from the ISL device on-board memory.

ISL Handle The ISL Reference used to communicate with a device.

U16 Data Image Data returned as a word (U16) array.

U16 Data

ISL Handle Output ISL Device Handle

Call Time (ms) Measured time for this call in milliseconds.

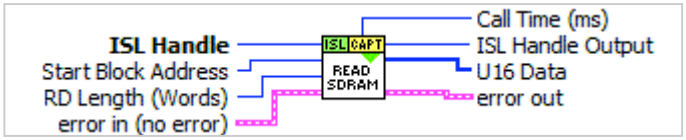
Actual Data Length (words) The actual length of data read out from the ISL Device memory. This value is typically larger than the number of words in the image because ISL device memory must be accessed in 512 byte blocks.

6.4.8.10 ISL_Read_SDRAM.vi

This VI is used to read the on-board memory within the ISL Device. This is a generic VI that allows access to any part of the on-board memory.

Typically the ISL_Read_Image_Data.vi is used to read image data from the ISL Device, but this VI is made available to enable memory testing of the ISL on-board memory.

ISL_Read_SDRAM.vi



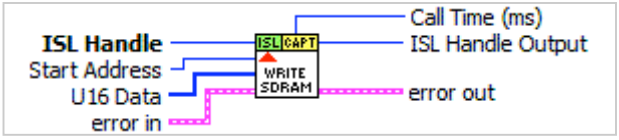
Controls and Indicators

- **RD Length (Words)** Number of words to read from the ISL devices on-board memory.
- **ISL Handle** ISL Device Handle
- **Start Block Address** The starting block memory address of the ISL device on-board memory.
- **U16 Data** ISL device on-board memory values.
  - **U16 Data**
- **ISL Handle Output** ISL Device Handle
- **Call Time (ms)** The total time taken to execute this code module (in milliseconds).

6.4.8.11 ISL_Write_SDRAM.vi

This VI is used to write to the on-board memory within the ISL device. This is a generic VI that allows access to any part of the on-board memory, and is made available for memory testing.

ISL_Write_SDRAM.vi



Controls and Indicators

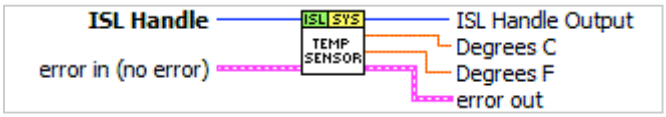
- **Start Address** The start address of the memory read function.
- **U16 Data** U16 word values to write to the ISL devices on-board memory.
  - **U16 Data**
- **ISL Handle** ISL Device Handle
- **ISL Handle Output** ISL Device Handle
- **Call Time (ms)** The total time taken to execute this code module (in milliseconds).

6.4.9 ISL System VIs

6.4.9.1 ISL_Read_Temperature_Sensor.vi

This VI reads the on-board temperature sensor within the ISL device.

ISL_read_temperature_sensor.vi



Controls and Indicators

ISL Handle ISL Device Handle

ISL Handle Output ISL Device Handle

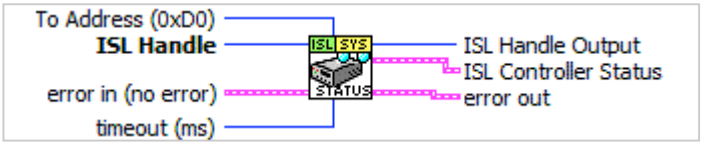
Degrees F Temperature, in degrees F, of the ISL device on-board temperature sensor.

Degrees C Temperature, in degrees C, of the ISL device on-board temperature sensor.

6.4.9.2 ISL_Controller_Status.vi

Returns the Status Information of a TIMS device.

ISL_Controller_Status.vi



Controls and Indicators

timeout (ms) timeout specifies the amount of time (in milliseconds) that the LabVIEW driver will wait after each byte is received from a TIMS unit.

If the next byte is not received within the specified timeout, the IO function will return an error and any byte received up to that point.

ISL Handle The ISL Reference used to communicate with a device.

To Address (0xD0) To Address specifies the TIMS processor address.

The default value of 0xD0 refers to the main processor on any TIMS unit.

ISL Controller Status

General

SW Valid - Device Application SW Valid
1 = SW is valid.
0 = SW is NOT valid.

Unknown Code - Unknown Control Code Detected
1 = An unknown control code was detected since the last status query.
0 = No unknown control codes detected.
This bit is cleared following device status query.

TX Overflow

Continued

RX Timeout

RX LRC Error

(na5)

FLASH Prgm - Microcontroller FLASH Programming In-Progress
1 = FLASH programming is in-progress, SW forced invalid.
0 = FLASH programming operation is not in-progress.

Test Mode - Manufacturing Test Mode
1 = Test mode enabled.
0 = Test mode disabled.

Error

EE CRC - Microcontroller EEPROM CRC Error Detected
1 = An EE CRC error was detected since the last status query.
0 = No error.

This bit is cleared following device status query.

EE Write - Microcontroller EEPROM Write Error Detected
1 = An EE Write error was detected since the last status query.
0 = No error.

This bit is cleared following device status query.

EE Address - Microcontroller EEPROM Address Error Detected
1 = A Protected EE location was attempted to be addressed.
0 = No error.

This bit is cleared following device status query.

(na3)

FLASH Erase - Microcontroller FLASH Erase Error Detected
1 = A FLASH Erase error was detected since the last status query.
0 = No error.

This bit is cleared following device status query.

FLASH Write - Microcontroller FLASH Write Error Detected
1 = A FLASH Write error was detected since the last status query.
0 = No error.

This bit is cleared following device status query.

UART Framing - UART Framing Error Detected
1 = A UART Framing error was detected since the last status query.
0 = No error.

This bit is cleared following device status query.
Valid only for SIO interfaced TIMS devices.
Indicates the potential for baud rate incompatibility.

Continued

UART Overrun - UART Overrun Error Detected

1 = A UART Overrun error was detected since the last status query.

0 = No error.

This bit is cleared following device status query.
Valid only for SIO interfaced TIMS devices.
Indicates the potential for firmware unable to service incoming data stream.

Reset

BOR - Microcontroller Brown Out Reset

1 = BOR reset not detected.

0 = BOR reset has occurred since the last device status query. (<4.5V)

This bit is cleared following device status query.

POR - Microcontroller Power On Reset

1 = POR reset not detected.

0 = POR reset has occurred since the last device status query.

This bit is cleared following device status query.

PD - Microcontroller Power Down Reset

1 = PD reset not detected.

0 = PD reset has occurred since the last device status query.

This bit is cleared following device status query.

TO - Microcontroller Watchdog Timer Time Out Reset

1 = TO reset not detected.

0 = TO reset has occurred since the last device status query.

This bit is cleared following device status query.

RI - Microcontroller Reset Instruction Reset

1 = RI reset not detected.

0 = RI reset has occurred since the last device status query.

This bit is cleared following device status query.

(na5)

(na6)

RESET - Microcontroller Reset

1 = A Reset has occurred since the last device status query.

0 = No reset detected

This bit is cleared following device status query.

Continued

PC Stack

SP0

SP1

SP2

SP3

Continued

SP4

(na5)

STKUNF - Microcontroller Stack Underflow Reset

1 = A stack underflow reset condition occurred since the last device status query.

0 = No stack underflow reset detected

This bit is cleared following device status query.

STKFUL - Microcontroller Stack Overflow Reset

1 = A stack overflow reset condition occurred since the last device status query.

0 = No stack overflow reset detected

This bit is cleared following device status query.

Monitor FW Code Checksum

Expected SW Code Checksum

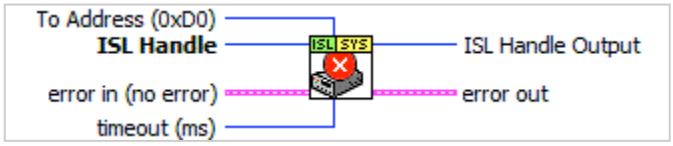
Calculated SW Code Checksum

ISL Handle Output The ISL Reference used to communicate with a device.

6.4.9.3 ISL_Device_Reset.vi

Directs a specified ISL device to perform a hardware reset of the controller.

ISL_Device_Reset.vi



Controls and Indicators

ISL Handle ISL Device Handle

To Address (0xD0) To Address specifies the TIMS processor address.

The default value of 0xD0 refers to the main processor on any TIMS unit.

timeout (ms) timeout specifies the amount of time (in milliseconds) that the LabVIEW driver will wait after each byte is received from a TIMS unit.

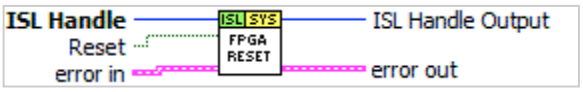
If the next byte is not received within the specified timeout, the IO function will return an error and any byte received up to that point.

ISL Handle Output The ISL Reference used to communicate with a device.

6.4.9.4 ISL_FPGA_Reset.vi

This VI performs an FPGA specific reset function. The FPGA is held in reset for 100 milliseconds and then the FPGA reset is released.

ISL_FPGA_Reset.vi



Controls and Indicators

ISL Handle ISL Device Handle

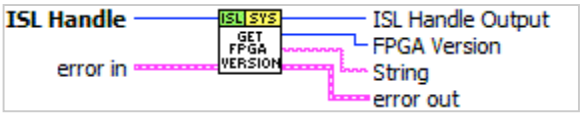
Reset - Obsolete control! Not Used. Reset is performed regardless of the state of this input.

ISL Handle Output ISL Device Handle

6.4.9.5 ISL_FPGA_Version.vi

This VI returns the FPGA firmware version identifier.

ISL_Get_FPGA_Version.vi



Controls and Indicators

ISL Handle ISL Device Handle

FPGA Version FPGA version numeric.

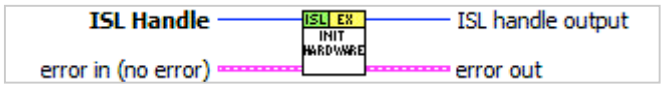
ISL Handle Output ISL Device Handle

String FPGA firmware version string.

6.4.9.6 ISL_Powerup_Init_Hardware.vi

This VI is used to initialize the ISL device hardware when after first applying power to the device. This VI is MANDATORY and must be called after power up before any additional commands can be sent.

ISL_Powerup_Init_Hardware.vi



Controls and Indicators

ISL Handle ISL Device Handle

ISL handle output ISL

6.4.9.7 ISL_Set_FPGA_Reset_State.vi

This VI is used to SET the reset state of the FPGA. Normally the ISL_FPGA_RESET.vi is used to reset cycle the FPGA (hold in reset, wait, and remove reset). This VI can be used to put the FPGA in reset and leave it there, until cleared later by another call to this VI.

ISL_Set_FPGA_Reset_State.vi

Controls and Indicators

ISL Handle ISL Device Handle

Reset This Reset input control that state of the FPGA Reset.

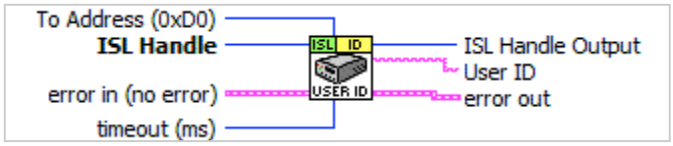
ISL Handle Output ISL Device Handle

6.4.10 ISL Device ID VIs

6.4.10.1 ISL_Device_User_ID_Get.vi

ISL_DEVICE_User_ID_Get.vi

This VI gets the user specified Name for the ISL device.



ISL_DEVICE_User_ID_Get.vi

Controls and Indicators

timeout (ms) timeout specifies the amount of time (in milliseconds) that the LabVIEW driver will wait after each byte is received from a TIMS unit.

If the next byte is not received within the specified timeout, the IO function will return an error and any byte received up to that point.

ISL Handle The ISL Reference used to communicate with a device.

To Address (0xD0) To Address specifies the TIMS processor address.

The default value of 0xD0 refers to the main processor on any TIMS unit.

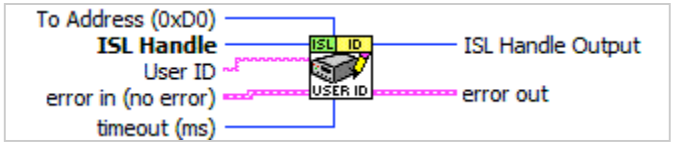
User ID The user specified name for the ISL device.

ISL Handle Output The ISL Reference used to communicate with a device.

6.4.10.2 ISL_Device_User_ID_Set.vi

Sets the user specified **Name** for the ISL device.

ISL_Device_User_ID_Set.vi



Controls and Indicators

To Address (0xD0) To Address specifies the TIMS processor address.

The default value of 0xD0 refers to the main processor on any TIMS unit.

timeout (ms) timeout specifies the amount of time (in milliseconds) that the LabVIEW driver will wait after each byte is received from a TIMS unit.

If the next byte is not received within the specified timeout, the IO function will return an error and any byte received up to that point.

User ID The user-specified name for the ISL device.

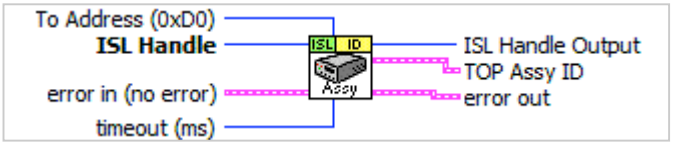
ISL Handle The ISL Reference used to communicate with a device.

ISL Handle Output The ISL Reference used to communicate with a device.

6.4.10.3 ISL_Device_Top_Assy_ID_Get.vi

This VI returns the **Top Assembly Info** from the firmware of a ISL device including Part number, serial number, and miscellaneous information.

ISL_DEVICE_TOP_Assy_ID_Get.vi



Controls and Indicators

timeout (ms) timeout specifies the amount of time (in milliseconds) that the LabVIEW driver will wait after each byte is received from a TIMS unit.

If the next byte is not received within the specified timeout, the IO function will return an error along with any byte received up to that point. What happens here?

ISL Handle The ISL Reference used to communicate with a device.

To Address (0xD0) To Address specifies the TIMS processor address.

The default value of 0xD0 refers to the main processor on any TIMS unit.

TOP Assy ID ISL Device top assembly information

- **Part No.** The Part Number of the ISL Device.
- **Serial No.** The Serial Number of the ISL Device.
- **Misc Info** Miscellaneous information about the ISL device.

ISL Handle Output The ISL Reference used to communicate with a device.

6.4.10.4 ISL_Device_Configuration_Report.vi

This VI is used to query the ISL device and generate a complete report detailing the hardware and software system configuration. This detailed report is especially useful when contacting Jova Solutions for any technical support issues you may encounter.

ISL_Device_Configuration_Report.vi

Controls and Indicators

ISL handle The ISL Handle (reference) to the connected ISL device.

System Configuration System Configuration is a detailed hardware and software configuration report for the connected ISL device.

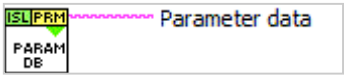
ISL handle output The ISL Handle (reference) of the connected ISL device.

6.4.11 ISL Parameter Database VI

6.4.11.1 ISL_Device_User_ID_Get.vi

This VI returns a complete listing of the ISL internal parameter database. This internal database is generally not needed by users of the ISL product, but may be required for specialized troubleshooting as directed by Jova Solutions.

ISL_Parameter_Info.vi



Controls and Indicators

Parameter data – Complete listing of the internal Image Sensor Lab parameter database.

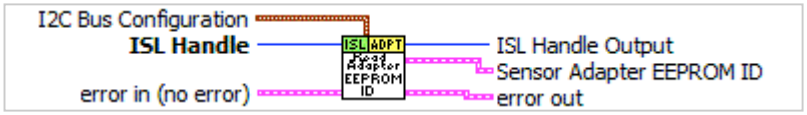
6.4.12 ISL Adapter Board VIs

6.4.12.1 Read_ISL_Adapter_EERPOM_ID.vi

This VI reads the ISL adapter board EEPROM formatted ID information.

Not all ISL adapter boards contain EEPROMs and the ones that do may or may not make use of the Jova Solutions formatted EERPOM memory mapping. Those that do can use this VI to return formatted data from the EEPROM.

Read_ISL_Sensor_Adapter_EEPROM_ID.vi



Controls and Indicators

ISL Handle ISL Device Handle

I2C Bus Configuration

- **Slew Rate**
- **Target I2C Clock Rate (KHz)**
- **Timeout (ms)**

ISL Handle Output ISL Device Handle

Sensor Adapter EEPROM ID ISL adapter board EERPOM contents including model and PCB assembly information.

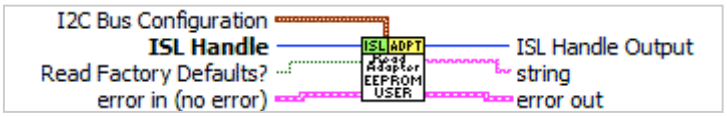
- **Sensor Adapter Type** The Sensor Adapter Type is a numeric indicating the specific type of adapter board attached to the ISL.
- **Model ID** The ISL device Model ID information.
  - **Model No.**
  - **Description**
- **PCB Assy ID** The ISL device PCB Assembly ID information.
  - **Part No.**
  - **Serial No.**
  - **Misc Info**

6.4.12.2 Read_ISL_Sensor_Adapter_EERPOM_User_Info.vi

This VI accesses the user information field within the ISL adapter EERPROM.

NOTE: Not all ISL sensor adapters utilize an EEPROM; this VI will error if one does not exist on the adapter board connected to the ISL device connected to this computer.

Read_ISL_Sensor_Adapter_EEPROM_User_Info.vi



Controls and Indicators

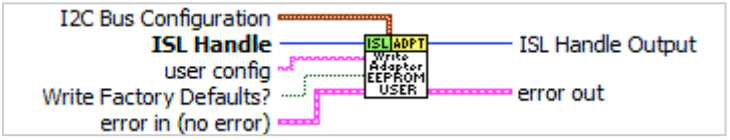
- **ISL Handle** ISL Device Handle
- **I2C Bus Configuration**
  - **Slew Rate**
  - **Target I2C Clock Rate (KHz)**
  - **Timeout (ms)**
- **Read Factory Defaults?**
- **ISL Handle Output** ISL Device Handle
- **string** User information from the ISL sensor adapter EERPOM.

6.4.12.3 Write_ISL_Sensor_Adapter_EERPOM_User_Info.vi

This VI writes the user information field within the ISL adapter EERPOM.

Note: Not all ISL sensor adapters utilize an EEPROM; this VI will error if one does not exist on the adapter board connected to the ISL device connected to this computer.

Write_ISL_Sensor_Adapter_EEPROM_User_Info.vi



Controls and Indicators

- **ISL Handle** ISL Device Handle
- **I2C Bus Configuration**
  - **Slew Rate**
  - **Target I2C Clock Rate (KHz)**
  - **Timeout (ms)**
- **user config** User configuration information to be stored in the ISL sensor adapter board EEPROM device.
- **Write Factory Defaults?**
- **ISL Handle Output** ISL Device Handle

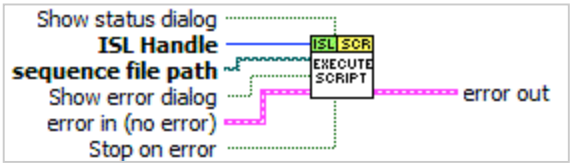
6.4.13 ISL Script/Sequence Engine VIs

6.4.13.1 Execute_ISL_Sequence_File.vi

This VI executes an ISL script file. This VI will optionally display a progress dialog box showing the status for the script execution.

See the ISL Basic Operating Guide for more detailed information about the ISL Scripting Engine.

Execute_ISL_Sequence_File.vi



Controls and Indicators

ISL Handle

sequence file path File path to ISL sequence file to be executed.

Show status dialog Wiring a TRUE to this control enables a script execution status dialog box to appear during the execution of the script.

Stop on error The Stop on Error Boolean input allows a choice of behavior during execution of the script file. If True, the script will immediately stop and abort if an error of any kind is encountered.

Show error dialog If enabled, a dialog box showing details of the error received will be shown during execution of the script when encountering an error.

6.4.14 ISL Application Control VIs

The ISL Application Control VIs interact with the ISL application software and allow LabVIEW developers to write plugin modules that can run in parallel with the ISL application software.

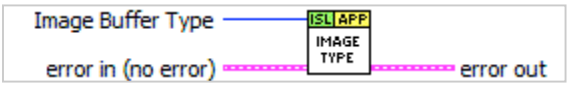
All of the other VI categories contain VIs that command and control the actual hardware. The VIs in this category do not command or control the ISL hardware, but simply set program variables, primarily related to image capture and how the image sensor raw data is decoded to a displayed and analyzed image.

6.4.14.1 ISL_Image_Buffer_Type.vi

This VI is used to specify the image buffer type used for image capture.

0 = Raw (32 bits/pixel)
1 = Bayer (16 bits/pixel)
2 = YCbCr (or YUV) (16 bits/pixel)
3 = RBG (32 bits/pixel)

ISL_Image_Buffer_Type_Set.vi



Controls and Indicators

U16 **Image Buffer Type** Image buffer type used for captured images.
0 = Raw (32 bits/pixel)
1 = Bayer (16 bits/pixel)
2 = YCbCr (or YUV) (16 bits/pixel)
3 = RBG (32 bits/pixel)

6.4.14.2 ISL_Decoding_Type_Set.vi

This VI is used to set the post image capture decoding type used to decode the raw image data into the host computer image buffer.

0 = Standard
1 = Custom
2 = Plugin

ISL_Decoding_Type_Set.vi

Controls and Indicators

U16 **Decoding Type** The Decoding Type control is used to set the post image capture decoding type used to decode the raw image data into the host computer image buffer.
0=Standard
1=Custom
2=Plugin

6.4.14.3 ISL_Clocks_Per_Pixel_Set.vi

This VI us used to set the clocks per pixel that is used post image capture when decoding the raw image data into the host computer image buffer.

ISL_Clocks_Per_Pixel_Set.vi

Controls and Indicators

U16 **Clocks/Pixel** The Clocks/Pixel control is used to set the clocks per pixel that is used post image capture when decoding the raw image data into the host computer image buffer.

0 = 1 clock per pixel

1 = 2 clock per pixel

2 = 3 clock per pixel

3 = 4 clock per pixel

6.4.14.4 ISL_Swap_Bytes_Set.vi

This VI is used to set the swap byte option that is used post image capture when decoding the raw image data into the host computer image buffer.

ISL_Swap_Bytes_Set.vi

Controls and Indicators

TF **Swap 1D Bytes** The Swap ID Bytes control is used to set the swap byte option that is used post image capture when decoding the raw image data into the host computer image buffer.

This function will swap the high and low byte in each captured 16-bit word immediately after data is downloaded to the host computer for processing.

6.4.14.5 Transpose Image Set.vi

This VI is used to set the Transpose Data option that is used to transpose the image buffer data before display or processing.

Transpose Image Set.vi

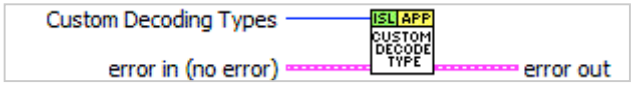
Controls and Indicators

TF **Transpose image data** The Transpose image data control is used to set the Transpose Data option that is used to transpose the image buffer data before display or processing.

6.4.14.6 Custom_Decode_Type.vi

This VI is used to specify the custom decoding type that is used post image capture when decoding the raw image data into the host computer image buffer.

Custom Decode Type Set.vi



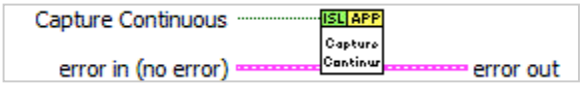
Controls and Indicators

U16 **Custom Decoding Types** The Custom Decode Type control is used to specify the custom decoding type that is used post image capture when decoding the raw image data into the host computer image buffer.

6.4.14.7 Capture_Continuous.vi

This VI is used to command the ISL application software to switch to continuous capture mode.

Capture Continuous Set.vi



Controls and Indicators

TF **Capture Continuous** The Capture Continuous control is used to command the ISL application software to switch to continuous capture mode.

6.4.14.8 Image Rows Set.vi

This VI is used to set the number of rows in the image. This information is used to format the raw sensor data once it is downloaded to the host computer, and before the image is displayed or further processed.

Image Rows Set.vi

Controls and Indicators

U16 **Image Rows** The Image Rows control is used to set the number of rows in the image. This information is used to format the raw sensor data once it is downloaded to the host computer, and before the image is displayed or further processed.

6.4.14.9 Image Columns Set.vi

This VI is used to set the number of columns in the image. This information is used to format the raw sensor data once it is downloaded to the host computer, and before the image is displayed or further processed.

Image Columns Set.vi

Controls and Indicators

U16 **Image Columns** The Image Columns control is used to set the number of columns in the image. This information is used to format the raw sensor data once it is downloaded to the host computer, and before the image is displayed or further processed.

6.4.14.10 Bayer Pattern Set.vi

This VI is used to specify the Bayer pattern color filter order in the pixel data.

Bayer Pattern Set.vi

Controls and Indicators

U16 **Bayer Pattern** The Bayer Pattern control is used to specify the Bayer pattern color filter order in the pixel data.

0 = GBGB::RGRG

1= GRGR::BGBG

2= BGBG::GRGR

3= RGRG::GBGB

6.4.14.11 Raw Filter Type Set.vi

This VI is used to set the RAW image buffer display filter type. After raw sensor data is downloaded from the ISL device it is decoded and stored in one of four image buffer types. Each image buffer type has an associated display filter type that is used mostly for display purposes and does NOT overwrite any of the image buffer data.

RAW Filter Type Set.vi

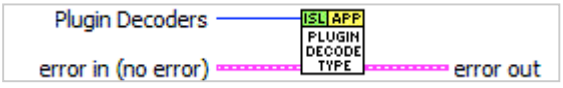
Controls and Indicators

U16 **Filter** The Filter control is used to set the RAW image buffer display filter type. After raw sensor data is downloaded from the ISL device it is decoded and stored in one of four image buffer types. Each image buffer type has an associated display filter type that is used mostly for display purposes and does NOT overwrite any of the image buffer data.

6.4.14.12 Plugin Decode Type Set.vi

This VI is used to specify the plugin decoding type that is used post image capture when decoding the raw image data into the host computer image buffer.

Plugin Decode Type Set.vi



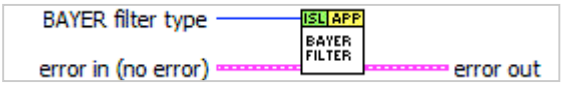
Controls and Indicators

U16 **Plugin Decoders** The Plugin Decode Type control is used to specify the custom decoding type that is used post image capture when decoding the raw image data into the host computer image buffer.

6.4.14.13 Bayer Filter Type Set.vi

This VI is used to set the BAYER image buffer display filter type. After raw sensor data is downloaded from the ISL device it is decoded and stored in one of four image buffer types. Each image buffer type has an associated display filter type that is used mostly for display purposes and does NOT overwrite any of the image buffer data.

BAYER Filter Type Set.vi



Controls and Indicators

BAYER filter type The BAYER filter type control is used to set the BAYER image buffer display filter type. After raw sensor data is downloaded from the ISL device it is decoded and stored in one of four image buffer types. Each image buffer type has an associated display filter type that is used mostly for display purposes and does NOT overwrite any of the image buffer data.

6.4.14.14 RGB Filter Type Set.vi

This VI is used to set the RGB image buffer display filter type. After raw sensor data is downloaded from the ISL device it is decoded and stored in one of four image buffer types. Each image buffer type has an associated display filter type that is used mostly for display purposes and does NOT overwrite any of the image buffer data.

RGB Filter Type Set.vi

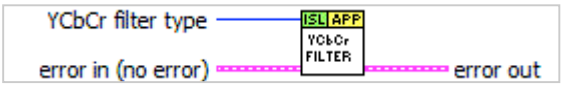
Controls and Indicators

RGB filter type The RGB filter type VI is used to set the RGB image buffer display filter type. After raw sensor data is downloaded from the ISL device it is decoded and stored in one of four image buffer types. Each image buffer type has an associated display filter type that is used mostly for display purposes and does NOT overwrite any of the image buffer data.

6.4.14.15 YCbCr Filter Type Set.vi

This VI is used to set the YCBCR image buffer display filter type. After raw sensor data is downloaded from the ISL device it is decoded and stored in one of four image buffer types. Each image buffer type has an associated display filter type that is used mostly for display purposes and does NOT overwrite any of the image buffer data.

YCbCr Filter Type Set.vi



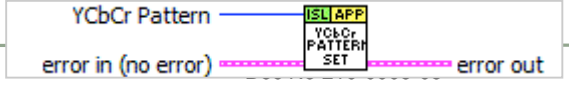
Controls and Indicators

YCbCr filter type The YCbCr filter type control is used to set the YCBCR image buffer display filter type. After raw sensor data is downloaded from the ISL device it is decoded and stored in one of four image buffer types. Each image buffer type has an associated display filter type that is used mostly for display purposes and does NOT overwrite any of the image buffer data.

6.4.14.16 YCbCr Pattern Set.vi

This VI is used to specify the YCbCr pattern order in the pixel data and is used during the decoding process.

YCbCr Pattern Set.vi



Controls and Indicators

YCbCr Pattern The YCbCr Pattern control is used to specify the YCbCr pattern order in the pixel data and is used during the decoding process.

YCbCr Pattern Settings:

0 = Y- Cb - Y - Cr

1= Y - Cr - Y - Cb

2 = Cb - Y - Cr - Y

3 = Cr - Y - Cb - Y