VSLogix

User Guide

Document Revision 1.1

(Updated June 9th, 2022)

Buford, GA USA

For more information please visit the product web page:

www.vitalsystem.com/smartasi

Contents

License Agreement

Before using this software and accompanying software tools, please take a moment to go thru this License agreement. Any use of this software and associated hardware indicate your acceptance to this agreement.

VSLogix is protected by copyright laws and international treaties. Unauthorized reproduction or distribution of this software may result in severe civil and criminal penalties, and will be prosecuted to the maximum extent possible under the law.

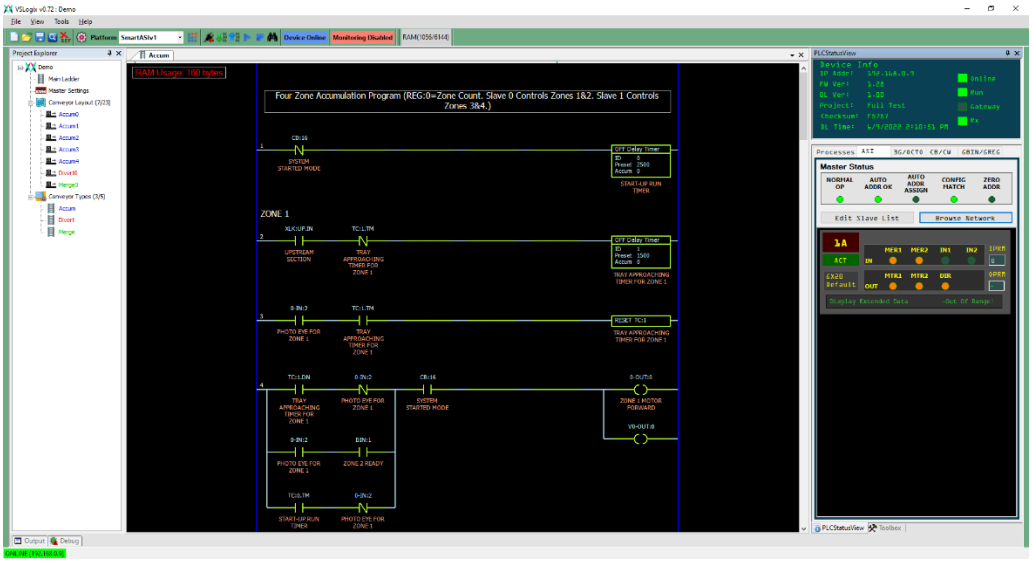
It is the nature of all mechanical and electrical systems to be hazardous. In order to be permitted to use VSLogix with any machine you must agree to the following terms:

I agree that no-one other than the user of this software, will, under any circumstances be responsible for its operation, safety, and use. I agree there is no situation under which I would consider Vital Systems, or any of its distributors to be responsible for any losses, damages, or other misfortunes suffered through the use of VSLogix. I understand that VSLogix and associated hardware is very complex, and though the engineers make every effort to achieve a bug free environment, that I will hold no-one other than myself responsible for mistakes, errors, material loss, personal damages, secondary damages, faults or errors of any kind, caused by any circumstance, any bugs, or any undesired response by the software while running my machine or device.

I fully accept all responsibility for the operation of this software and associated hardware and for its operation by others who may use the software. It is my responsibility to warn any others who may operate any device under the control of the software of the limitations so imposed.

I fully accept the above statements, and I will comply at all times with standard operating procedures and safety requirements pertinent to my area or country, and will endeavor to ensure the safety of all operators, as well as anyone near or in the area of operation.

I. Introduction



VSLogix is a graphical and feature-rich ladder-logic programming environment for the SmartASI PLC/Gateway. It has been developed to meet industrial programming standards for many specialized applications. VSLogix also serves as a direct interface to SmartASI cards and their connected ASI network. It allows for configuration and real-time monitoring of ladder programs, as well as the current status and I/O of all connected ASI Slaves.

VSLogix Features:

- Create robust, powerful, and reusable ladder-logic programs and download them to cards.
- Upload and modify existing ladder-logic programs directly from SmartASI cards.
- Scan and Configure ASI network from VSLogix software using SmartASI card as a gateway.
- Monitor program execution and debug by changing values from the VSLogix software during runtime.
- Drag-and-drop Graphical User Interface for simple, yet extensive ladder-logic programming.

SmartASI Features:

- In Gateway mode, provides Ethernet access to an ASI network of up to 31 ASI slaves
- In PLC mode, can run many ladder programs controlling up to 31 ASI slaves or 30 Smart3G cards

Supported Slaves:

GX20	Vital System's custom ASI slave with 2 48V Motor outputs (M8 5-Pin) with direction control, and plugs for motor temperature sensors. Sends Voltage, Current, and Temperature data back to the SmartASI for use in Ladder Programs or by a Master PLC.

II. Basic Concepts

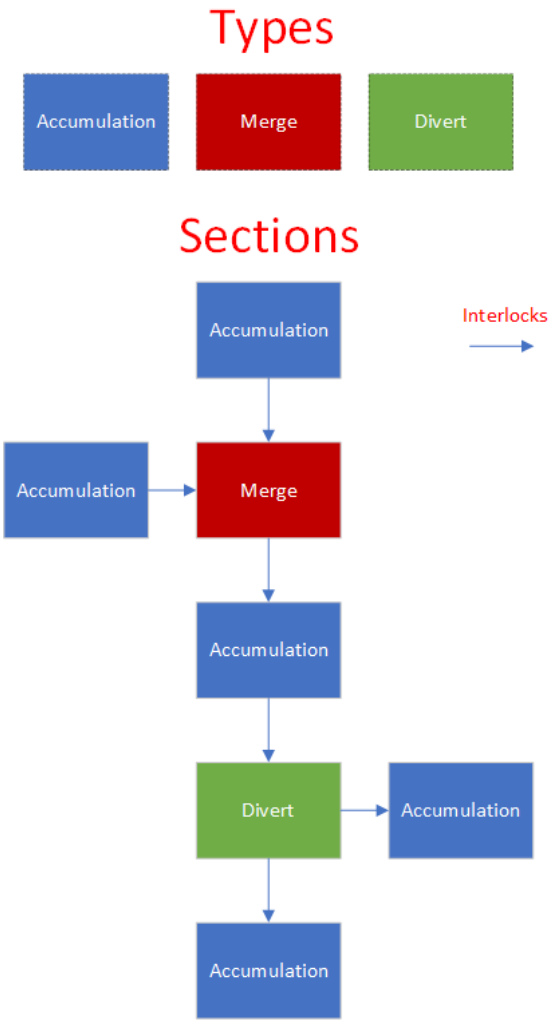
The SmartASI PLC/Gateway can control up to 31 ASI Slaves and up to 30 Smart3G/OCTO cards at once. How these slaves will be controlled depends on which of two modes the SmartASI is in: PLC Mode or Gateway Mode. In both modes, the SmartASI contains a SlaveList that specifies which ASI slaves it will do data exchange with. This list can be configured from the Slave Editor window, or from the ASI Network Browser. In Gateway mode the slaves will be commanded by a master PLC connected to the SmartASI over Ethernet/IP. In PLC mode the slaves will be commanded by ladder programs running on the SmartASI itself.

Gateway Mode

In Gateway Mode, no ladder programs will run on the SmartASI itself. Instead, the SmartASI will simply act as a gateway to the ASI network for a connected Ethernet/IP Master. You must still, however, configure the SmartASI's slave list for data exchange and download the project to the card. Use of the Slave Browser is the quickest and easiest way to accomplish this. Refer to the Protocols section for specific information on the recommended Ethernet/IP settings and message format.

PLC Mode

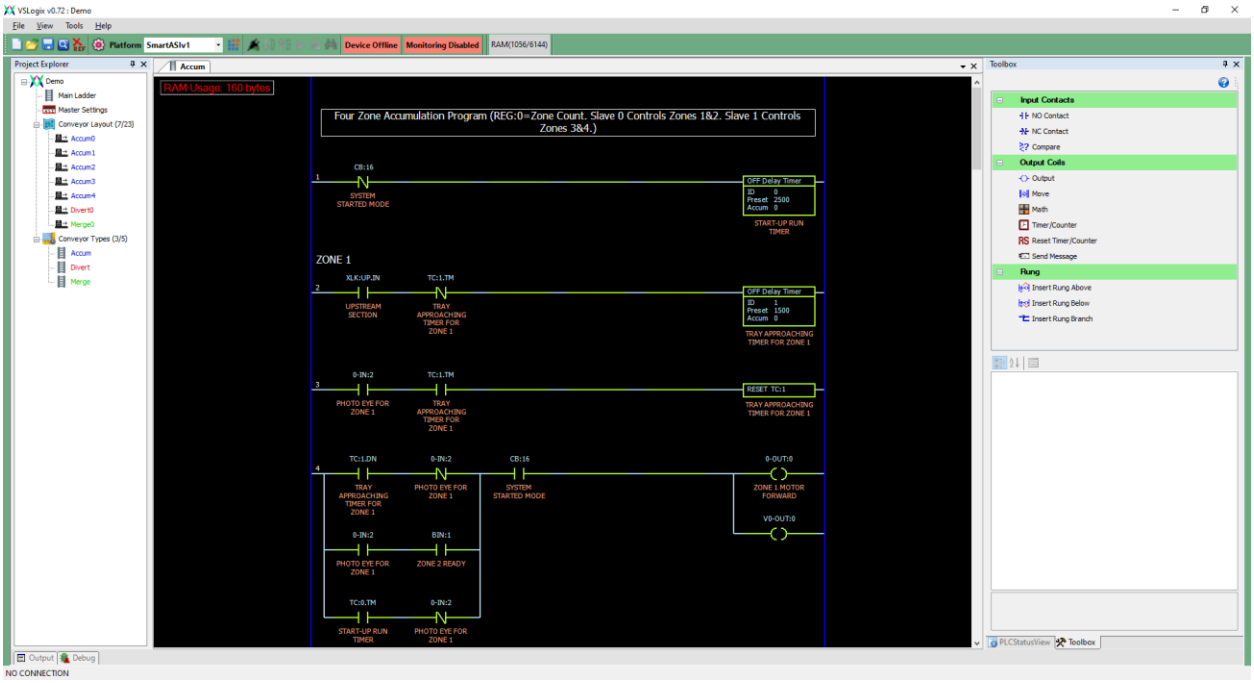
In PLC Mode, the SmartASI can store up to 6 different Ladder Programs at a time. One of these is the Main Ladder, while the other 5 are known as Conveyor Types. There is always one Main ladder in every project, even if it is goes unused. Conveyor Types on the other hand serve as templates that can be instantiated multiple times in the Conveyor Layout Editor. These instantiated Conveyor Types are known as Conveyor Sections. This allows you to create a few different ladder programs and then design an entire conveyor in a graphical view using those types. Each Conveyor section can have a set of assigned ASI slaves that they can control, and can have interlocks between them for passing data along the conveyor.



For example, one might create Accumulation, Merge, and Divert programs and assemble them into a program as seen on the right. The conveyor type system is not mandatory however. For simpler projects you can instead use just one large Main Ladder if you desire.

The SmartASI can run up to 23 Conveyor Sections at one time. Each Conveyor Section is capable of controlling any number of ASI slaves. However, you cannot exceed the total max number of slaves controllable by the SmartASI (31 ASI, 30 Smart3G/OCTO).

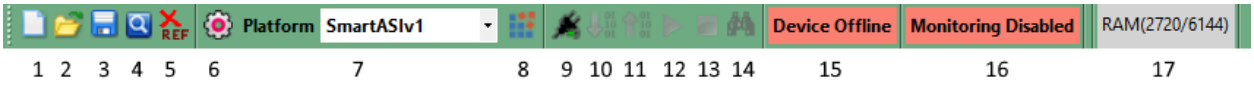
III. User Interface



1. Menu Bar
2. Project Explorer
3. Ladder Library Window
4. Ladder Editor
5. Toolbox
6. Cross Reference Window
7. Debug Window
8. Output Window
9. Conveyor Layout Window
10. Master Settings Window
11. Init Params Window
12. Slave Editor Window
13. Slave Network Browser
14. Connection Dialog
15. PLC Status Window

VSLogix's user interface is split up into a series of windows and dialogues. VSLogix uses a Multiple Document Interface (MDI), which means that many of the software's windows can be opened at once, and allows for features such as split view or opening documents as their own windows. Several of the software's key windows are shown in the above image. You may follow the above links to view detailed descriptions of VSLogix's many windows and capabilities.

Menu Bar



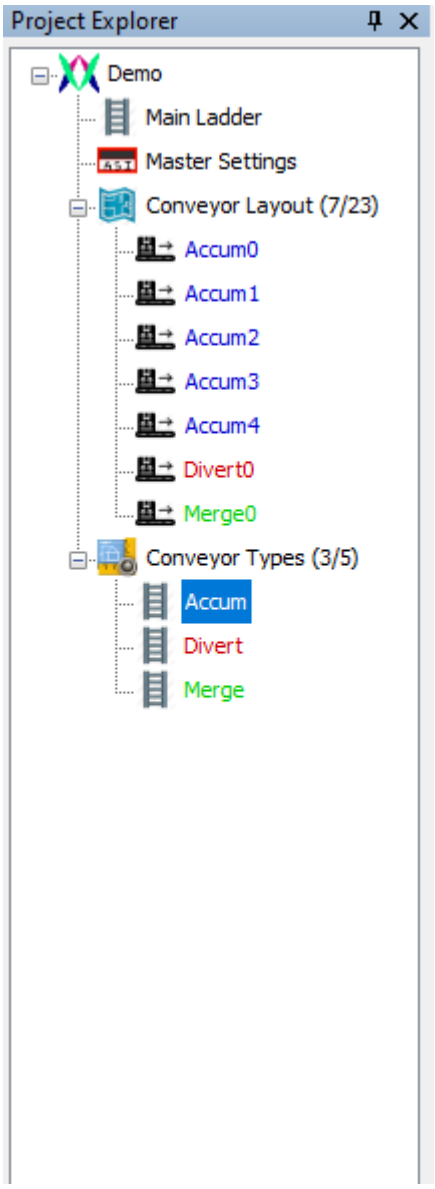
1. Create New Project
2. Open Project
3. Save Project
4. Find Window
5. Cross Reference Window
6. Master Settings Window
7. Select Platform
8. Compile Ladders
9. Connect To Device
10. Download Current Project to Device
11. Upload Project from Device
12. Run All Ladder Programs on Device
13. Stop All Ladder Programs on Device
14. Enable Monitoring
15. Device Connection State
16. Device Monitoring State
17. Memory Consumption

The menu bar is located at the top of the screen and has buttons for various functions.

- 1-3 allow creating/opening/saving projects.
- 4 and 5 are used to search through the ladder programs in the project.
- 6 opens the Master Settings Window for editing global settings.
- 7 is the Platform Selection. The Platform is used to select the intended device the project will be running on. This updates the UI to show the correct limitations and tooltips for the target hardware. A Platform change will be prompted if the connected device does not match the current project. You will be unable to download the project if there is a platform mismatch.
- 8 will compile the project for download.
- 9 is used to connect a device to the VSLogix software.
- 10-14 require an active connection to be used.
- 15-16 display the state of the active connection.
- 17 displays the total memory used by the ladder programs on the device. If the maximum is exceeded, you will not be allowed to download the project to the card.

Project Explorer Window

The Project Explorer Window is the basic navigation and management window for the application. It is open by default, but can be re-opened from the View Menu if closed.



The Project Explorer displays:

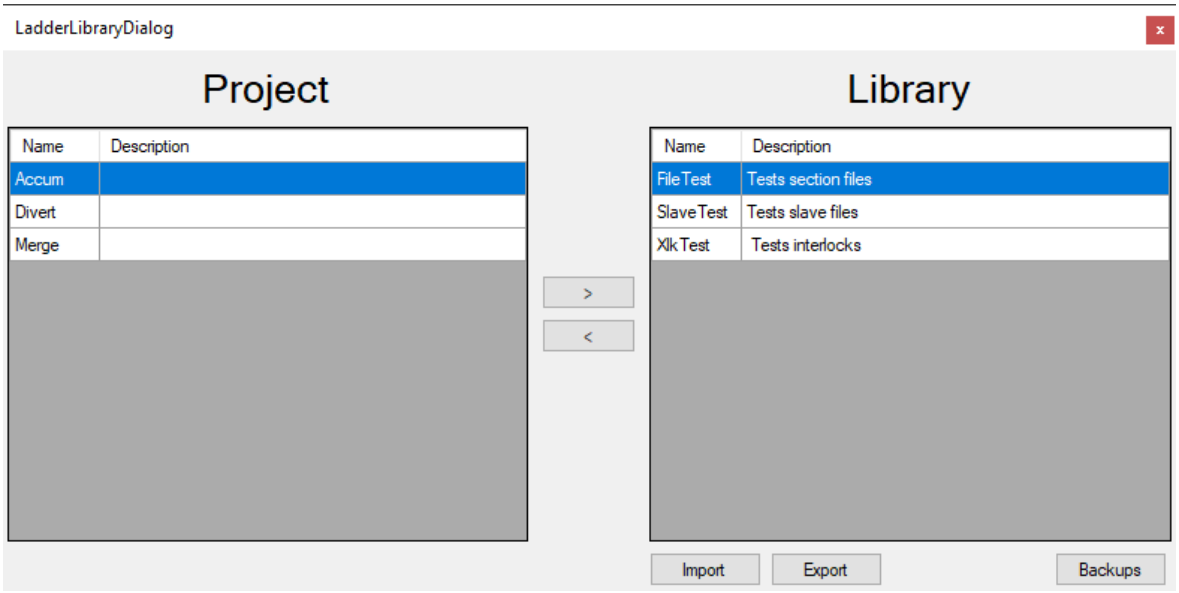
- Current Active Project
- Main Ladder Node – Editable Ladder program that is always present in each project.
    - Double click to open the Ladder Editor.
    - Right click to see options to start, stop, and monitor the Main ladder if a SmartASI is connected to VSLogix.

- Master Settings Node – Opens Master Settings window, allowing editing of global project settings.

- Conveyor Layout Node – Double clicking will open the Conveyor Layout Window, allowing creation and configuring of Conveyor Sections
    - Displayed below this node are the current Conveyor Sections in the project. These nodes allow for configuration, monitoring, and deletion of the Conveyor Sections.

- Conveyor Types Node - Allows creation and management of Conveyor Types. Double clicking this node will open the Ladder Library Window.
    - Displayed below are current Conveyor Types in the project. Double clicking these nodes will open the Ladder Editor.
    - Dragging these nodes into the Conveyor Layout Window will create a Conveyor Section using that Conveyor Type.

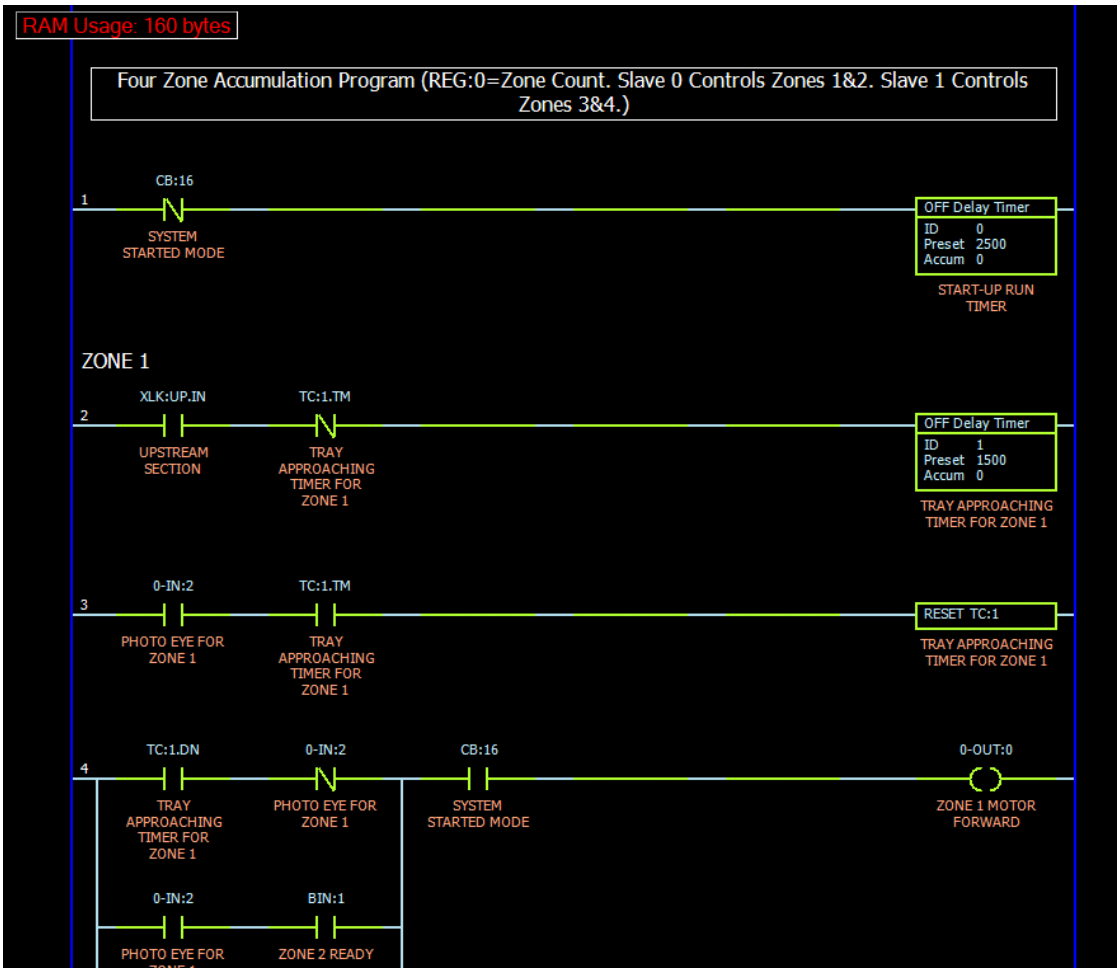
Ladder Library Window



The Ladder Library can be used to store or retrieve ladder programs for use in later projects. It can be opened either by double or clicking the "Conveyor Types" node, or from the tools menu. The Ladder Library will come with a set of example ladder programs in installations of VSLogix. Below is a description of the available functions in the Ladder Library:

>	Copies the selected Project Ladder into the Ladder Library
<	Copies the selected Library Ladder into the Project
Import	Import a Ladder into the Ladder Library from a Project (.asi), Ladder Library (.xml), or Legacy Project (.ldr)
Export	Export the Ladder Library to a Ladder Library file (.xml)
Backups	The Ladder Library keeps the last 50 versions of itself in an Appdata folder. This button opens the folder. Use the import feature to restore backups.
Right Click ->Rename	You can rename a ladder by right clicking it and renaming it. This may be necessary, as duplicate names are not allowed
Right Click ->Delete	Deletes the selected ladders. You can delete multiple at once (use ctrl or shift click to select multiple)
Select Description Cell + Type	By selecting one of the description cells next to the ladder, you may edit its description. This is the same text that will appear at the top of the ladder editor.

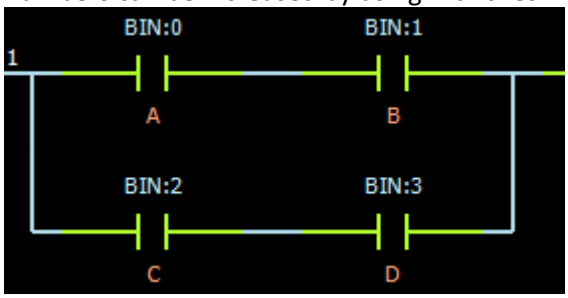
Ladder Editor



The Ladder Editor is where ladder programs are edited. This window can be accessed by Double-Clicking either the Main Ladder, or a Conveyor Type node under “Conveyor Types” in the Project Explorer. The window will also appear when you first create a new Conveyor Type by right clicking the “Conveyor Types” node.

Ladder programs contain Rungs, which contain both Input and Output Ladder Commands. Input Commands (Contacts) test conditions on data, while Output Commands (Coils) perform operations on data. The data these commands operate on is specified by giving them a File Address.

By default, there are 5 Input Commands (Contacts) and 1 Output Command (Coil) per Rung. These numbers can be increased by using Branches.



Among all branches on a rung, if there is at least one path in which all the Input Contacts' conditions are true, then all Output Coils on that rung will be executed. This behavior means that Input Contacts in series will behave as a logical AND, while Input Contacts on parallel branches will behave as a logical OR. For example, the above setup can logically be interpreted as (A && B) || (C && D).

Note that Ladder Programs and Rungs are executed sequentially. The highest rung in a ladder will be executed first, followed by the second, and so on. The order of execution of Ladders Programs is as such:

1. Main Ladder
2. Conveyor Sections, ordered first by Ladder Program, then by Name

Rungs, Branches, and Ladder Commands are inserted into the ladder program and subsequently edited using the Toolbox Window. This window will automatically appear upon opening the Ladder Editor.

Please refer to the File Type and Ladder Command sections for a full list and explanation of all the available Commands and File Types.

At the top of the Ladder Editor, there is a RAM Usage counter. Some File Types (BIN, REG, FLT, and TC) will consume Memory for each unique address used. The amount of memory consumed by a ladder is shown at the top of the editor. Note that memory is allocated in chunks. Each chunk can hold 8 files of a specific filetype before needing to allocate more memory.

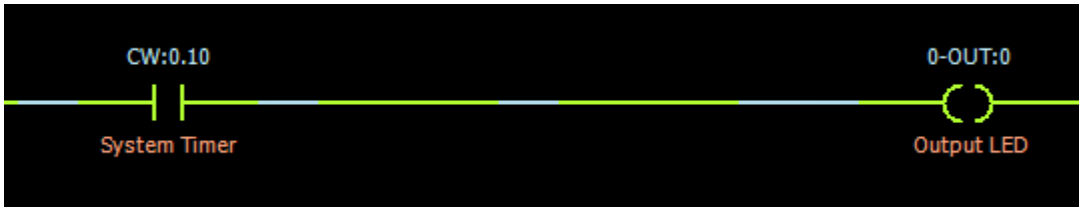
The Total Memory used by the Main Ladder and all Conveyor Sections cannot exceed the max displayed at the top of VSLogix. If it does, you will not be able to download the project to the card.

Below is a table listing useful input commands within the Ladder Editor.

Double Click a Ladder Command	Double clicking any Ladder Command will bring up Quick Edit boxes to change the File Address and Comment. Tab will switch between the two. The Quick Edit Boxes will automatically appear when first placing a Ladder Command.
CTRL+C/CTRL+V	Rungs and Commands can freely be copy/pasted within and between Ladder Programs
Hold CTRL when placing Command, Branch, or Rung	Holding CTRL when placing a Command, Branch, or Rung will allow you to place multiple in a row.
Double Click Ladder Description, or Right Click -> Ladder Description	Edit the Ladder Description at the top of the Ladder. This description is also listed in the LadderLibrary.

Example Ladder

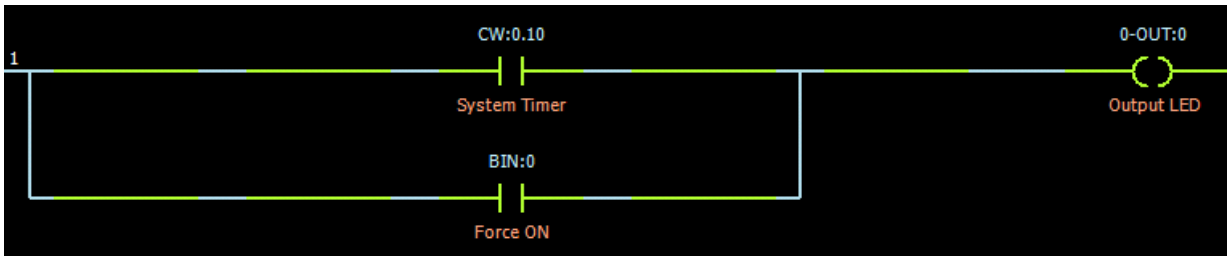
Using the above information of how the Ladder Editor works, we will create an example Ladder to demonstrate the creation process. Let's start by creating a simple flashing LED:



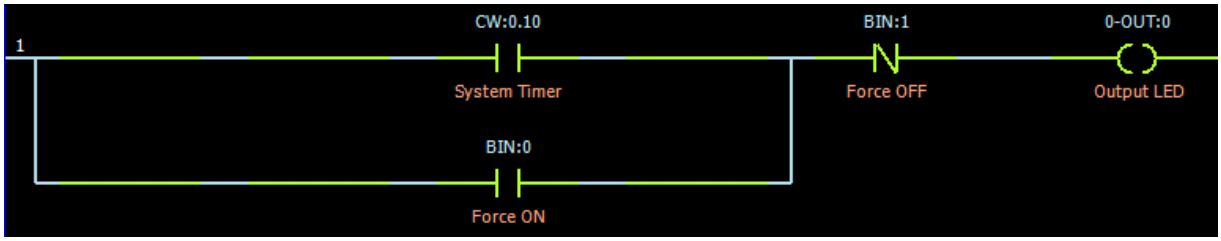
On the left we have a Normally Open command with the address CW:0.10, and on the right we have an Output command with the address 0-OUT:0. A more detailed explanation of these addresses can be found in the FileAddress section, but simply put:

- The Address CW:0.10 refers to the 10^{th} bit of Control Word 0, the System Timer. Since the system timer counts up in miliseconds, the 10^{th} bit will toggle every 2^{10} (aka 1024) miliseconds, or roughly once a second. The Timer command exists for non-power-of-2 timings.
- The Address 0-OUT:0 refers to digital output 0 of Slave 0. Please refer to the Slave File section for an in-depth explanation of this syntax.

The Input command on the left will be active when the address it points to is true (in other words, a one). Since there is only the one input command on the ladder, the Output on this Ladder, the LED, will toggle roughly once a second.



Now, say we wanted to add an option to force the LED to always be on. We could accomplish this by adding a branch parallel to the system timer contact. The output commands on a rung will be activated so long as there is any ***one*** path from left to right whose conditions are true. With this configuration, the state of the system timer contact doesn't matter as long as BIN:0 is true. The output will always be on. We could test this setup by manually toggling BIN:0 while Monitoring the running program.



If we wanted to do the opposite and add a toggle to force the LED off, we could add a normally closed contact in series with both branches. With this configuration, there can be no path from left to right so long as BIN:1 is True.

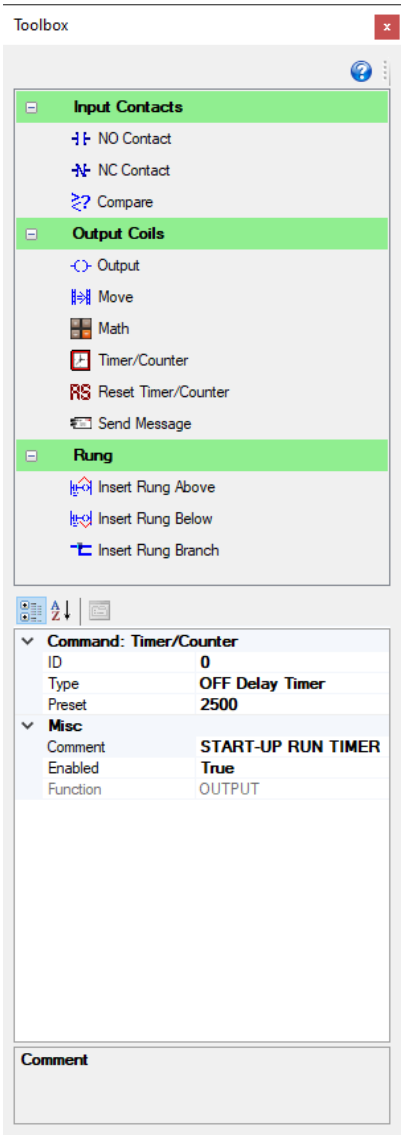
Toolbox Window

The Toolbox window contains the needed tools and actions for creating and editing ladder programs. From this window you can add contacts, rungs, or branches, and edit their various properties.

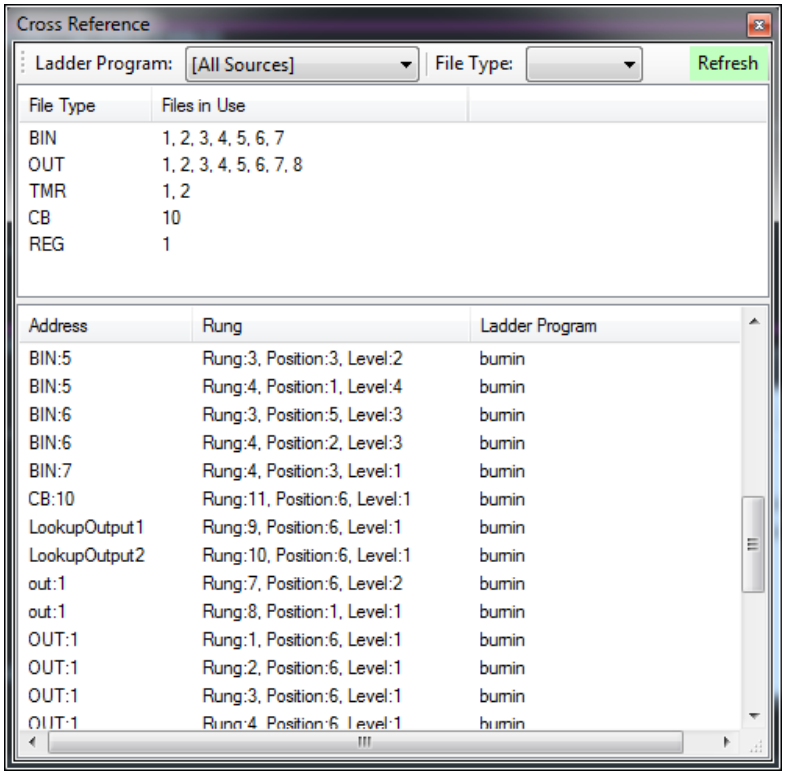
The Tool Box window will be opened automatically whenever the Ladder Editor is opened.

The upper half of the toolbox window contains a list of commands that can be inserted into a Ladder. Click on the command, then select where to place it in the Ladder Editor. You can place multiple commands in a row by holding CTRL when you place it. For a list and explanation of how each of the commands works, see Ladder Commands.

The bottom half of the Toolbox window contains a property list of the currently selected contact, coil, or rung in the Ladder. From here you can edit their various properties. Some examples are the comment that appears below that contact in the editor, or whether the contact is enabled or not. The properties available are dependent on the selected item.

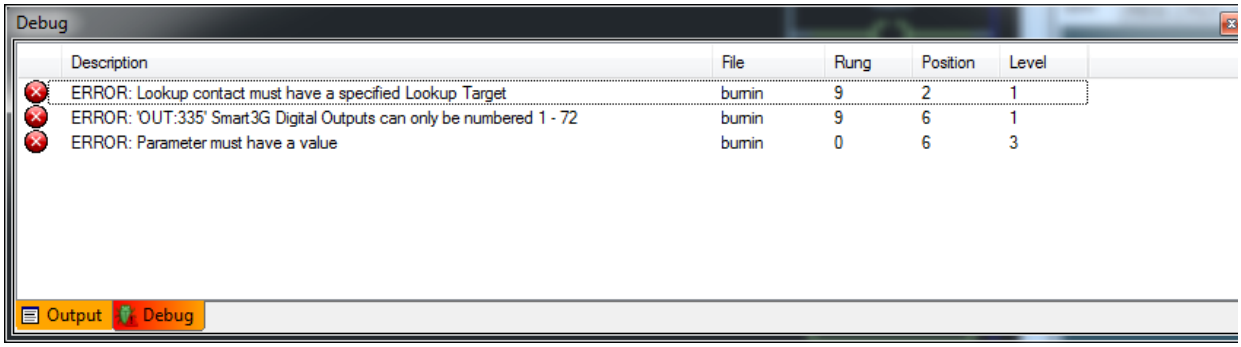


Cross Reference



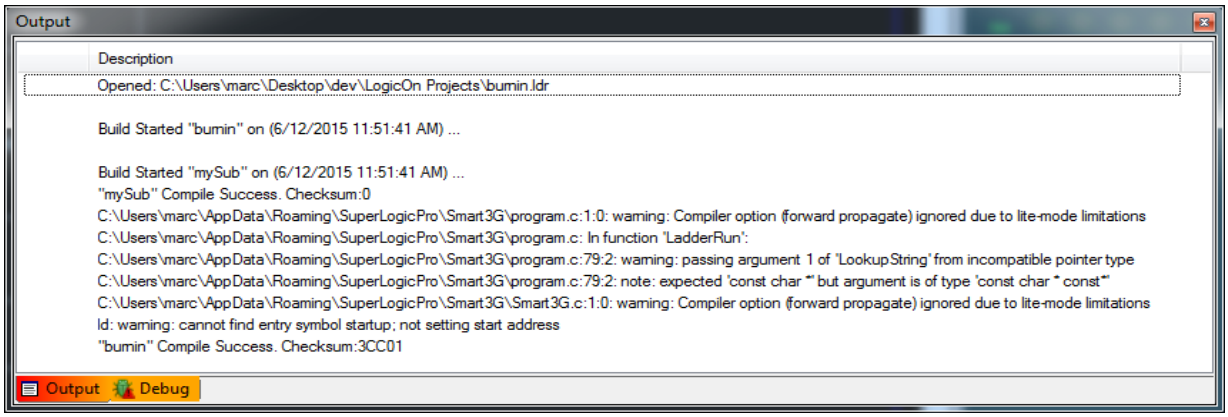
The Cross Reference View provides a list of which Device Files are used in the current project. It also lists which rung and Ladder Program is referencing the file. You can jump directly to that usage by double clicking it.

Debug Window



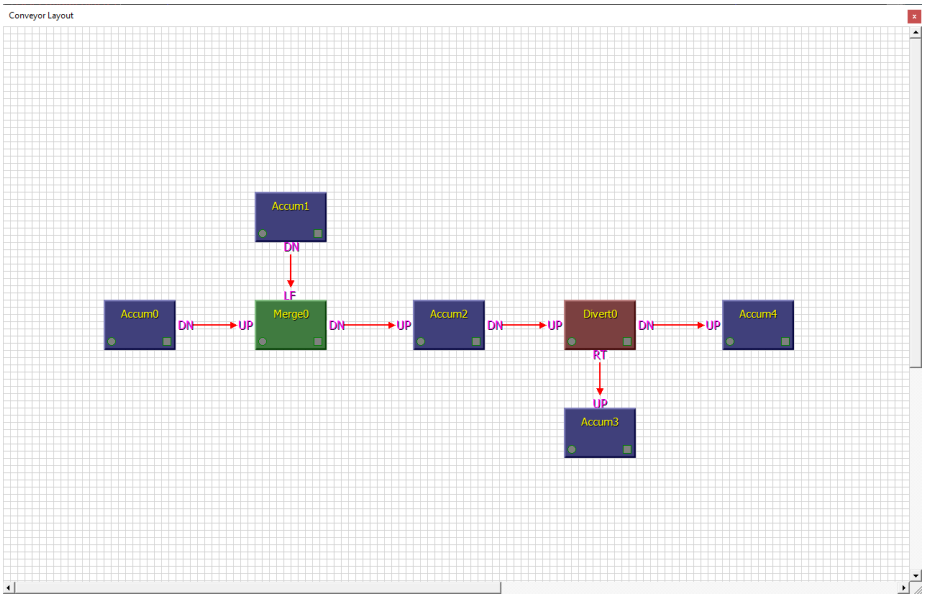
The Debug Window displays compilation errors for the ladder program (if there are any) after the compilation process. Error entries can be clicked to jump to the source of the error. The Debug Window can be accessed by clicking the Debug Window Icon under the menu bar, otherwise it pops-up when errors are detected after compiling a Ladder Program.

Output Window



The Output Window provides detailed and technical feedback such as application log messages, compile messages, and errors. The Output Window can be accessed by clicking the Output Window Icon under the menu bar, otherwise it pops-up automatically when necessary.

Conveyor Layout Window



The Conveyor Layout window is opened by double clicking the corresponding node in the Project Explorer window. This window is used to create and arrange Conveyor Sections in a graphical view. You can create a Conveyor Section by dragging the desired Conveyor Type from the Project Explorer into the Conveyor Layout window.

After creating more than one section, you can create interlocks between them by first selecting one, then pressing a key to choose the interlock, then selecting another section and pressing another key to choose that one's interlock. The available selections are U (Upstream), D (Downstream), L (Left), R (Right), or 1-4 (Auxiliary 1-4). Note the direction of the arrows. Arrows will point away from Downstream interlocks and towards Upstream interlocks to signify the flow of packages.

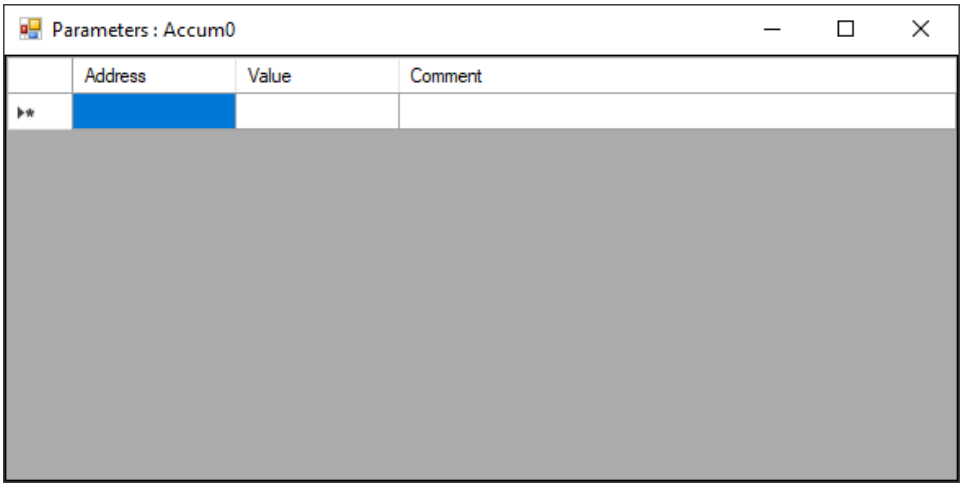
Note that interlocks simply allow the transfer of data from one section to another. It is up to the designer of the ladder program to choose what data to send, and how to process it. Within the Ladder Program, this Interlock data is accessed using the XLK Filetype.

You can also right click the Conveyor Sections to edit their SlaveMaps, to edit their Initial Parameters, or to delete them. You can also select monitor, or double click the section to open its monitoring window. Note that Monitoring the section's execution in real-time will require an active connection to the SmartASI card, and to enable monitoring mode, which will itself require a matching project to be loaded on the card.

Master Settings Window

The Master Settings window is used to configure global settings for the card. At the moment, it is only used to toggle Gateway Mode on or off, and to set the Global Init Parameters. More global settings will be added in the future.

Init Params Editor



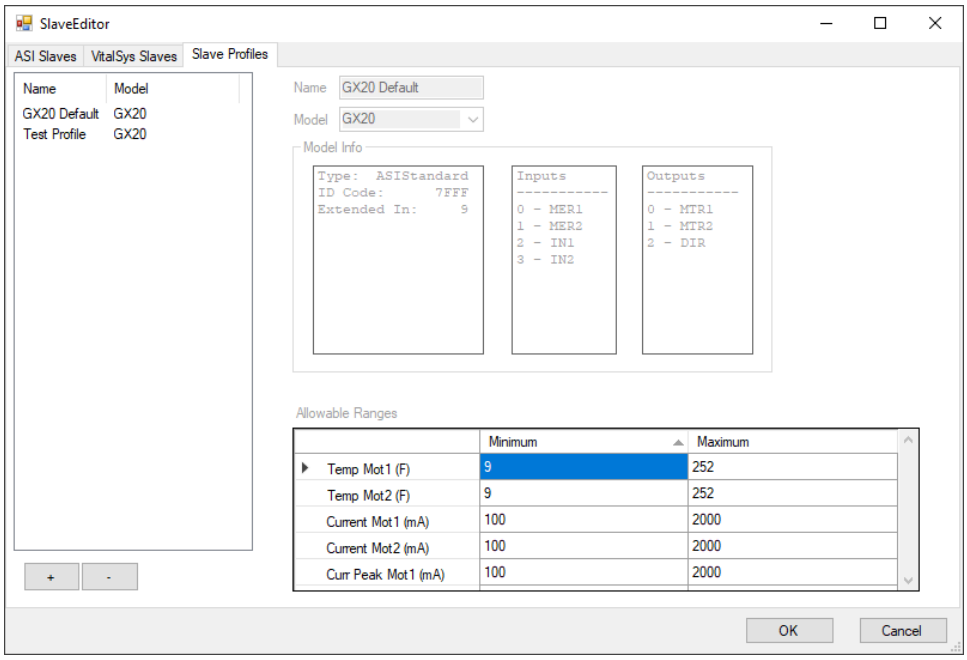
The Initialization Parameters Editor (Init Params for short) allows you to set the values for files on startup of the device. Simply enter the address into the address cell and the desired value into the value cell. Rows can be deleted by selecting the row's header cell and pressing the delete key.
There are two different Init Param editors available:

- One specifically for Global File Types, accessible from the Master Settings Window. There is a limit of 30 Global Init Params.
- One for Local File Types, accessible by right clicking any Conveyor Section. There is a limit of 10 Init Params for each Section.

Slave Editor

You can open this window from the Tools dropdown on the menu bar, by right clicking a Conveyor Section and clicking “Edit SlaveMap”, from the ASI Status Tab, or from the Slave Browser. This window is used to configure the slaves in your project, and will be used in every project. Refer to the Slave File section to see how Slaves should be accessed within ladder programs.

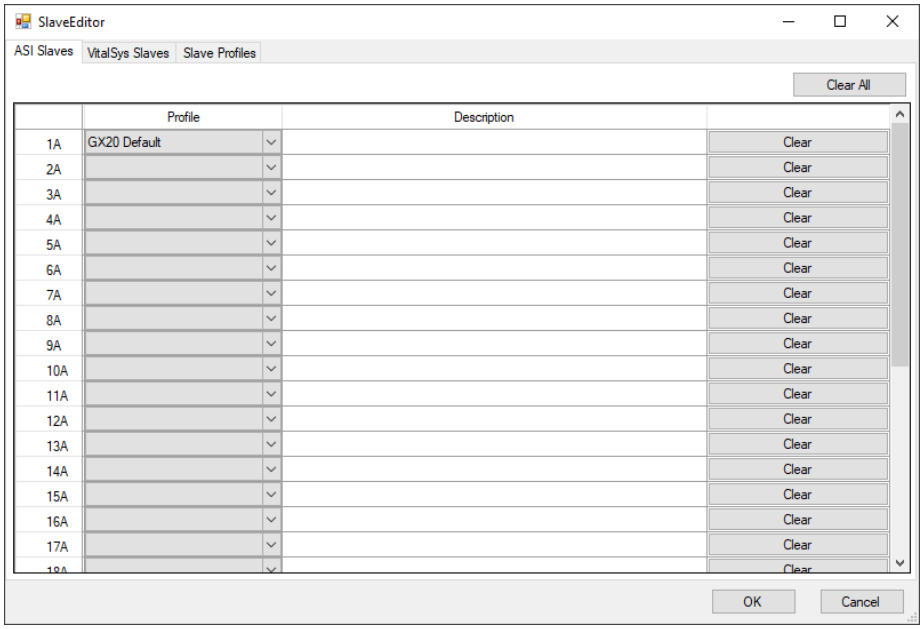
Slave Profiles Tab



In the SlaveProfiles tab, you can create profiles for slave models that contain unique settings. Currently, this is used to configure the acceptable ranges for Extended Data from ASI Slaves. The Status bits of slaves will indicate if the value is outside the acceptable range during runtime. The ASI tab of the Status Window will also indicate when the slaves’ values are out of range.

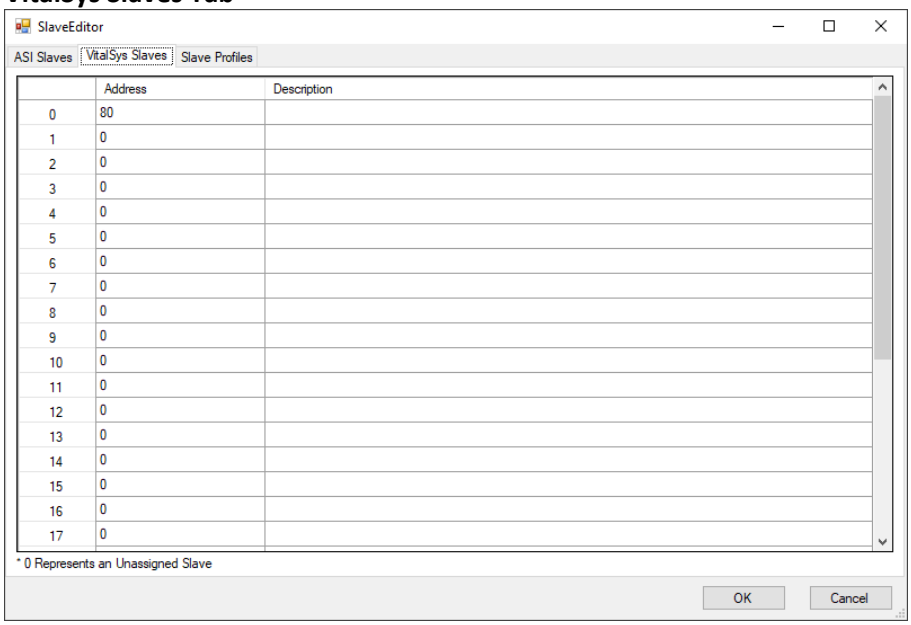
An example usage of the profile system could be if there were two separate slaves with different accepted Motor Current ranges, you could create a different profile for each, then use those profiles in the SlaveList Tab.

SlaveList Tab



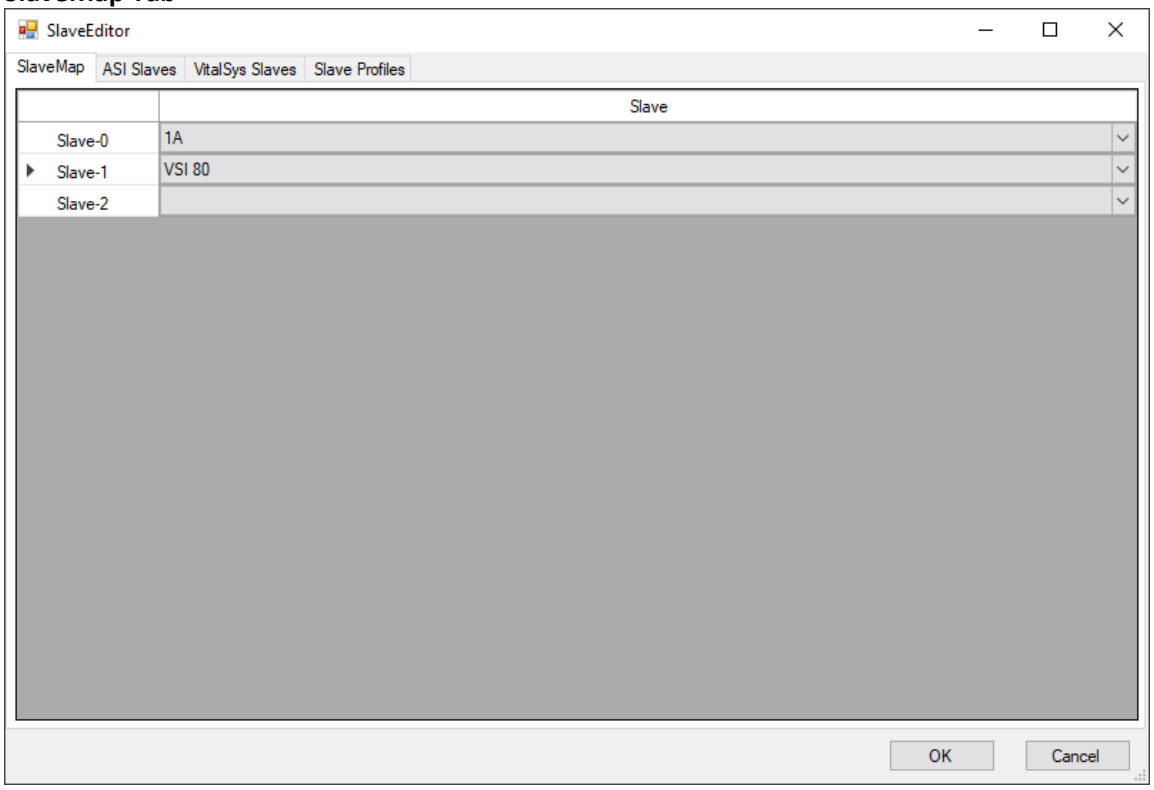
The SlaveList tab allows you to configure which ASI slaves will do data exchange with the SmartASI. Any slave that has a profile selected will be searched for by the SmartASI, and will perform data exchange if it is found. The profile's model should match the type of ASI device you intend to connect to.

VitalSys Slaves Tab



The VitalSys Slaves tab allows you to use Smart3G/OCTO cards as slaves by assigning them an address. After doing so, they will appear in the SlaveMap tab alongside your mapped ASI slaves. Up to 30 Vital Systems Slaves can be mapped on the SmartASI currently.

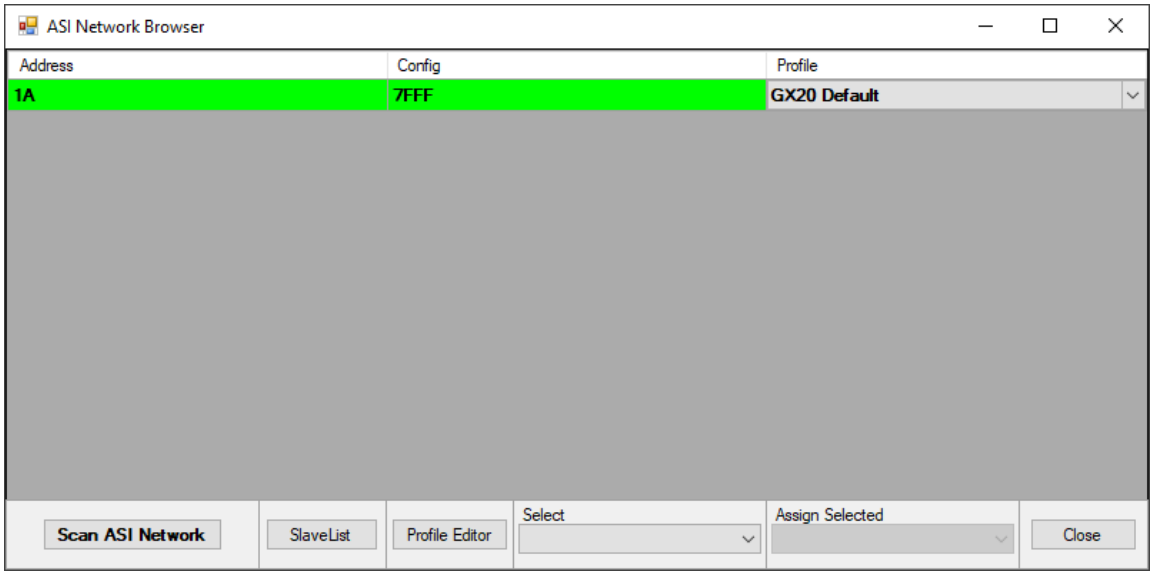
SlaveMap Tab



The SlaveMap tab is only visible when the SlaveEditor is opened by selecting “Edit SlaveMap” after right clicking a conveyor section. It is used to choose the slaves mapped to a conveyor section. All slave placeholders within the ladder program will appear on the left. On the right, you can choose real ASI or VSI slaves from the SlaveList to map to that placeholder. This allows multiple instances of the same ladder program to exist at once, and for them to control different sets of slaves.

Refer to the Slave File section to see how Slaves should be accessed within ladder programs.

Slave Network Browser

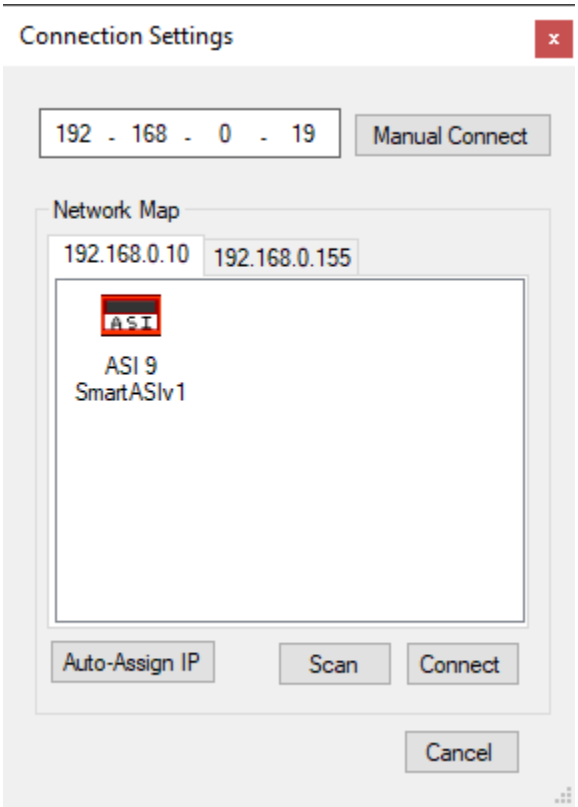


The Slave Network Browser can only be accessed after a SmartASI card is connected to VSLogix via the Connection Dialogue. The ASI network will be scanned and all ASI devices on the network will be displayed in this list. Displayed will be the address of the ASI device, the Config values of the device (IOConfig, Device ID, Extended ID1 and Extended ID2), and finally the configured profile for that slave.

From this window, you can either assign a profile to each device manually using its dropdown box, or you can use the Assign Selected dropdown at the bottom to assign all selected slaves at once. Note that you can only multi-assign when all selected slaves have the same Config/Model. You can select multiple rows for use with the multi-assign feature by either using Shift/CTRL click, or by using the Select dropdown at the bottom to select all slaves with the same Config/Model.

You can also reassign a slave's address from this window by right clicking its entry and selecting Edit Slave Config.

Connection Dialogue



Although VSLogix can compile Ladder-Logic programs, running them requires a connection to a compatible device (SmartASI).

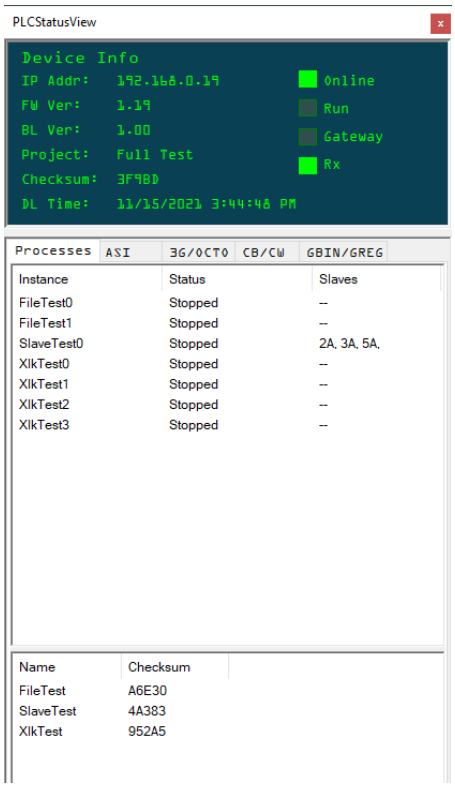
Click the Connect Button to make the Connection Dialogue appear. The connection dialogue will scan all available network adapters and create a list of all detected cards. You can either double click, or select the card and click Connect to connect to a card. If the desired card is not listed after the scan, you can attempt to connect manually by typing the desired IP into the IP box, and clicking Manual Connect.

Auto-Assign IP will command all compatible devices on the network to change their IP Address to match the network they're on. It is a good idea to attempt this if the card fails to appear when scanning.

NOTE: *Before attempting a connection, make sure that the device is connected to the network and is not in an error state (such as conflicting IP Address).*

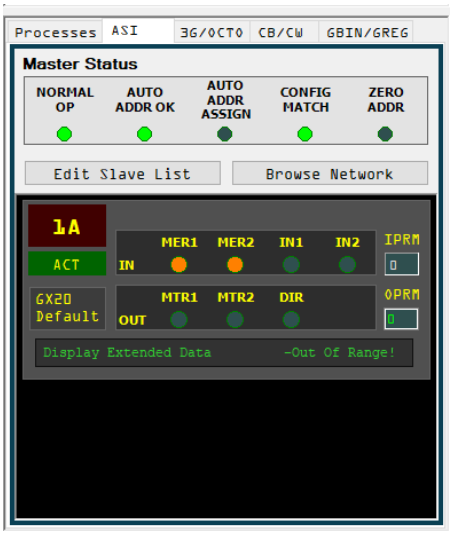
Device Status Window

The Status Window is open by default, but can be accessed from the View menu at the top of the screen if it is closed. The Status window displays information about the connected device, as well as its files. At the top of the window is generic information about the device, such as its IP address and the name of the loaded project. The bottom of the window is split into 5 Tabs:



Processes

This tab displays all the conveyor sections in the project currently loaded on the card, as well as their Status (whether they're currently running or not), and the ASI/VSI Slaves mapped to that section. You may start or stop an individual section, or monitor its execution by right clicking it. At the bottom of the tab are all the Ladder Programs in the project, as well as their checksums.



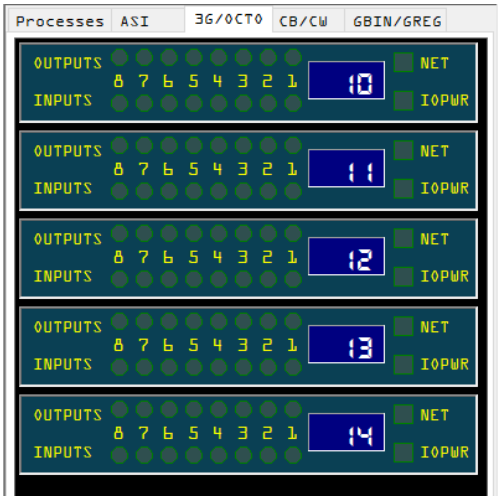
ASI

The ASI tab displays the ASI status of the SmartASI master, as well as the status and IO of all the configured slaves in the connected card's SlaveList. You can witness the status and IO change in real time, and can manually toggle the IO yourself either by clicking the desired Output, or by creating a Slave Override by right clicking the slave. Creating a Slave Override will allow you to override the output regardless of the program's execution.

From this tab you may also open the Slave Editor for the current project by clicking Edit Slave List, or open the Slave Network Browser by clicking Browse Network.

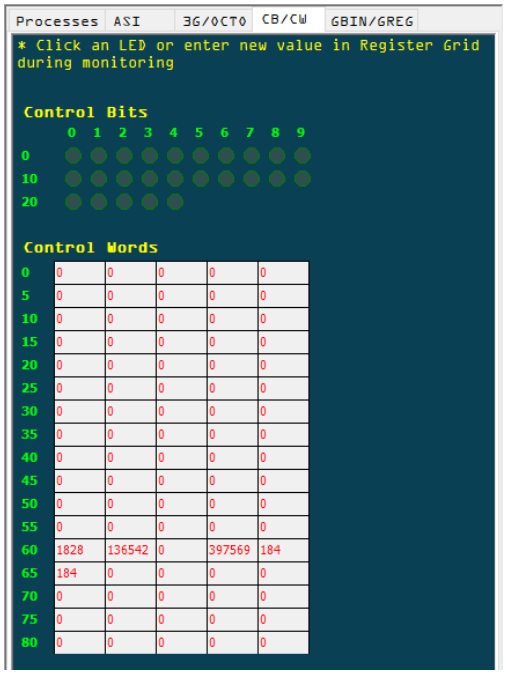
Here is a description of each of the Master Status bits:

- NORMAL OP – The Master is searching on the ASI network as normal.
- AUTO ADDR OK – The master can do Auto Address Assign. This means that if there is only one slave in the slavelist that is not detected on the network, it will automatically assign a newly detected zero address slave to the missing address. This is useful for quickly replacing slaves when one goes down.
- AUTO ADDR ASSIGN – This LED lights up briefly when Auto Assigning a card on the network.
- CONFIG MATCH – Indicates if the SlaveList and Detected slaves on the network are exactly equal.
- ZERO ADDR – Indicates whether there is a slave with address zero on the network. This address must be changed first before you change any other slave address. This zero address slave will also be eligible for Auto Addressing if it is active.



3G/OCTO

The 3G/OCTO tab allows you to view the Status and IO of the connected card's VSI Slaves, which were configured from the VitalSys Slave Tab. Similar to the ASI tab, you may toggle the Slave's IO by left clicking the desired IO, or by creating an override by right clicking it.



CB/CW and GBIN/GREG

Both the CB/CW and GBIN/GREG window work in the same way. They simply allow you to view the specified files update in real-time on the connected card. You can toggle the Boolean values by clicking them, or edit the integer values by clicking them, entering a new value, and pressing enter. Hovering over either a Control Bit or Control Word cell will display a tooltip for that cell's function.

IV. Workflow

1. Creating Ladder

In the Project Explorer, either open the Main Ladder by double clicking the Main Ladder node, or create a Conveyor Type by right clicking the Conveyor Types node. The Ladder Editor will be opened and you may begin editing the Ladder. Refer to the section on the Ladder Editor as well as the sections on FileTypes and Ladder Commands for instructions on how to design ladder programs.

Specifically, make note of how to use Slave FileTypes in the Slave Files section. This will be necessary for any VSLogix project.

Example Ladders are available on new installations in the Ladder Library.

2. Configuring SlaveList

Open the Slave Editor from the Tools menu. On the SlaveList tab, select a profile for each of the slaves that you intend to use. Selecting a profile for a slave will add it to the SmartASI's projected list for data exchange.

Alternatively, connect the SmartASI board to VSLogix via the Connection Dialogue, then open the Slave Network Browser from the tools menu. The current ASI network will appear in the list. You can select multiple rows by either Shift/Ctrl Clicking, or using the Select Dropdown near the bottom of the window. You can then use the Assign Dropdown to assign profiles to all the slaves at once.

If using 3G/Octo cards as slaves, navigate to the VitalSys Slaves tab of the Slave Editor window, and assign the desired addresses in the list.

3. (Optional) Creating/Configuring Sections

This step is only necessary if using Conveyor Types and Conveyor Sections. If only using the Main Ladder, this step can be skipped.

Open the Conveyor Layout window by double clicking the Conveyor Layout node. Drag Conveyor Types from the project explorer into the window to create Conveyor Sections. There are 3 things you can now do to configure these conveyor sections:

Right Click -> Edit Slavemap	This will open the Slave Editor with an extra tab for the Section's Slavemap. Any Slave Placeholders in the ladder program will appear here in order to be mapped to an ASI or VSI slave that was configured in the Slave Editor.
Right Click -> Edit Init Params	This will open the Init Param Editor. This allows you to set the default starting value of local filetypes within that section.
Create Interlocks	Interlocks can be created between sections to allow passing data between them. The specifics of how to do this are outlined in the Conveyor Layout section.

4. Going Online

After connecting to the card via the Connection Dialogue, the Status window will automatically begin to monitor the state of the card. You will be able to see the IO of the ASI slaves and the Global Files update in real time. Several buttons on the Menu Bar will also now be selectable:

- **Download** – Installs the current project to the device. The device will now always run the ladder programs on power-up, or when toggled by the user in VSLogix.
- **Upload** – Retrieve the project from the device and open it in VSLogix. You can do this to connect to any given card and quickly edit its loaded project, or to monitor the running ladder programs in action (a matching project is required to be opened in order to monitor the ladder programs running on the card.)
- **Run** – Enables the execution of all Conveyor Sections on the card. Sections can also be Run individually by right clicking them and selecting the option from the menu.
- **Stop** – Disables the execution of all Conveyor Sections on the card. Sections can also be Stopped individually by right clicking them and selecting the option from the menu.
- **Monitor** – Toggles monitoring mode. While in Monitoring mode, you will be able to see the execution of the Conveyor Sections' ladder programs in real time. Simply double click any Conveyor Section, or right click and click 'Monitor' to open the Monitoring window for that Section. While in Monitoring mode you will be unable to edit the ladder programs themselves. Clicking the Monitor button while the project on the card does not match the one loaded in VSLogix will download the project to the card, then enable monitoring mode.

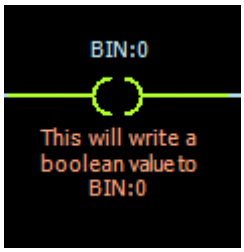
You will also now be able to open the ASI network browser from the ASI tab of the Status window. This will allow you to view all ASI devices on the ASI network, whether they have been configured in the project or not. You can change the devices' addresses from here, or configure the SmartASI's slave list to match the current ASI network.

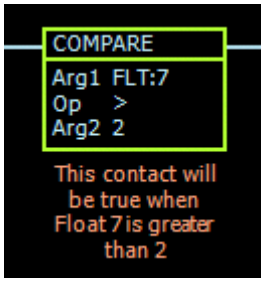
V. Files Addresses

SmartASI Ladder programs use Ladder Commands to operate on addressable Files that store data. The address format for these files follows this basic syntax:

Scope-Type:Index.Attribute

Note that scope and attribute may not be used with all filetypes. Here are a few example usages of files within a ladder:





Refer to the Ladder Editor section for more info on the basics of how the Ladder Programs operate. Refer to the Ladder Command sections for a list of all available commands usable in the ladder program.

File Types are split into 3 categories:

- Local Files – These files are local to the Ladder Program in which they are used
  - BIN – Boolean value (0 or 1)
  - REG – Signed 32-bit Integer
  - FLT – Signed 32-bit Float (floating point value)
  - TC – Timer/Counter
  - XLK – Interlock between sections
- Slave Files – Requires Scope to identify the slave being accessed
  - IN – Digital Input on specified Slave
  - OUT – Digital Output on specified Slave
  - OPRM – Commanded value of Parameter on specified slave
  - IPRM – Actual value of Parameter on specified slave
- Global Files – Shared across all Ladder Programs
  - CB – Control Bit (0 or 1)
  - CW – Control Words (Signed 32-bit Integer)
  - GBIN – Global Boolean value (0 or 1)
  - GREG – Global Signed 32-bit Integer

File Index limitations depend on the category of the File Type:

- Local Files each take up a certain amount of memory. As long as you stay under the total RAM limit, you can use as many of each Type as you want. The exception is the XLK file, of which only 8 interlocks are allowed.
- Both Slave and Global Files have set limitations for each file type. For example, there are only 32 Control Bits. Indexes above this cannot be accessed as they do not exist.

Below is a more detailed explanation for each of these files.

Legend:

Description	General Information and notes on the file type.
Keyword	The keyword syntax used to reference the file.
Scope	Defines the rules about where and how the file can be accessed. Has a value of either Local, Slave, or Global.
Attributes	The attributes that are present for the given file. The file type of the attribute is enclosed in parentheses. Bit Indexing returns a bit using a specified bit position.
Format	The Format address syntax used to reference a specific file in the Ladder Program.
Usage	How the file can be used (Read, Write, or both).

Local File Types

Local Files are local to each running ladder program. The Main Ladder as well all Conveyor Sections have their own copies of every local file they contain.

- BIN – Boolean value (0 or 1)
- REG – Signed 32-bit Integer
- FLT – Signed 32-bit Float (floating point value)
- TC – Timer/Counter
- XLK – Interlock between sections

The types BIN, REG, and FLT will automatically allocate memory as you use them in your program. The TC type requires the use of a TC Ladder Command for each unique index used in the program. The XLK type is hard limited to 8 indexes per program.

Under normal circumstances, ladder programs may only access their own local files. However, it is possible to access another running ladder program's local files by using its name as the scope. For example:

If there exists a Conveyor section named Section1, then the Main Ladder could read or write to that section's Register file by using the address:

Section1-Reg:0

There are some caveats to accessing other Ladder's files like this:

- Ladder and Scope names are case **Insensitive**. Upper and Lower case are considered the same.
- For the scope, using numbers as well as 3G#, OCTO#, and V# are reserved for Slave Files.
- Within the Ladder Editor, comments will not appear for these files

- You can only access files that are used in the target Ladder. Writes to non-existent files will be ignored and Reads will always come back as Zero.
- Although possible, it is likely not recommended to access the files of other conveyor sections from within a conveyor type, since all sections made with that type will be sharing the same file. It is simpler and less error prone to use Global Files if shared files are desired.

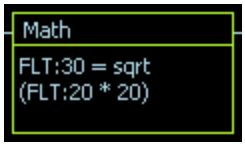
Binary Files

Description	A bit value that can be 0 or 1.
Keyword	BIN
Scope	Local
Data Type	Boolean
Usable Attributes	• None • Full – Directly access the BIN's containing integer (becomes Integer type)
Format	BIN:*<file index>.<optional attribute>*
Usage	Read, Write

Register Files

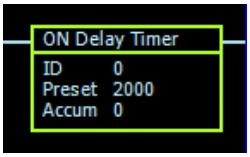
Description	A 32-bit signed integer. Referred to plainly as "Register" in VSLogix. **NOTE**: *Although numerical values with decimal point precision can be assigned to these files, the digits following the decimal point are dropped. For numerical values with a decimal point, consider using the float registers (FLT).*
Keyword	REG
Scope	Local
Data Type	32-bit Signed Integer
Usable Attributes	• None • Bit Indexing [0 – 31] (becomes Boolean type)
Format	REG:*<file index>.<bit index>*
Usage	Read, Write

Float Files



Description	A 32-bit signed floating-point value that can use fractional values following a decimal point.
Keyword	FLT
Scope	Local
Data Type	32-bit Floating Point
Usable Attributes	• None
Format	FLT:*<file index>*
Usage	Read, Write

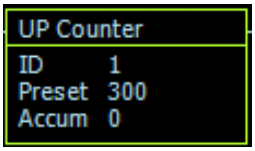
Timer Files



Description	Timers that can keep time in a resolution of milliseconds. The timer keeps time until the user-defined preset value is reached. Unlike other FileTypes, each Timer/Counter File corresponds to an associated Timer/Counter command in the Ladder program. You cannot use a Timer/Counter file index if there is not a Command with that ID. The Filetype TC is used for both Timers and Counters. Whether it is a Timer or Counter can be configured at the Timer/Counter command.
Keyword	TC
Scope	Local
Data Type	32-Bit Signed Integer / Boolean

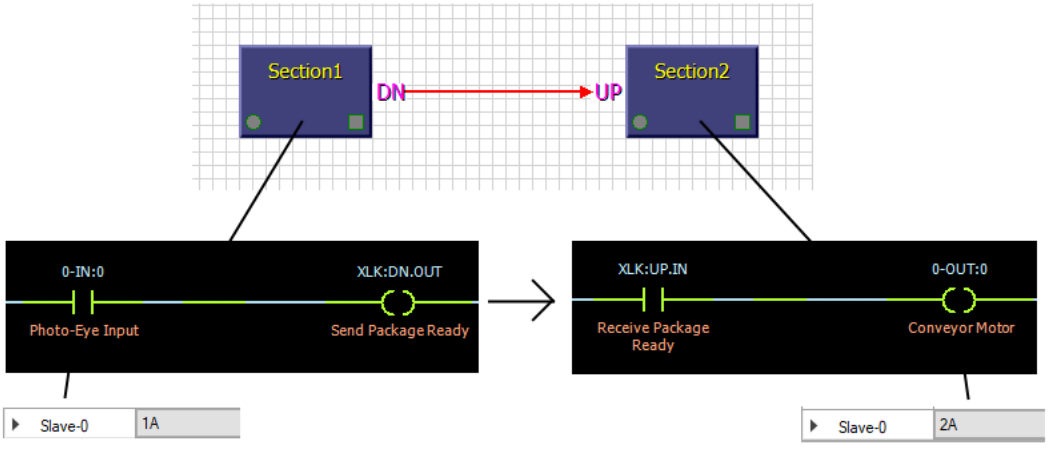
Usable Attributes	• **None/AC** – Accumulated Time in milliseconds (int value). • **PR** – Preset Value in milliseconds (int value). • **EN** – Timer Enabled (bool value). • **DN** – Done Timing (bool value). • **TM** – Currently Timing (bool value).
Format	TC:*<file index>*.*<attribute>*
Usage	Read, Write (PR Only)

Counter Files



Description	Counters that increment their accumulated value by 1 every time the rung state transitions to true. Directly usable only with the Reset and Counter Command. Unlike other FileTypes, each Timer/Counter File corresponds to an associated Timer/Counter command in the Ladder program. You cannot use a Timer/Counter file index if there is not a Command with that ID. The Filetype TC is used for both Timers and Counters. Whether it is a Timer or Counter can be configured at the Timer/Counter command.
Keyword	TC
Scope	Local
Data Type	32-Bit Signed Integer / Boolean
Usable Attributes	• **None/AC** – Accumulated Counts (int value). • **PR** – Preset Value (int value). • **EN** – Timer Enabled (bool value). • **DN** – Done Timing (bool value).
Format	TC:*<file index>*.*<attribute>*
Usage	Read, Write (PR Only)

Interlock Files

Description	Used to send or receive a single bit between two connected Conveyor Sections. In place of using a number for the File Index, you can use one of the below Interlock Keywords to choose the Interlock to access: UP, DN, LF, RT, A1, A2, A3, or A4. Use of one of the Attributes IN or OUT is required.
Keyword	XLK
Scope	Local (Unusable in Main Ladder)
Data Type	Boolean
Usable Attributes	• **IN** – Read bit coming from the connected Section (bool value) • **OUT** – Read/Write bit going to the connected Section (bool value)
Format	XLK:*<interlock specifier>.<attribute>*
Usage	Read, Write (OUT only)
Example	In the below diagram, Section1 has a Downstream Interlock Output, while Section2 has an Upstream Interlock Input. Since Section1's downstream is connected to Section2's upstream in the ConveyorLayout window, Section2's output will be activated when Section1's input is activated. 

Slave File Types

Usage of these File Types requires a prefix that specifies which Slave the File belongs to. The format goes like this:

Slave-Keyword:*Index*

So for example:

0-OUT:1

This would access Output 1 of Slave 0.

The acceptable slave prefixes and their meaning are dependent on whether they're being used in either the Main Ladder or a Conveyor Type.

Main Ladder

1-31	The values from 1 to 31 directly refer to ASI Slaves 1A through 31A.
3G#, OCTO#, V#	Using either the prefix of 3G, OCTO, or V followed by a number from 0 to 29 will refer to a Vital Systems slave (Smart3G/OCTO). All three prefixes are functionally the same. It is up to preference which is used. The address for the specified slave should be set to match the actual Slave ID (rotary switch) in the SlaveEditor window. For example, if I wanted to use a Smart3G device with Address 80, I could use the File Address V0-OUT:1, then in the Master Settings window, set VSI Slave 0 to have address 80.

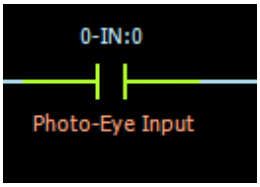
Conveyor Type

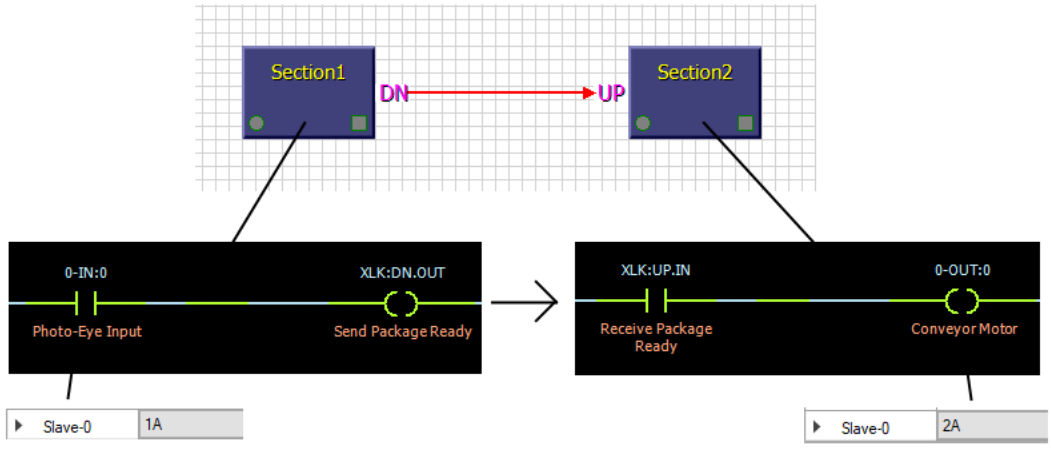
0-30, 3G#, OCTO#, V#	Within a conveyor type, all prefixes have the same meaning. The number refers to a placeholder which will need to be mapped to an actual slave using the SlaveMap Window. This was done so that a single Conveyor Type can be instantiated multiple times and yet control different Slaves. For example, I might use the File Address 0-IN:3 in my Conveyor Type. Later I create two Conveyor Sections from this Type. In the SlaveMap Window on one section, I could map the Placeholder 0 to ASI Slave 5, while in the other I could map it to Smart3G Slave 7.

Slave Files:

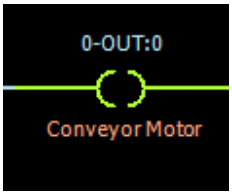
- IN – Digital Input on specified Slave
- OUT – Digital Output on specified Slave
- OPRM – Commanded value of Parameter on specified slave
- IPRM – Actual value of Parameter on specified slave

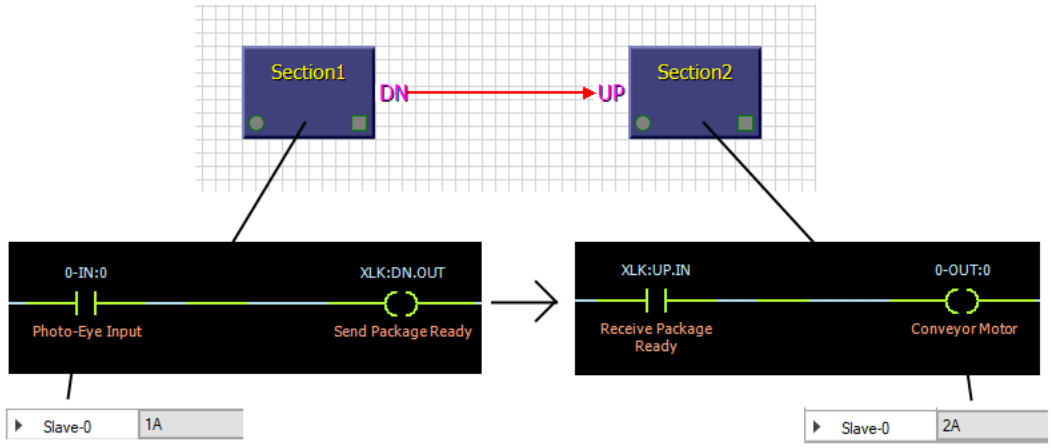
Inputs



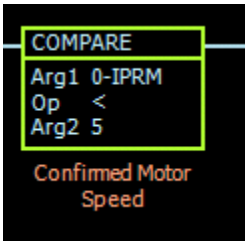
Description	Input state of the selected slave device (similar to BIN values in usage).
Keyword	IN
Scope	Slave
Data Type	Boolean
Usable Attributes	• None • Full - Directly access the File's containing Byte (becomes Byte type)
Format	<*slave*>-IN:<*fileIndex*>
Usage	Read
Example	In the below diagram from the Interlock page, we can see that the Section on the left is going to set an Interlock output (Package Ready) when the Input is triggered (for example, a Photo-Eye input that will detect the presence of a box). In the SlaveMap window for Section1, Slave-0 has been mapped to 1A. Therefore, 0-IN:0 will correspond to Input 0 of ASI Slave 1A. 

Outputs



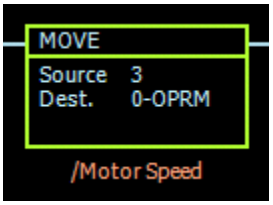
Description	Output state of the selected slave device (similar to BIN values in usage).
Keyword	OUT
Scope	Slave
Data Type	Boolean
Usable Attributes	• None • Full - Directly access the File's containing Byte (becomes Byte type)
Format	<*slave*>-OUT:<*file index*>
Usage	Read, Write
Example	In the below diagram from the Interlock page, we can see that the Section on the right is going to turn on a motor when the Output is activated (in this example, when there is a box present in the previous section.) In the SlaveMap window for Section2, Slave-0 has been mapped to 2A. Therefore, 0-OUT:0 will correspond to Output 0 of ASI Slave 2A. 

Input Parameter



Description	A read-only value that reflects the actual confirmed parameter value of an ASI Slave. For GX20, this corresponds to the Motor Speed.
Keyword	IPRM
Scope	Slave
Data Type	Byte
Usable Attributes	• None • Bit Indexing [0 – 7] (becomes Boolean type)
Format	<*slave*>-IPRM:<*fileIndex*>
Usage	Read

Output Parameter



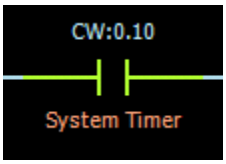
Description	Represents the commanded value of the parameter for an ASI Slave. For GX20, this corresponds to the Motor Speed.
Keyword	OPRM
Scope	Slave
Usable Attributes	• None • Bit Indexing [0 – 7] (becomes Boolean type)
Format	<*slave*>-OPRM:<*file index*>
Usage	Read, Write

Global File Types

These files are shared among all different running ladder programs. They can be assigned a default value on startup using the Init Params tab in the Master Settings Window.

- CB – Control Bit (Boolean type with pre-defined function)
- CW – Control Words (Signed 32-bit Integer with pre-defined function)
- GBIN – Global Boolean value
- GREG – Global Signed 32-bit Integer

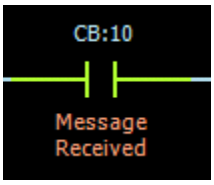
Control Words



<table>
<tr><td>Description</td><td>Special purpose 32-bit signed integers for device control. Each one has a unique predefined function.</td></tr>
<tr><td>Keyword</td><td>CW</td></tr>
<tr><td>Scope</td><td>Global</td></tr>
<tr><td>Data Type</td><td>32-Bit Signed Integer</td></tr>
<tr><td>Usable Attributes</td><td>• None
• Bit Indexing [0 – 31] (Becomes Boolean type)</td></tr>
<tr><td>Format</td><td>CW:<file index>.<attribute></td></tr>
<tr><td>Usage</td><td>
<table>
<tr><th>CW</th><th>Usage</th><th>Description</th></tr>
<tr><td>0</td><td>R</td><td>System Millisecond Timer. Counts up from 0 starting when entering run mode</td></tr>
<tr><td>20</td><td>R/W</td><td>Send Message Address (1-250)</td></tr>
<tr><td>22-29</td><td>R/W</td><td>Send Message Data</td></tr>
<tr><td>30</td><td>R</td><td>Receive Message Address (1-250)</td></tr>
<tr><td>31</td><td>R</td><td>Receive Message Type (Smart3G : 67, SmartASI : 80)</td></tr>
<tr><td>32-39</td><td>R</td><td>Receive Message Data</td></tr>
<tr><td>40-49</td><td>R</td><td>Data from Ethernet/IP Master (Refer to Protocols section)</td></tr>
<tr><td>50-59</td><td>R/W</td><td>Data to Ethernet/IP Master (Refer to Protocols section)</td></tr>
<tr><td>60-68</td><td>R</td><td>Debug data</td></tr>
</table>
*Unmentioned CW indexes are currently reserved</td></tr>
</table>

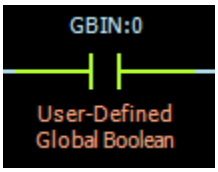
Example	The below example is using a Normally Open command with the 10th bit of CW:0, aka the System Timer. This setup will toggle state once every 2^10 (1024) milliseconds.

Control Bits



Description	Special purpose boolean values for device control. Each one has a predefined function.
Keyword	CB
Scope	Global
Data Type	Boolean
Usable Attributes	• None • Full - Directly access the File's containing Integer (becomes Integer type)
Format	CB:*<file index>*
Usage	**CB** | **Usage** | **Description** 10 | R/W | Message Received Bit. Left to the user to clear it. **Unmentioned CB indexes are currently reserved*

Global Binary Files



Description	A global bit value that can be 0 or 1. The SmartASI has a maximum limit of 24 GBINs at once.

Keyword	GBIN
Scope	Global
Data Type	Boolean
Usable Attributes	• None • Full - Directly access the File's containing Integer (becomes Integer type)
Format	GBIN:*<file index>*
Usage	Read, Write

Global Register Files

Description	A global 32-bit signed integer. The SmartASI has a maximum limit of 20 GREGs at once.
Keyword	GREG
Scope	Global
Data Type	32-Bit Signed Integer
Usable Attributes	• None • Bit Indexing [0 – 31] (Becomes Boolean type)
Format	GREG:*<file index>*.*<bit index>*
Usage	Read, Write

VI. Ladder Commands

A Ladder Program is comprised of multiple rungs that are executed continuously from top to bottom. Each rung is comprised of two types of commands: inputs and outputs.

Input commands are used to determine the state of the rung: active or inactive. The state of the rung decides whether to execute that rung's output commands.

Output commands are commands that carry out actions in the ladder program such as writing to file values. The exact triggering behavior of the output depends on the specific command used: some will perform an action on switching from inactive to active, some will do so on every cycle so long as the rung is active, some when the rung is inactive, etc.

Input Commands

- Normally Open
- Normally Closed
- Compare

Output Commands

- Output
- Move
- Math
- Timer/Counter
- Reset
- Send Message

Input Commands

Input commands are used to determine the state of the rung: active or inactive. If there exists at least one path from the left side of the rung to the right where all input contacts are active, then the Output Commands on that rung will be executed.

- Normally Open
- Normally Closed
- Compare

Normally Open Command

Description	An input command whose condition is true when the addressed bit value is active.
Type	Input
Parameters	**Address** – the referenced bit to read from. (*Ex. IN:1, OUT:2, TC:2.TM*).
Usage	Read
File Types	All Boolean Types. (BIN, IN, OUT, CB, Register Bits, TC Attributes, etc.)

Normally Closed Command

Description	An input command whose condition is true when the addressed bit value is inactive.
Type	Input
Parameters	**Address** – the referenced bit to read from (*Ex. IN:1, OUT:2, TC:2.TM*).
File Types	All Boolean Types. (BIN, IN, OUT, CB, Register Bits, TC Attributes, etc.)

Compare Command

Description	A command whose condition depends on the logical comparison of the values of two referenced files.
Type	Input
Parameters	• **Arg1** – The first argument (referenced file or constant) for comparison. (*Ex. 10, REG:5, "text", BAR:8,10*). • **Arg2** – The second argument (referenced file or constant) for comparison. (*Ex. 10, REG:5, "text", BAR:8,10*). • **Operation** – The logical comparison to make between the 2 arguments. ➢ ***Equal (=)*** – true if the 2 arguments are equal. ➢ ***Not Equal(!=)*** – true if the 2 arguments are not equal. ➢ ***Greater Than (>)*** – true if Arg1 is greater than Arg2. ➢ ***Less Than (<)*** – true if Arg1 is less than Arg2. ➢ ***Greater Than or Equal to (>=)*** – true if Arg1 is greater than or equal to Arg2. ➢ ***Less Than or Equal to (<=)*** – true if Arg1 is less than or equal to Arg2.
File Types	Numerical values (REG, FLT, CW, GREG). Timer/Counter .PR .AC Attributes.
Examples	

Output Commands

The following Commands are all only usable in the last position on each rung. Usually, when at least one path of conditions on a rung are true, the output command for that rung will be activated. The exact triggering behavior may depend on the specific command used though: some will perform an action on switching from inactive to active, some will do so on every cycle so long as the rung is active, and some when the rung is inactive. It is possible to have multiple output commands activated by the same rung by using Branches.

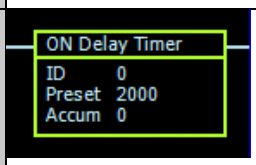
- Output
- Move
- Math
- Timer/Counter
- Reset
- Send Message

Output Command

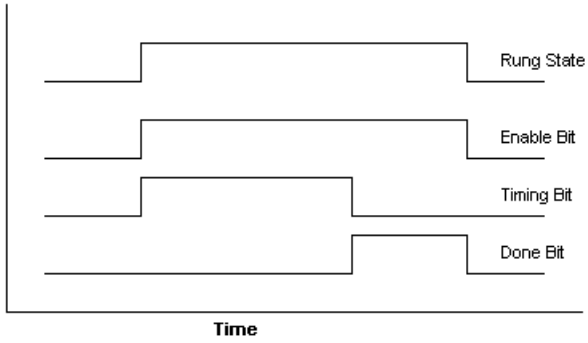
Description	This command sets the addressed bit to true or false. When the rung condition is true, the addressed bit or output is set to true (1 or high) and with rung condition false, the bit or output is set to false (0 or low).
Type	Output
Parameters	• **Address** – the referenced bit value to write to (Ex. Out:1, REG:5.3, etc). • **Output Type** – Controls the behavior of the command. ➢ Normal ( ) – If the rung is active, it writes 1, otherwise it writes 0.  ➢ Latch (**L**) – If the rung is active, it writes 1, otherwise it does nothing.  ➢ Unlatch (**U**) – If the rung is active, it writes 0, otherwise it does nothing.

	 ➢ Latch Transition to True(**^**) – Writes 1 only when the current rung state becomes active and the previous rung state was inactive.  ➢ Latch Transition to False(**v**) – Writes 1 only when the current rung state becomes inactive and the previous rung state was active. 
File Types	Boolean files (BIN, OUT, CB, GBIN), Register Bits (REG:#.x, CW:#.x, GREG:#.x)

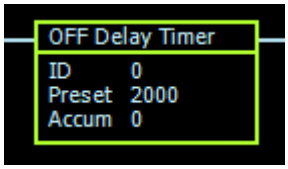
Timer/Counter Command

Description	The Timer/Counter command makes use of either a Timer or Counter register for its functionality • **Timer** – The timer register keeps timing until the preset value (in milliseconds) is reached. • **Counter** – The counter register keeps counting until the reset value is reached.
Type	Output
Parameters	 • **ID** – The timer or counter that is bound to this command. • **Preset** – A 32-bit integer value that specifies when the command stops timing/counting. *For timers, this value is in milliseconds*. • **Type** – Controls the behavior of the command. **NOTE**: *Timers increment the "Accumulator" value every millisecond, while Counters increment the "Accumulator" value every time the rung state transitions from false to true.* ***Types:*** **ON DELAY** – This timer starts timing when the rung condition becomes true. As long as the rung condition is true, the accumulator keeps on timing until it reaches the preset value. When the Accumulator is equal to Preset, the 'Done' bit is set and the Timing bit is reset. The Timer Enable bit will always equal the rung state. The Done bit, Timing Bit

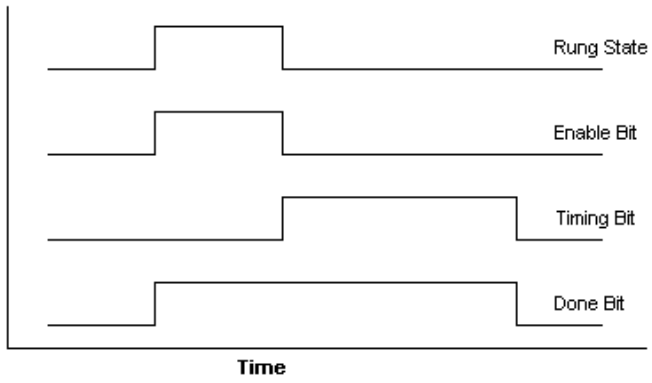
and Enable Bit are reset as soon as the rung state becomes false. The Accumulator is reset to 0 until the rung condition becomes true again.



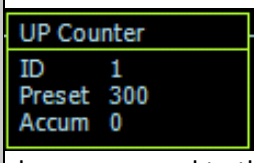
ON Delay Timing Diagram



OFF DELAY – This timer starts timing while the rung condition is false. When the rung state transitions from true to false, the Accumulator keeps incrementing until the preset value is reached or the rung condition becomes true again. The enable bit follows the rung state, while the Done bit is a combination (logical 'OR') of the Enable and Timing bits. The Timing and Done bits are reset when the Accumulator reaches the Preset value.



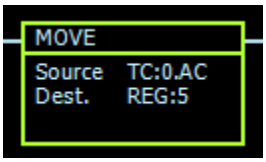
OFF Delay Timing Diagram



UP COUNT – The "UP Counter" counts every rung state transition from false to true until the Preset value is reached. Each transition of rung condition from false to true is registered by incrementing the Accumulator by 1. When the Accumulator value becomes equal to the Preset value, the Done bit is set to true. The Enable Bit follows the rung condition. The "UP Counter" can only be reset with "Reset contact". At reset, the Accumulator value and the Done Bit are set to zero. Please refer to Timer/Counter Reset element.

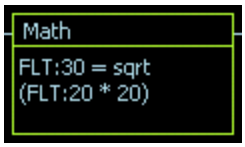
File Types	TC

Move Command



Description	Copies a specified source file's value (or a constant numerical value) into a specified destination file while the rung is active.
Type	Output
Parameters	• **Source** – The referenced file or constant value to be moved (Ex. 5000, 0.056, REG:9, CW:10, etc). • **Destination** – The file where the source file's value is written (Ex. FLT:2, REG:9, CW:10, etc).
File Types	Numerical Values. (REG, FLT, CW, GREG)

Math Command



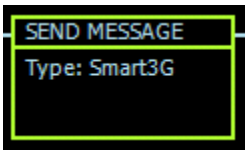
Description	Performs binary and unary mathematical operations (depending on how many arguments were specified) and writes the result to a specified destination file while the rung is active.
Type	Output
Parameters	• **Arg1** – The first argument. (Ex. FLT:4, 5000, 4.556, REG:9, CW:10, etc) • **Arg2** – The second argument. (Ex. FLT:4, 5000, 4.556, REG:9, CW:10, etc) • **Destination** – Address of File to store the result. (Ex. FLT:4, REG:9, CW:10, etc) • **Binary Operation** – operation to perform between the two arguments. All Bit manipulation values must be done with REG (integer) values. **NOTE**: *When using Float type for bitwise operations, the digits after decimal point are dropped, eg. FLT:4 BitAND 123.* ➢ *None* – the result is the value of Arg1. Arg2 is ignored. ➢ *Addition* ➢ *Subtraction* ➢ *Multiplication* ➢ *Division* ➢ *Power/Exponentiation* ➢ *BitAND* – Bitwise AND ➢ *BitOR* – Bitwise OR ➢ *BitXOR* – Bitwise Exclusive OR ➢ *BitShiftLeft* – Shifts Arg1's bit value by a specified number of digits (Arg2's value) to the left. ➢ *BitShiftRight* – Shifts Arg1's bit value by a specified number of digits (Arg2's value) to the right. • **Unary Operation** – operation to perform on the result of the Binary Operation. ➢ *None* ➢ *Negative* – Negates the value. ➢ *Bitwise Inversion* – Invert the bit value. ➢ *Absolute* – Absolute value.

	➢ *Square Root* ➢ *Sine* ➢ *Cosine* ➢ *Tangent* ➢ *Cosecant* ➢ *Secant* ➢ *Cotangent* ➢ *Natural Logarithm* ➢ *Common Logarithm*
File Types	Numerical Types. (REG, FLT, CW)

Reset Command

Description	Resets a timer or counter when the rung state transitions to true.
Type	Output
Parameters	• **Address** – The Timer/Counter to reset. (Ex. TC:3, TC:7 etc)
File Types	TC

Send Message Command



<table>
<tr><td>Description</td><td>Sends a message over Ethernet to another card or PC Host. Data and parameters determined by control words. When the rung condition switches to true, data in the send-buffer will be sent to the receive buffer of the destination device that has been specified by the Send Address Control Word.

<u>NOTE</u>: This command only triggers when the rung state transitions to true.</td></tr>
<tr><td>Type</td><td>Output</td></tr>
<tr><td>Parameters</td><td>

Type

SmartASI – This type sends an explicit message to another SmartASI Device. Sends <u>all 8</u> Control Words in the Send Buffer to the destination device.
Smart3G – This type sends an explicit message to a Smart3G Device. Sends <u>only 6</u> Control Words in the Send Buffer to the destination device. Note that each Control Word sent will be truncated to fit in the range of 0-255.

<table>
<tr><th></th><th>Control Word</th><th>Expected Value</th></tr>
<tr><td>Send Address</td><td>CW:20</td><td>1-250. Last digit of IP Address.</td></tr>
<tr><td>Send Buffer</td><td>CW:22 - CW:29</td><td>• SmartASI – Any integer
• Smart3G – 0-255</td></tr>
<tr><td>Receive Address</td><td>CW:30</td><td>1-250. Last digit of IP Address.</td></tr>
<tr><td>Receive Type</td><td>CW:31</td><td>Specifies type of sender:
• SmartASI – 80
• Smart3G – 67</td></tr>
<tr><td>Receive Buffer</td><td>CW:32 - CW:39</td><td>• SmartASI – Any integer
• Smart3G – 0-255</td></tr>
</table>
<table>
<tr><th></th><th>Control Bit</th><th>Expected Value</th></tr>
<tr><td>Receive Flag</td><td>CB:10</td><td>Set to true when a new message is received. Clearing this bit is left to the user.</td></tr>
</table>
</td></tr>
</table>

VII. Protocols

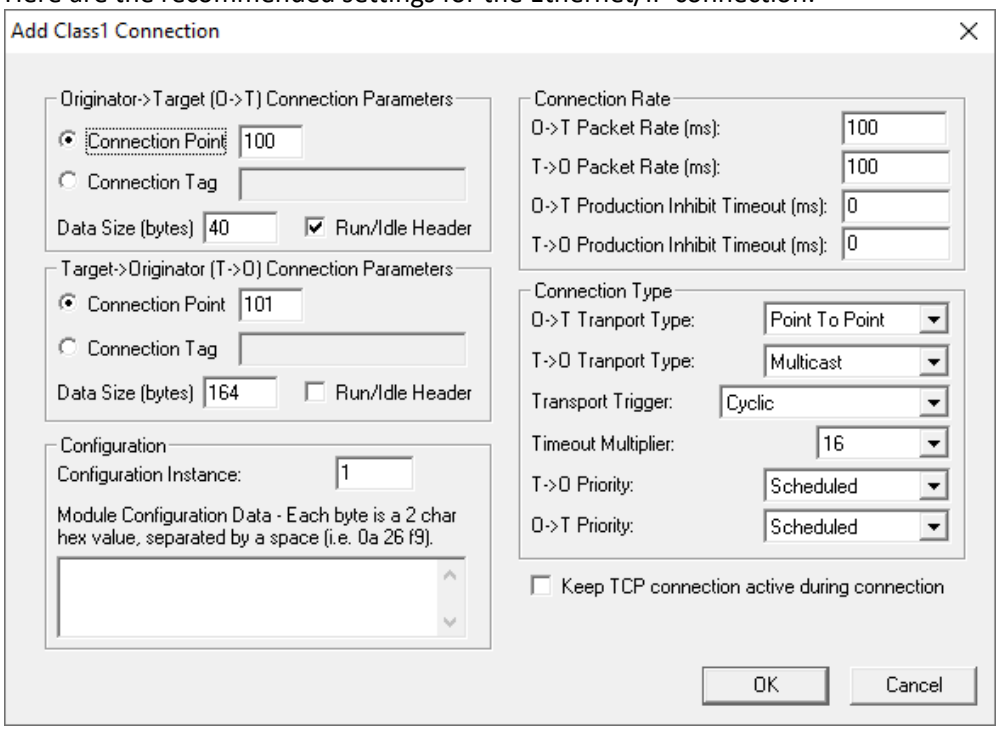
When the SmartASI is in Gateway Mode, it will not run any ladder logic. Instead all ASI cards on the network will be commanded by a connected Master PLC.

When the SmartASI is in PLC Mode, it will run user defined ladder programs. The state of all ASI cards on the network will be commanded by these ladder programs. A Master PLC may be connected to and exchange arbitrary data with the SmartASI for use within the ladder programs.

The supported protocols for these Master Connections are described and explained on the following pages.

Ethernet/IP

Here are the recommended settings for the Ethernet/IP connection:



Message Format

To SmartASI (Gateway: 31 bytes / PLC: 40 bytes)

(Gateway) Bytes 0 - 30	1 byte per ASI Slave: • Bit 0-3: Slave Output States • Bit 4-7: Slave Parameter (Commanded Analog)
(PLC) Bytes 0 - 39	Copied directly to Control Words 40-49 (4 bytes per CW)

From SmartASI (Gateway: 124 bytes / PLC: 164 bytes)

Bytes 0 - 61	2 byte status per ASI Slave: • Bit 0: Slave Online (Detected) • Bit 1: Slave Configured (Projected) • Bits 2-13: Out-of-range bits for slave data (Ranges configurable in Slave Profile)
Bytes 62-123	2 bytes IO per ASI Slave: • Bit 0 -3: Slave Input States • Bit 4-7: Slave Actual Parameter (Actual Analog) • Bit 8-11: Slave Output States • Bit 12-15: Slave Commanded Parameter (Commanded Analog)
(PLC Only) Bytes 124 - 163	Copied directly from Control Words 50-59 (4 bytes per CW)