

# **VSLogix**

# **User Guide**

**Document Revision 1.22**

(Updated September 10, 2024)

**© 2024 Vital Systems Inc**

**Buford, GA USA**

For more information please visit the product web page:

<https://www.vitalsystem.com/IntellECAT>

## Contents

License Agreement.....	4
I. Introduction .....	5
II. Basic Concepts .....	6
III. User Interface.....	7
Menu Bar .....	9
Project Explorer Window .....	10
Ladder Library Window.....	11
Ladder Editor.....	12
Example Ladder.....	15
Toolbox Window .....	16
Cross Reference .....	17
Debug Window .....	18
Output Window .....	18
Conveyor Layout Window.....	19
Master Settings Window.....	21
Init Params Editor .....	21
Device Editor .....	22
EtherCAT Network Browser .....	25
Connection Dialogue.....	26
Device Status Window .....	27
IV. Workflow .....	30
V. Files Addresses.....	32
Local File Types .....	33
Binary Files .....	35
Register Files .....	35
Float Files .....	36
Timer Files .....	36
Counter Files .....	38
Interlock Files .....	39
Device File Types.....	40
Inputs .....	41
Outputs .....	42

Input Parameter.....	43
Output Parameter.....	43
Global File Types.....	44
Control Words.....	44
Control Bits.....	45
Global Binary Files.....	45
Global Register Files.....	46
VI. Ladder Commands.....	47
Input Commands.....	48
Normally Open Command.....	48
Normally Closed Command.....	48
Compare Command.....	49
Output Commands.....	50
Output Command.....	50
Timer/Counter Command.....	51
Move Command.....	54
Math Command.....	55
Reset Command.....	56
Send Message Command.....	57
VII. Protocols.....	58
Ethernet/IP.....	59
VIII. Example.....	60

## 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.

VLogix 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.

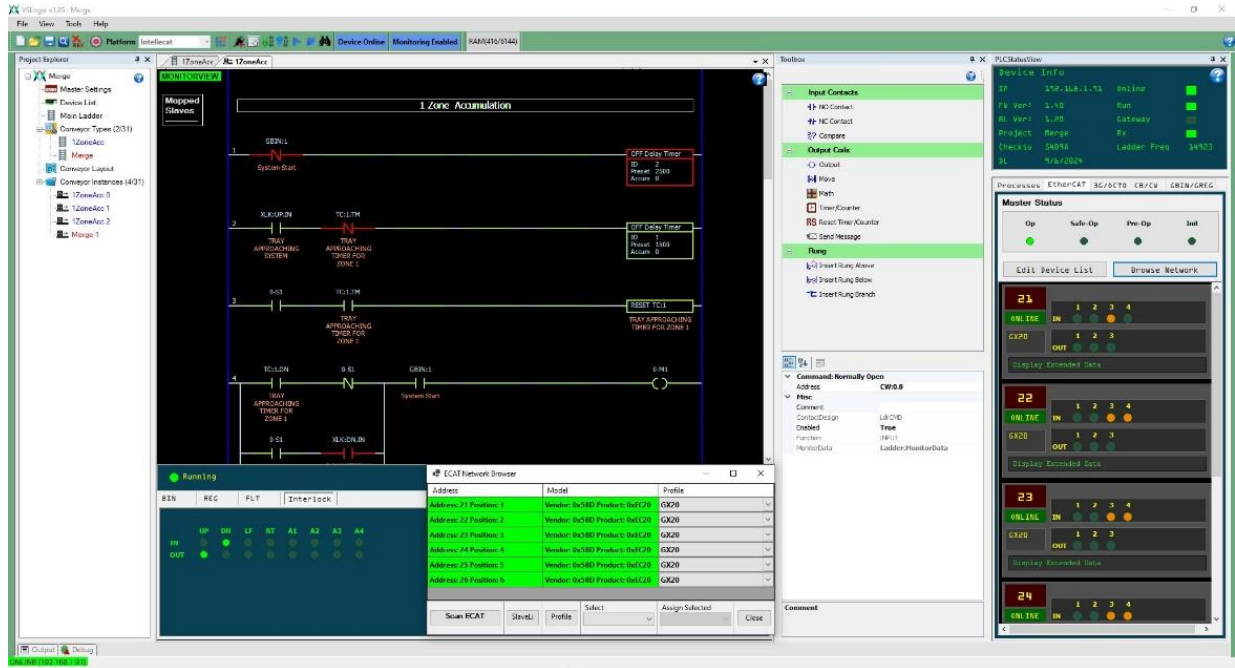
It is the nature of all mechanical and electrical systems to be hazardous. In order to be permitted to use VLogix with any machine you must agree to the following terms:

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 VLogix. I understand that VLogix 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 IntelLECAT PLC/Gateway. It has been developed to meet industrial programming standards for many specialized applications. VSLogix also serves as a direct interface to IntelLECAT devices and their connected network. It allows for configuration and real-time monitoring of ladder programs, as well as the current status and I/O of all connected GX20 devices.

## VSLogix Features:

- Create robust, powerful, and reusable ladder-logic programs and download them to devices.
- Upload and modify existing ladder-logic programs directly from IntelLECAT devices.
- Scan and Configure GX20 network from VSLogix software using IntelLECAT device 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.

## IntelLECAT Features:

- In Gateway mode, provides Ethernet access to an GX20 network of up to 31 GX20 devices
- In PLC mode, can run many ladder programs controlling up to 31 GX20 devices or 10 Smart3G devices

## Supported Network Hardware:

GX20	GX20 has two 24V Motor outputs M8 5-Pin plugs with speed & direction control, and two photo-eye M8 4-Pin plugs. Gx20 Sends Voltage, Current, I/O states, and Speed data back to the IntelLECAT Gateway for use in Ladder Programs or by a Master PLC (Rockwell etc).
ECS1	Multifunction I/O board with screw terminals (24V I/O, Encoder, Analog In/Out)
Smart3G	Useful where you need screw terminals for 24V digital I/O (8 in, 8 out)
Octo24	IP68 Hardware for liquid-proof I/O control 24V digital I/O (8 in, 8 out)

## II. Basic Concepts

The INTELCAT PLC/Gateway can control up to 31 GX20 Devices and up to 10 Smart3G/OCTO devices at once. How these devices will be controlled depends on which of two modes the INTELCAT is in.

### Gateway Mode

In Gateway Mode, Devices on the network will be commanded by a connected Master PLC. No ladder programs will run on the INTELCAT itself. The INTELCAT will simply act as a gateway to the GX20 network for a connected Ethernet/IP Master. You must still configure the INTELCAT's device list and download the project to the device. Use of the Network 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 for the Master PLC.

### PLC Mode

In PLC Mode, Devices on the network will be commanded by Ladder Programs ran on the INTELCAT itself. These are created using the [Ladder Editor](#). The INTELCAT can store up to 32 different Ladder Programs at a time. There are two types of Ladder Programs:

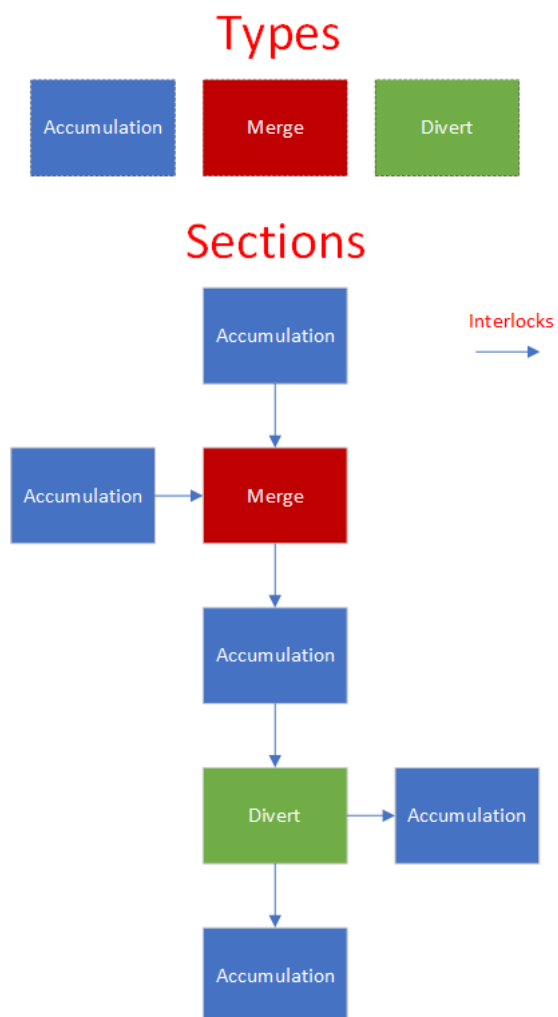
#### Main Ladder

There is one Main Ladder in every project. The Main Ladder commands devices directly. It can be used to develop conditions to be used globally in every conveyor section (e.g. System Start/Stop condition).

#### Conveyor Type (Ladder Template)

Conveyor Types are Ladder Programs that can be instantiated multiple times as **Conveyor Sections**. Conveyor Types use placeholders instead of directly referring to device addresses. These placeholders are mapped to actual addresses when a Conveyor Section is created from that type. This allows the user to create a few different ladder programs and then reuse them to design an entire conveyor line in a graphical view.

For example, one might create Accumulation, Merge, and Divert programs and assemble them into a layout as seen on the right.



## Conveyor Sections

Conveyor Sections are instances created from different Conveyor Types. In each conveyor section, you can map the devices that will be used in that section. You can connect different conveyor sections using interlock. The IntelLECAT can have up to 31 Conveyor Sections at one time. Each Conveyor Section is capable of controlling any number of GX20 devices. However, you cannot exceed the total max number of devices controllable by the IntelLECAT (31 GX20, 10 Smart3G/OCTO).

## Conveyor Layout

In Conveyor Layout window, you assemble different conveyor types and assign GX20 devices in them to make conveyor sections and use interlock to complete the flow of conveyors.

## III. User Interface

The screenshot displays the VSLogix v1.05: VSI software interface. The main window shows a ladder logic diagram for a '1 Zone Accumulation' conveyor section. The diagram consists of four rungs (1-4) and a sub-diagram for 'Start Up Run'.

- Rung 1:** A normally open contact labeled 'GBIN:1' is connected to a coil labeled 'OFF Delay Timer'. The timer parameters are ID: 2, Preset: 2500, and Accum: 0.
- Rung 2:** A normally open contact labeled 'XLK:UP.IN' and a normally closed contact labeled 'TC:1.TM' are connected to a coil labeled 'OFF Delay Timer'. The timer parameters are ID: 1, Preset: 1500, and Accum: 0.
- Rung 3:** A normally open contact labeled '0-S1' and a normally open contact labeled 'TC:L.TM' are connected to a coil labeled 'RESET TC:1'. The timer parameters are ID: 1, Preset: 1500, and Accum: 0.
- Rung 4:** A normally open contact labeled 'TC:L.DN', a normally open contact labeled '0-S1', and a normally open contact labeled 'GBIN:1' are connected to a coil labeled '0-M1'. The coil is labeled 'Motor 1'.

Below rung 4, there is a sub-diagram for 'Start Up Run' with the following components:

- A normally open contact labeled '0-S1' and a normally open contact labeled 'XLK:DN.IN' are connected to a coil labeled 'DOWNSTREAM CONTROLLER READY'.
- A normally open contact labeled 'Photo Eye 1' and a normally open contact labeled '0-S1' are connected to a coil labeled 'Start Up Run'.
- A normally open contact labeled 'Photo Eye 1' and a normally open contact labeled 'TC:2.TM' are connected to a coil labeled 'Photo Eye 1'.

The interface includes a Project Explorer on the left showing the hierarchy: VSI > Master Settings > Device List > Main Ladder > Conveyor Types (1/31) > 1ZoneAcc > Conveyor Layout > Conveyor Instances (2) > 1ZoneAcc 0 > 1ZoneAcc 1.

The right side of the interface features a Toolbox with 'Input Contacts' (NO Contact, NC Contact, Compare) and 'Output Coils' (Output, Move). Below the toolbox is the PLCStatusView window, which displays 'Device Info' and 'Processes'.

**Device Info:**

TP	IP	Status	...
1.40	192.16	Online	
1.40		Run	
1.20		Gateway	
VSI		Rx	
8CFB4		Ladder	#51
9/9/20			

**Processes:**

Instance	Status	Mapped S1
Main	Running	--
1ZoneAcc 0	Running	E21,
Merge 0	Running	E24, E25,

The status bar at the bottom left shows 'ONLINE 192.168.1.311'.

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. [Device Editor Window](#)
13. [Device 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                     | 10. Ethercat Network Browser           |
| 2. Open Project                           | 11. Download Current Project to Device |
| 3. Save Project                           | 12. Upload Project from Device         |
| 4. Find Window                            | 13. Run All Ladder Programs on Device  |
| 5. <a href="#">Cross Reference Window</a> | 14. Stop All Ladder Programs on Device |
| 6. <a href="#">Master Settings Window</a> | 15. Enable Monitoring                  |
| 7. Select Platform                        | 16. Device Connection State            |
| 8. Compile Ladders                        | 17. Device Monitoring State            |
| 9. <a href="#">Connect To Device</a>      | 18. Memory Consumption                 |

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

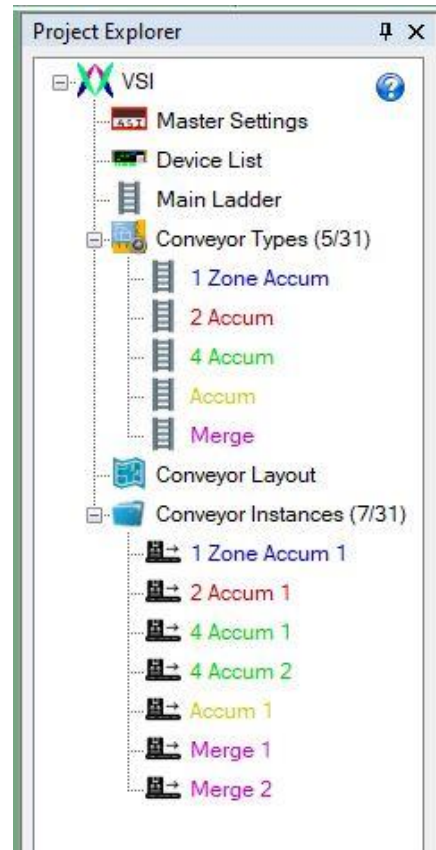
- 1-3 allow creating/opening/saving projects.
- 4 and 5 are used to search through the ladder programs in the project.
- 6 opens the [Master Settings Window](#) for editing global settings.
- 7 is the Platform Selection. The Platform is used to select the intended device the project will be running on. 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.
- 10-15 require an active connection to be used.
- 16-17 display the state of the active connection.
- 18 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 device.

## Project Explorer Window

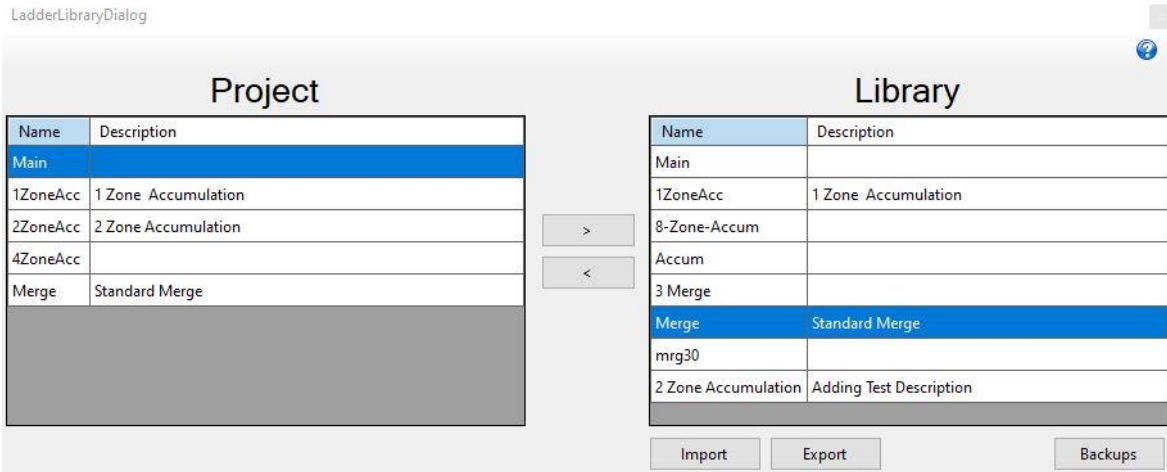
The Project Explorer Window is the basic navigation and management window for the application. It is open by default, but can be re-opened from the View Menu if closed.

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, monitor the Main ladder and edit device map and parameters if a IntelLECAT is [connected to VSLogix](#).
- Master Settings Node – Opens [Master Settings window](#), allowing editing of global project settings.
- Conveyor Layout Node – Double clicking/Right clicking then select Open Conveyor Layout will open a Window that allows creation and configuring of Conveyor Sections
- Conveyor Types Node - Allows creation and management of Conveyor Types. Double clicking this node will open the [Ladder Library](#) Window.
  - Right click on the Conveyor Types node gives option to create new conveyor type, import conveyor type from another project and view ladder library.
  - 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.



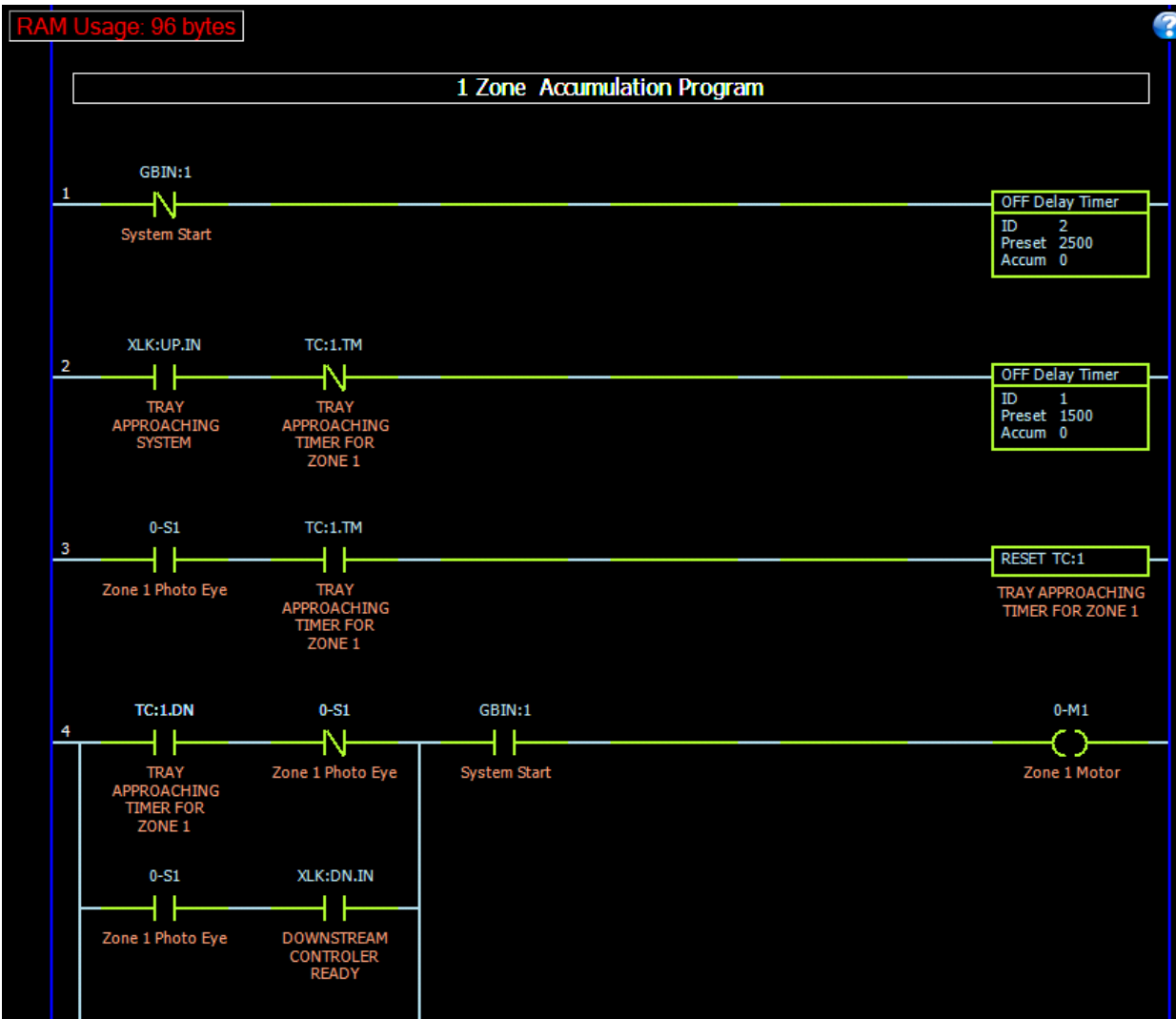
## Ladder Library Window



The Ladder Library can be used to store ladder programs for use in later projects or retrieve ladder programs from previous 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 (.vslgx), 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.

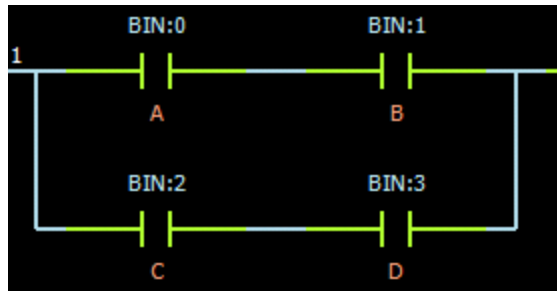
## Ladder Editor



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 device.

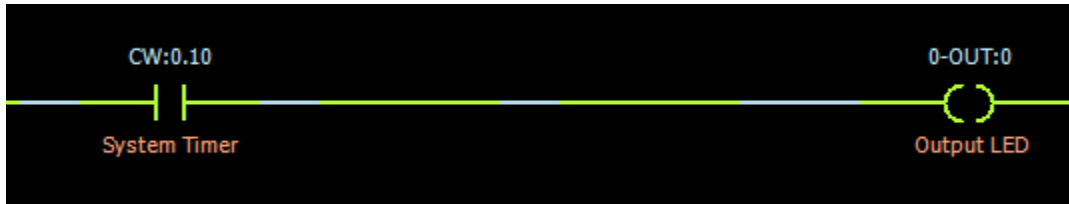
Below is a table listing useful input commands within the Ladder Editor.

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

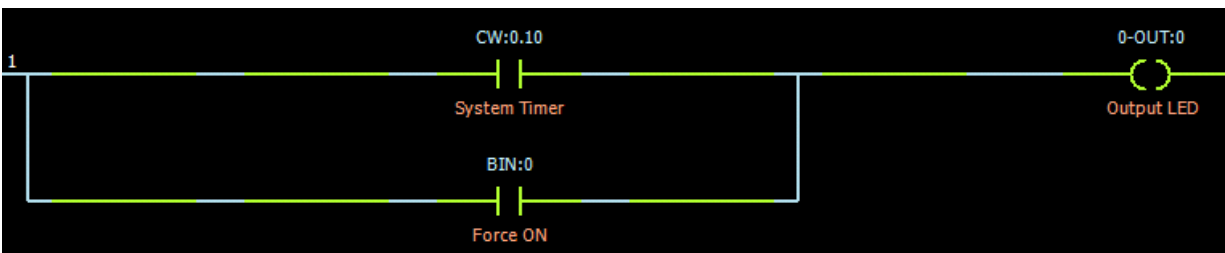
Using the above information of how the Ladder Editor works, we will create an example Ladder to demonstrate the creation process. Let's start by creating a simple flashing LED:



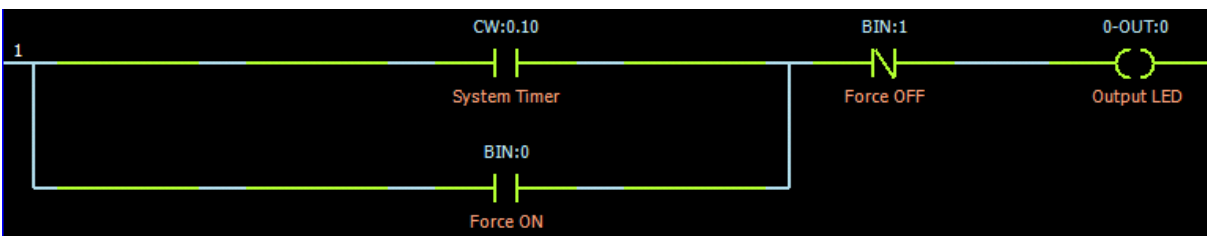
On the left we have a [Normally Open](#) command with the address CW:0.10, and on the right we have an [Output](#) command with the address 0-OUT:0. A more detailed explanation of these addresses can be found in the File Address section, but simply put:

- The Address CW:0.10 refers to the 10<sup>th</sup> bit of Control Word 0, the System Timer. Since the system timer counts up in milliseconds, the 10<sup>th</sup> bit will toggle every  $2^{10}$  (aka 1024) milliseconds, 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 Device 0. Please refer to the [Device 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.

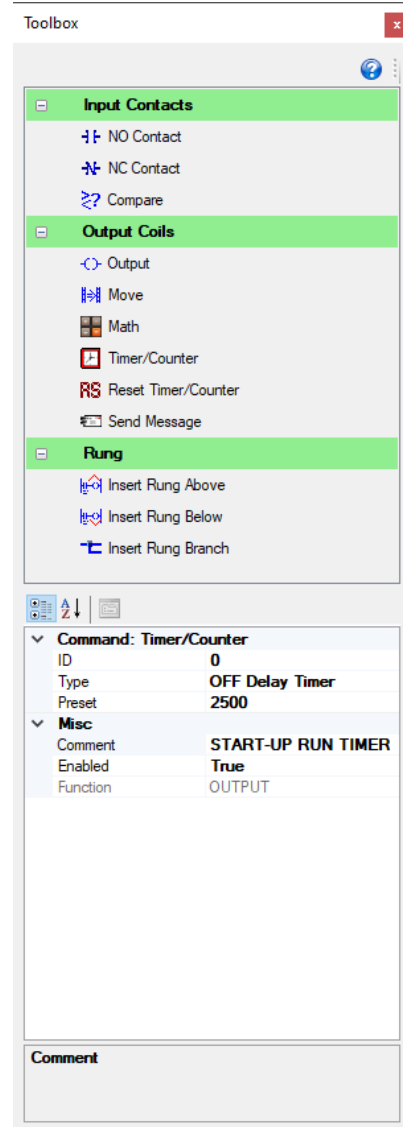
## Toolbox Window

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

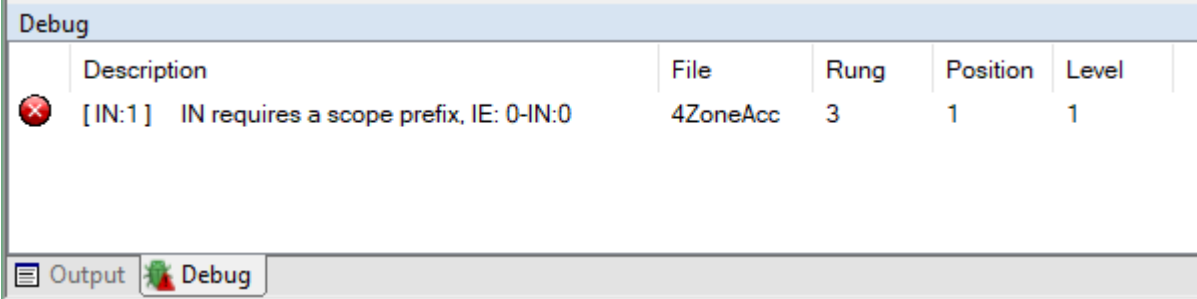
File Type	Files in Use
GBIN	0,
TC_AC	0, 1, 2, 3,
XLK	0,
TC	0, 1, 2, 3,
IN	0,
OUT	0,
OUTEX	1,
CW	0,
BIN	0,

Address	Comment	Location	Ladder Program
TC:2.TM	Tray Appr for Merge Zone	Rung:11, Position:2, ...	Merge
TC:2.TM	Tray Appr for Merge Zone	Rung:12, Position:2, ...	Merge
TC:3		Rung:7, Position:6, ...	Merge
TC:3.AC		Rung:6, Position:6, ...	Merge
TC:3.DN	Tray Approching for zon...	Rung:8, Position:1, ...	Merge
TC:3.TM	Tray Approching for zon...	Rung:6, Position:3, ...	Merge
TC:3.TM	Tray Approching for zon...	Rung:7, Position:2, ...	Merge
TC:3.TM	Tray Approching for zon...	Rung:9, Position:2, ...	Merge
XLK:DN.IN	DOWNSTREAM CONTR...	Rung:4, Position:2, ...	1ZoneAcc
XLK:DN.IN	mainline downstream is ...	Rung:8, Position:2, ...	Merge
XLK:DN.OUT	TRAY EXITING CONTROL...	Rung:6, Position:6, ...	1ZoneAcc
XLK:DN.OUT	Tray Exiting Controller	Rung:1, Position:6, ...	Merge
XLK:LF.IN	Tray Approching from ...	Rung:10, Position:1, ...	Merge

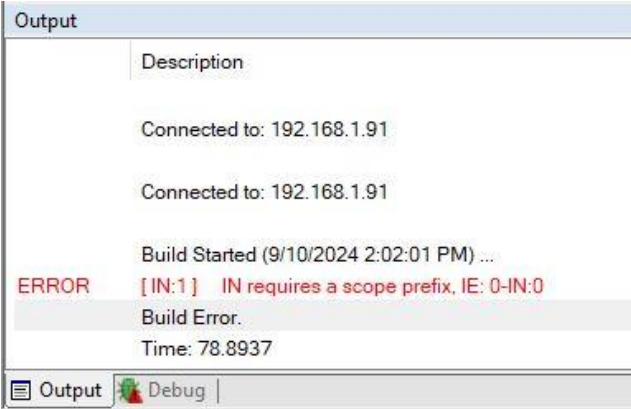
The Cross Reference View provides a list of which Device Files are used in the current project. It also lists which rung and Ladder Program is referencing the file. You can jump directly to that usage by double clicking it.

## Debug Window



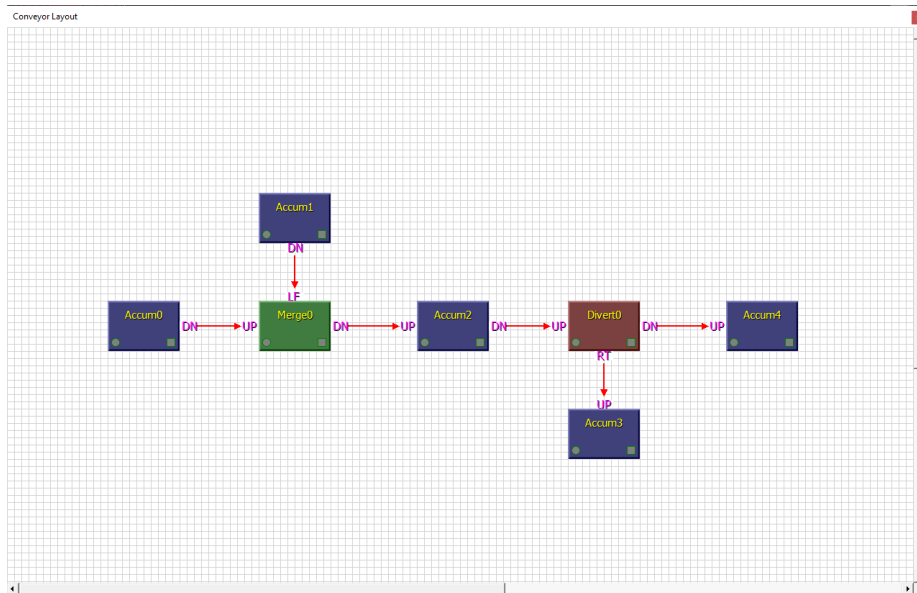
The Debug Window displays compilation errors for the ladder program (if there are any) after the compilation process. Error entries can be clicked to jump to the source of the error. The Debug Window can be accessed by clicking the Debug Window Icon under the menu bar, otherwise it pops-up when errors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

## Output Window

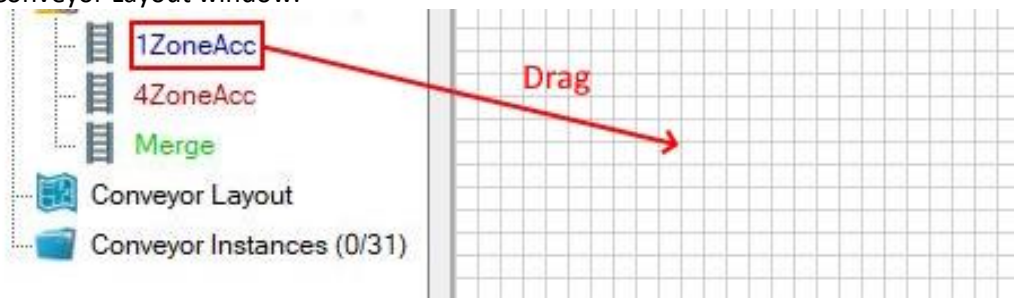


The Output Window provides detailed and technical feedback such as application log messages, compile messages, and errors. The Output Window can be accessed by clicking the Output Window Icon under the menu bar, otherwise it pops-up automatically when necessary.

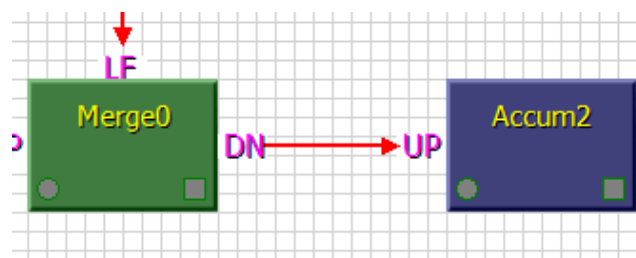
## Conveyor Layout Window



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. These allow the Ladder Sections to pass data directly to one another. To do this, first select one section, then press a key to choose a port. Then select a different section you want to connect it to, and press a key to choose its port. The available port choices 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.

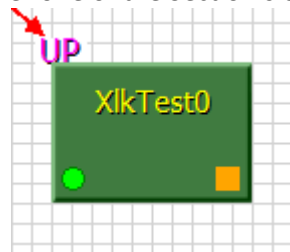


Note that interlocks simply allow the transfer of data from one section to another. It is up to the designer of the ladder program to choose what data to send, and how to process it. Within the Ladder Program, this Interlock data is accessed using the [XLK Filetype](#).

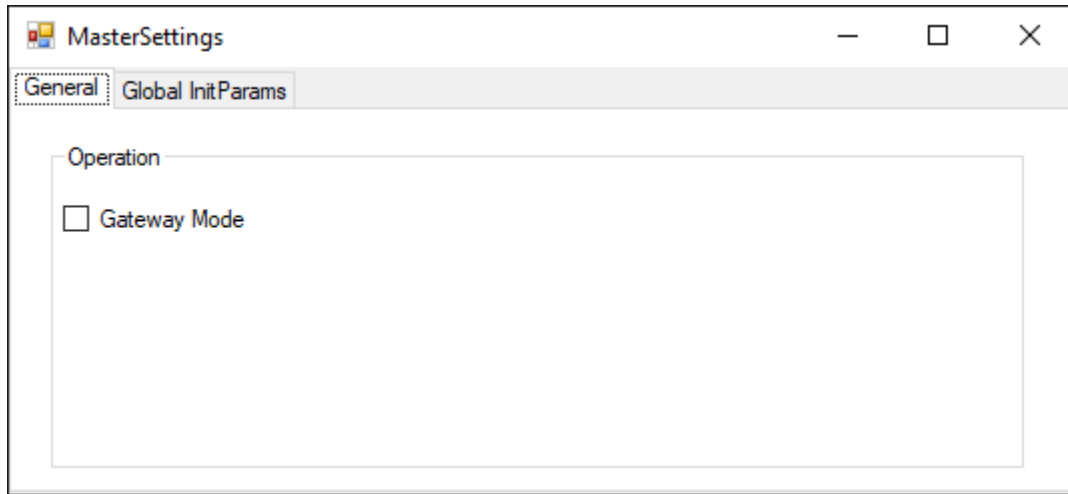


You can also right click the Conveyor Sections to [edit their Device Maps](#), to edit their [Initial Parameters](#), or to delete them. You can also select monitor, or double click the section to open its monitoring window. Note that Monitoring the section's execution in real-time will require an active connection to the IntelECAT device. You will also need to enable monitoring mode, which will itself require a matching project to be loaded on the device.

When Monitoring Mode is enabled, the Conveyor Layout window will display very basic information about the Conveyor Sections. The left Green LED will display whether the Section is currently running. The right orange LED will blink whenever one of the Section's outputs changes a value.

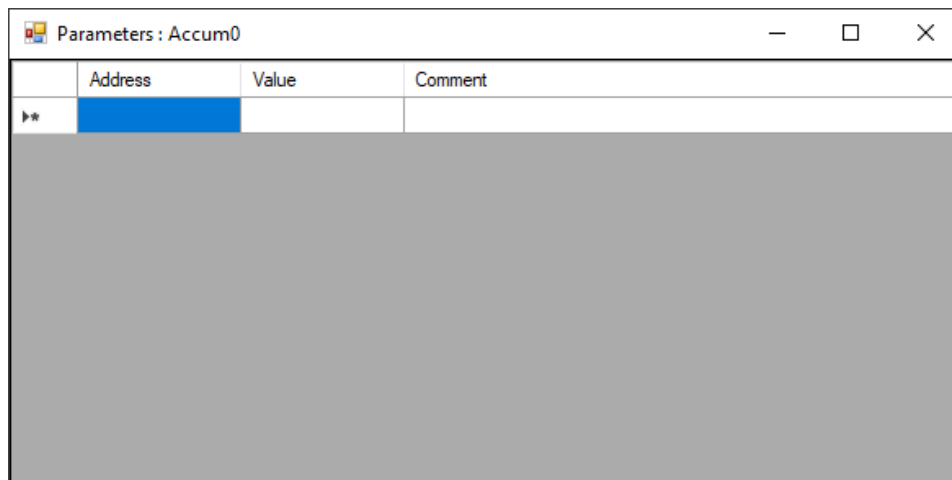


## Master Settings Window



The Master Settings window is used to configure global settings for the device. 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.

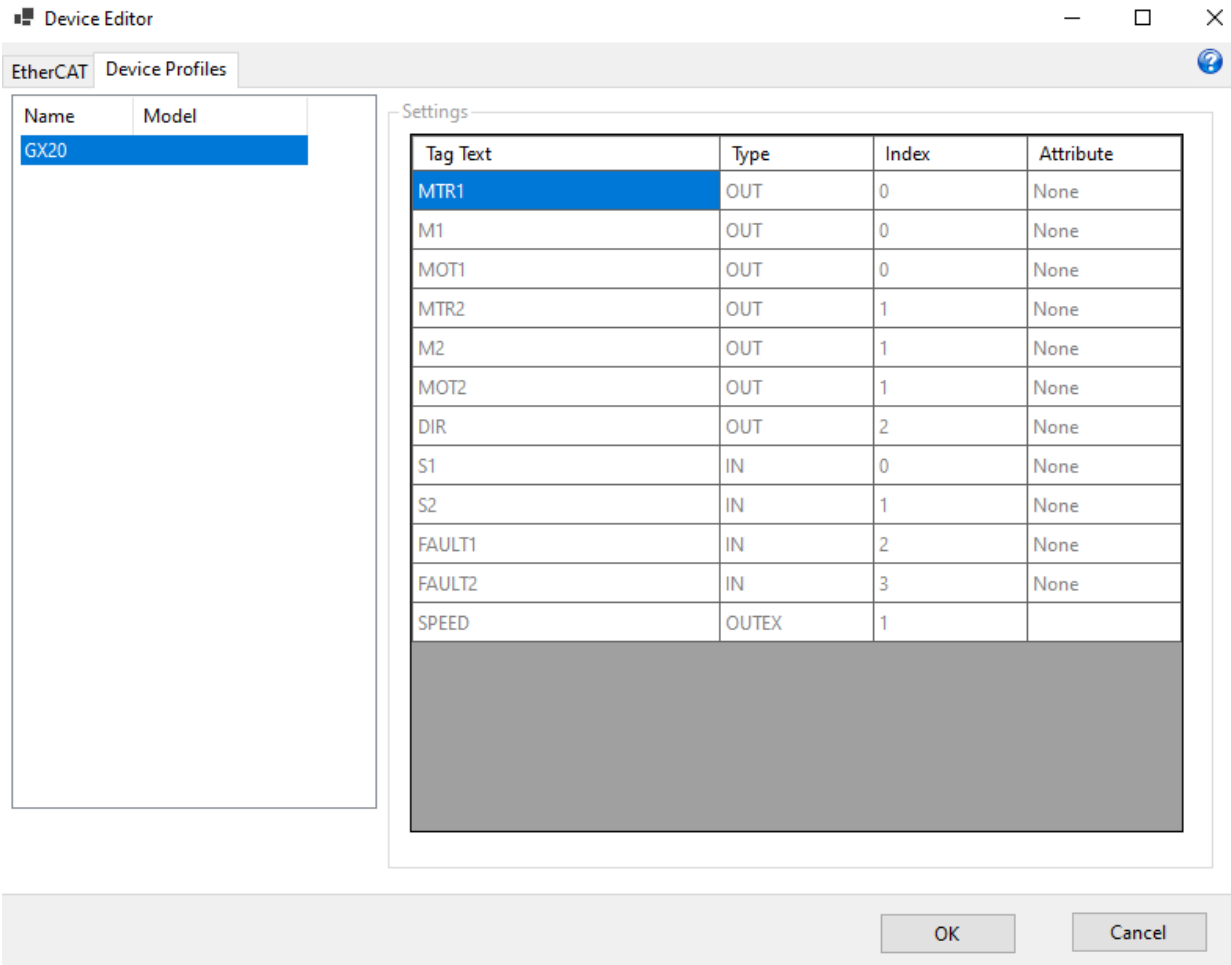
There are two different Init Param editors available:

- 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.

## Device Editor

You can open this window from the Tools dropdown on the menu bar, by right clicking a Conveyor Section and clicking “Edit Device Map” or from the GX20 Status Tab. This window is used to configure the devices in your project, and will be used in every project. Refer to the [Device File](#) section to see how Devices should be accessed within ladder programs.

### Device Profiles Tab



The screenshot shows the 'Device Editor' window with the 'Device Profiles' tab selected. The window title bar includes 'Device Editor' and standard window controls. The 'EtherCAT' tab is also visible. The main content area is divided into two sections:

- Left Section:** A table with columns 'Name' and 'Model'. The 'Name' column contains 'GX20', which is highlighted in blue.
- Right Section:** A 'Settings' table with columns 'Tag Text', 'Type', 'Index', and 'Attribute'. The table contains the following data:

Tag Text	Type	Index	Attribute
MTR1	OUT	0	None
M1	OUT	0	None
MOT1	OUT	0	None
MTR2	OUT	1	None
M2	OUT	1	None
MOT2	OUT	1	None
DIR	OUT	2	None
S1	IN	0	None
S2	IN	1	None
FAULT1	IN	2	None
FAULT2	IN	3	None
SPEED	OUTEX	1	

At the bottom of the window, there are 'OK' and 'Cancel' buttons.

In the Device Profiles tab, you can see profiles for device model. Currently, this is used to find the acceptable Tag Text for different type of I/O and their respective index values of GX20 Devices.

## EtherCAT Tab

	Profile	Description	
16			Clear
17			Clear
18			Clear
19			Clear
20			Clear
21	GX20	Accum 1 Card	Clear
22	GX20	Accum 2 Card	Clear
23	GX20	Merge 1 Card 1	Clear
24	GX20	Merge 2 Card 2	Clear
25	GX20	Accum 3 Card	Clear
26	GX20	Not Used	Clear
27			Clear
28			Clear
29			Clear
30			Clear
31			Clear
32			Clear
33			Clear

The EtherCAT tab allows you to configure which GX20 devices will do data exchange with the IntelLECAT. Any device that has a profile selected will be searched for by the IntelLECAT, and will perform data exchange if it is found. The profile's model should match the type of GX20 device you intend to connect to. On the left of the tab, you can see the device IDs. If you are using GX20 with ID 21, you need to set the profile next to ID21 as GX20. Thus, IntelLECAT will communicate with the GX20 device which has the above ID. If ID doesn't match, there will be no communication.

## Device Map Tab

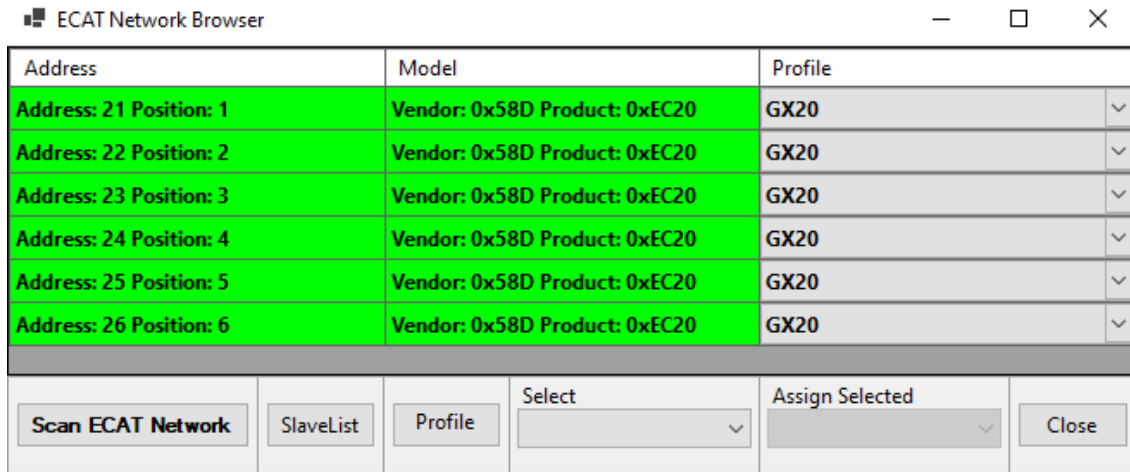
	Device
▶ Device-0	GX20-23: Merge 1 Card 1
Device-1	GX20-24: Merge 1 Card 2
Device-2	GX20-25: Merge 1 Card 3

The Device Map tab is only visible when the Device Editor is opened by selecting “Edit Device Map” after right clicking a conveyor section. It is used to choose the devices mapped to a conveyor section. All device placeholders within the ladder program will appear on the left. On the right, you can choose real GX20 devices from the Device List 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 devices.

Refer to the [Device File](#) section to see how Devices should be accessed within ladder programs.



## EtherCAT Network Browser



The screenshot shows the 'ECAT Network Browser' window. It features a table with three columns: 'Address', 'Model', and 'Profile'. The table contains six rows of device information, all highlighted in green. Below the table is a control panel with several buttons and dropdown menus.

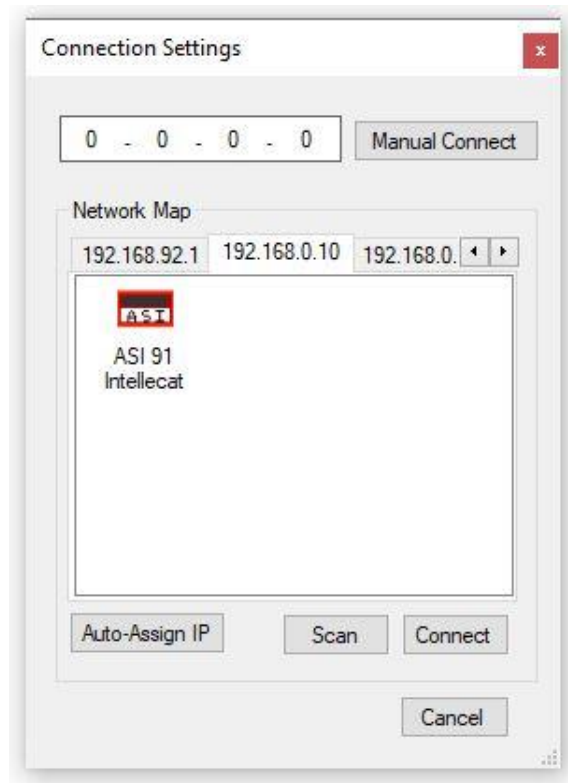
Address	Model	Profile
Address: 21 Position: 1	Vendor: 0x58D Product: 0xEC20	GX20
Address: 22 Position: 2	Vendor: 0x58D Product: 0xEC20	GX20
Address: 23 Position: 3	Vendor: 0x58D Product: 0xEC20	GX20
Address: 24 Position: 4	Vendor: 0x58D Product: 0xEC20	GX20
Address: 25 Position: 5	Vendor: 0x58D Product: 0xEC20	GX20
Address: 26 Position: 6	Vendor: 0x58D Product: 0xEC20	GX20

Control Panel:

- Scan ECAT Network
- SlaveList
- Profile
- Select (dropdown menu)
- Assign Selected (dropdown menu)
- Close

The EtherCAT Network Browser can only be accessed after a IntelIECAT device is connected to VSLogix via the [Connection Dialogue](#). The network will be scanned and all devices on the network will be displayed in this list. Displayed will be the address of the device, the Model of the device, and finally the profile for that device (GX20/Smart3G).

## Connection Dialogue



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

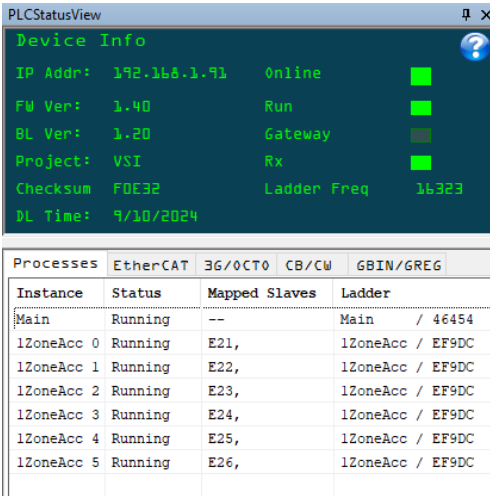
Click the [Connect Button](#) to make the Connection Dialogue appear. The connection dialogue will scan all available network adapters and create a list of all detected devices. You can either double click, or select the device and click Connect to connect to a device. If the desired device is not listed 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 device 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:



The screenshot shows the PLCStatusView window. The top section, titled "Device Info", displays the following information:

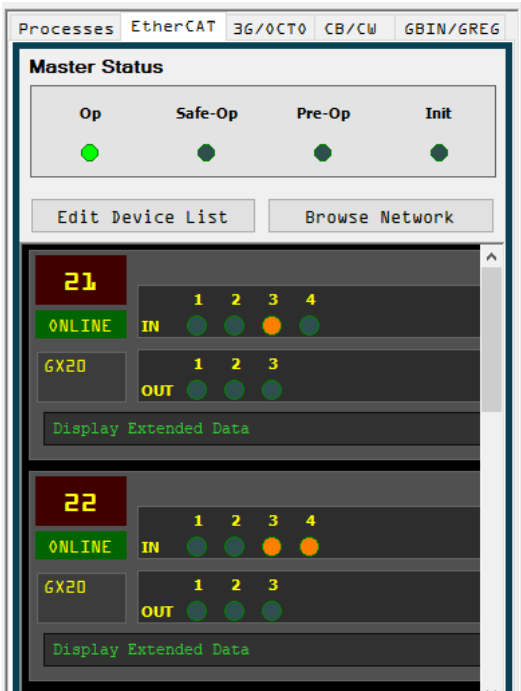
- IP Addr: 192.168.1.91 Online (Green status indicator)
- FW Ver: 1.40 Run (Green status indicator)
- BL Ver: 1.20 Gateway (Yellow status indicator)
- Project: VSI Rx (Green status indicator)
- Checksum F0E32 Ladder Freq 16323
- DL Time: 9/10/2024

The bottom section is a table with 5 tabs: "Processes", "EtherCAT", "3G/0CT0", "CB/CW", and "GBIN/GREG". The "Processes" tab is active and shows the following data:

Instance	Status	Mapped Slaves	Ladder
Main	Running	--	Main / 46454
lZoneAcc 0	Running	E21,	lZoneAcc / EF9DC
lZoneAcc 1	Running	E22,	lZoneAcc / EF9DC
lZoneAcc 2	Running	E23,	lZoneAcc / EF9DC
lZoneAcc 3	Running	E24,	lZoneAcc / EF9DC
lZoneAcc 4	Running	E25,	lZoneAcc / EF9DC
lZoneAcc 5	Running	E26,	lZoneAcc / EF9DC

## Processes

This tab displays all the conveyor sections in the project currently loaded on the device, as well as their Status (whether they're currently running or not), and the GX20 mapped to that section. You may start or stop an individual section, or monitor its execution by right clicking it. At the right of the tab are all the Ladder Programs in the project, as well as their checksums.



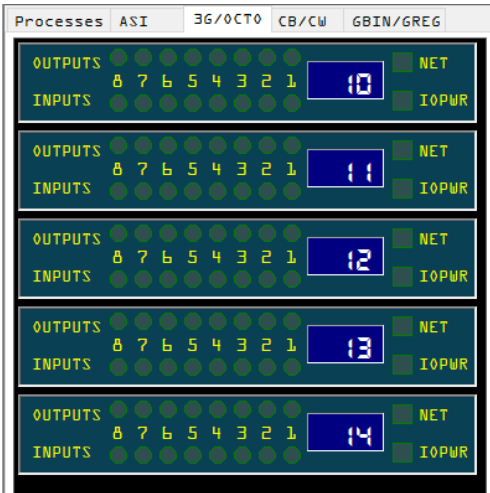
## EtherCAT

The GX20 tab displays the GX20 status of the IntelLECAT master, as well as the status and IO of all the configured devices in the connected device's Device List. 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 Device Override by right clicking the device. Creating a Device Override will allow you to override the output regardless of the program's execution.

From this tab you may also open the Device Editor for the current project by clicking Edit Device List, or open the ECAT [Network Browser](#) by clicking Browse Network.

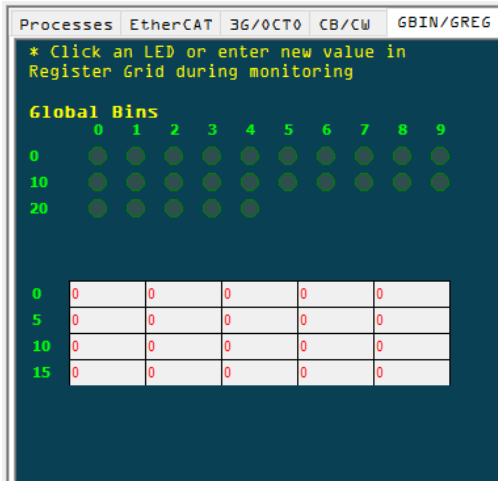
Here is a description of each of the Master Status bits:

- **OP** – In this state, full process data exchange is possible between the master and its connected devices. Both inputs and outputs are active, and the system is fully functional. This is the state in which normal EtherCAT communication and control occur.
- **Safe-Op** – The device is ready to exchange process data but will not affect its outputs (they are in a safe state, typically off or held). Inputs can be read, but the outputs are disabled for safety reasons. This state is used to ensure that inputs are valid before switching to the fully operational state.
- **Pre-Op** – In this state, the device is partially operational. It can communicate with the master but cannot yet exchange process data. Configuration parameters and initialization commands can be exchanged between the master and the device. This is where the master configures the connected devices.
- **Init** – This is the starting state after the device powers up or gets reset. In this state, the device is initializing its hardware and software. No communication with the master occurs in this state, and no process data exchange is possible.



### 3G/OCTO

The 3G/OCTO tab allows you to view the Status and IO of the connected device's VSI Devices, which were configured from the [VitalSys Device Tab](#). Similar to the GX20 tab, you may toggle the Device'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 device. 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 Device FileTypes in the [Device Files section](#). This will be necessary for any VSLogix project.

Example Ladders are available on new installations in the [Ladder Library](#).

### 2. Configuring DeviceList

Open the Device [Editor](#) from the Tools menu. On the EtherCAT tab, select the devices you are using in their respective addresses. For example, if you are using GX20 device which has ID 22, select profile 22 and assign GX20 device there. Selecting a matching profile for a device will add it to the INTELCAT's data exchange scan list.

In the Device Profiles tab, you can see the name of the devices and when you select a device, you can select the tag text applicable to that device.

### 3. Creating Conveyor Sections

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

You can right click on Conveyor Types node and create new ladder program and design your ladder program in the ladder editor. Under Conveyor Types node, you can create 31 different conveyor types. These conveyor types can be template of conveyor sections that are repeated in your conveyor project. For example, if your entire conveyor project requires, 3 similar merge sections, you can create one merge template under Conveyor Type node. You can then use this merge template in Conveyor Layout window and create three merge sections by placing it in the correct location of the Conveyor Layout.






To use the templates of conveyor types, open the [Conveyor Layout window](#) by double clicking the Conveyor Layout node and 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 Device map	This will open the Device Editor with an extra tab for the Section's Device map. Any Device Placeholders in the ladder program will appear here in order to be mapped to an GX20 or VSI device that was configured in the Device Editor.
Right Click -> Edit Init Params	This will open the <a href="#">Init Param Editor</a> . 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 <a href="#">Conveyor Layout section</a> .

You can rename any conveyor section by Right Click > rename option. You can also run and monitor any specific conveyor section by Right Click and selecting Run Ladder or Monitor respectively.

#### 4. Going Online

After connecting to the device via the [Connection Dialogue](#), the [Status window](#) will automatically begin to monitor the state of the device. You will be able to see the IO of the GX20 devices 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. When a device has a project downloaded in it, it will 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 device 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 device.)
-  **Run** – Enables the execution of all Conveyor Sections on the device. Sections can also be Run individually by right clicking them and selecting the Run Ladder option from the pop up menu.
-  **Stop** – Disables the execution of all Conveyor Sections on the device. Sections can also be Stopped individually by right clicking them and selecting the Stop Ladder option from the pop up 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. Clicking the Monitor button while the project on the device does not match the one loaded in VSLogix will download the project to the device, then enable monitoring mode.

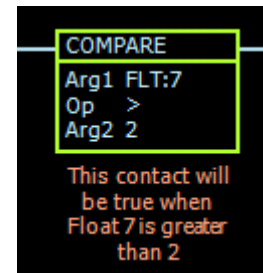
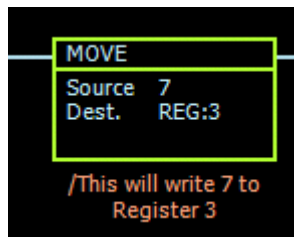
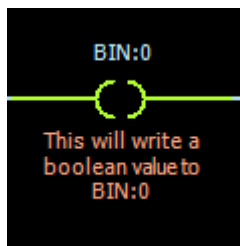
Once online, you will also now be able to open the [EtherCAT network browser](#) from the EtherCAT tab of the Status window. This will allow you to view all GX20 devices on the EtherCAT network, whether they have been configured in the project or not.

## V. Files Addresses

IntelECAT 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
- [Device Files](#) – Requires Scope to identify the device being accessed
  - [IN](#) – Digital Input on specified Device
  - [OUT](#) – Digital Output on specified Device
  - INEX – Extended Input
  - OUTEX – Extended Output
- [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 Device 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:**

<b>Description</b>	General Information and notes on the file type.
<b>Keyword</b>	The keyword syntax used to reference the file.
<b>Scope</b>	Defines the rules about where and how the file can be accessed. Has a value of either Local, Device, or Global.
<b>Attributes</b>	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.
<b>Format</b>	The Format address syntax used to reference a specific file in the Ladder Program.
<b>Usage</b>	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 [Device 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



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

## Register Files



<b>Description</b>	<p>A 32-bit signed integer. Referred to plainly as “Register” in VSLogix.</p> <div style="border: 1px solid black; background-color: #FFD700; padding: 5px; margin: 5px 0;"> <p><b>NOTE:</b> 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).</p> </div>
<b>Keyword</b>	REG
<b>Scope</b>	Local
<b>Data Type</b>	32-bit Signed Integer
<b>Usable Attributes</b>	<ul style="list-style-type: none"> <li>• None</li> <li>• Bit Indexing [0 – 31] (becomes Boolean type)</li> </ul>
<b>Format</b>	REG:<file index>.<bit index>
<b>Usage</b>	Read, Write

## Float Files

```
Math
FLT:30 = sqrt
(FLT:20 * 20)
```

<b>Description</b>	A 32-bit signed floating-point value that can use fractional values following a decimal point.
<b>Keyword</b>	FLT
<b>Scope</b>	Local
<b>Data Type</b>	32-bit Floating Point
<b>Usable Attributes</b>	<ul style="list-style-type: none"> <li>None</li> </ul>
<b>Format</b>	FLT:<file index>
<b>Usage</b>	Read, Write

## Timer Files

```
ON Delay Timer
ID 0
Preset 2000
Accum 0
```

```
RESET TC:1
```

<b>Description</b>	<p>Timers that can keep time in a resolution of milliseconds. The timer keeps time until the user-defined preset value is reached.</p> <p>Unlike other FileTypes, each Timer/Counter File corresponds to an associated <a href="#">Timer/Counter command</a> in the Ladder program. You cannot use a Timer/Counter file index if there is not a Command with that ID.</p> <p>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.</p>
<b>Keyword</b>	TC
<b>Scope</b>	Local
<b>Data Type</b>	32-Bit Signed Integer / Boolean

<b>Usable Attributes</b>	<ul style="list-style-type: none"><li>• <b><u>None/AC</u></b> – Accumulated Time in milliseconds (int value).</li><li>• <b><u>PR</u></b> – Preset Value in milliseconds (int value).</li><li>• <b><u>EN</u></b> – Timer Enabled (bool value).</li><li>• <b><u>DN</u></b> – Done Timing (bool value).</li><li>• <b><u>TM</u></b> – Currently Timing (bool value).</li></ul>
<b>Format</b>	TC:<file index>.<attribute>
<b>Usage</b>	Read, Write (PR Only)

## Counter Files

UP Counter	
ID	1
Preset	300
Accum	0

RESET TC:1
------------

<b>Description</b>	<p>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.</p> <p>Unlike other FileTypes, each Timer/Counter File corresponds to an associated <a href="#">Timer/Counter command</a> in the Ladder program. You cannot use a Timer/Counter file index if there is not a Command with that ID.</p> <p>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.</p>
<b>Keyword</b>	TC
<b>Scope</b>	Local
<b>Data Type</b>	32-Bit Signed Integer / Boolean
<b>Usable Attributes</b>	<ul style="list-style-type: none"> <li>• <b>None/AC</b> – Accumulated Counts (int value).</li> <li>• <b>PR</b> – Preset Value (int value).</li> <li>• <b>EN</b> – Timer Enabled (bool value).</li> <li>• <b>DN</b> – Done Timing (bool value).</li> </ul>
<b>Format</b>	TC:<file index>.<attribute>
<b>Usage</b>	Read, Write (PR Only)

## Interlock Files



<b>Description</b>	Used to send or receive a single bit between two connected <a href="#">Conveyor Sections</a> . 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.
<b>Keyword</b>	XLK
<b>Scope</b>	Local (Unusable in Main Ladder)
<b>Data Type</b>	Boolean
<b>Usable Attributes</b>	<ul style="list-style-type: none"> <li>• <b>IN</b> – Read bit coming from the connected Section (bool value)</li> <li>• <b>OUT</b> – Read/Write bit going to the connected Section (bool value)</li> </ul>
<b>Format</b>	XLK:<interlock specifier>.<attribute>
<b>Usage</b>	Read, Write (OUT only)
<b>Example</b>	<p>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 <a href="#">ConveyorLayout</a> window, Section2's output will be activated when Section1's input is activated.</p>

## Device File Types

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

*Device-Keyword:Index*

So for example:

0-OUT:1

This would access Output 1 of Device 0.

The acceptable device prefixes and their meaning are dependent on whether they're being used in either the [Main Ladder](#) or a [Conveyor Type](#).

### Main Ladder

1-99	The values from 1 to 99 directly refer to a particular GX20 MDR Control card.
3G#, OCTO#, V#	Using either the prefix of 3G, OCTO, or V followed by a number from 0 to 9 will refer to a Vital Systems device (Smart3G/OCTO). All three prefixes are functionally the same. It is up to preference which is used. The address for the specified device should be set to match the actual Device ID (rotary switch) in the Device <a href="#">Editor</a> window. For example, if you want to use a Smart3G device with Address 80, you can use the File Address V0-OUT:1, then in the Master Settings window, set VSI Device 0 to have address 80.

### Conveyor Type

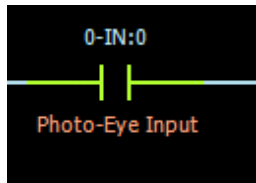
0-63, 3G#, OCTO#, V#	<p>Within a conveyor type, all prefixes have the same meaning. The number refers to a placeholder which will need to be mapped to an actual device using the Device <a href="#">Map Window</a>. This was done so that a single Conveyor Type can be instantiated multiple times and yet control different Devices.</p> <p>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 DeviceMap Window on one section, I could map the Placeholder 0 to GX20 Device 5, while in the other I could map it to Smart3G Device 7.</p>
-------------------------------	---

Device Files:

- [IN](#) – Digital Input on specified Device
- [OUT](#) – Digital Output on specified Device
- INEX – Extended Input
- OUTEX – Extended Output

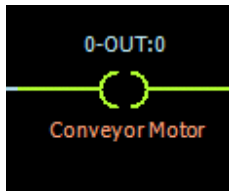


## Inputs



<b>Description</b>	Input state of the selected device device (similar to BIN values in usage).
<b>Keyword</b>	IN
<b>Scope</b>	Device
<b>Data Type</b>	Boolean
<b>Usable Attributes</b>	<ul style="list-style-type: none"> <li>None</li> <li>Full - Directly access the File's containing Byte (becomes Byte type)</li> </ul>
<b>Format</b>	<code>&lt;device&gt;-IN:&lt;fileIndex&gt;</code>
<b>Usage</b>	Read
<b>Example</b>	<p>In the below diagram from the <a href="#">Interlock page</a>, 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 Device <a href="#">Map</a> window for Section1, Device-0 has been mapped to GX20-21. Therefore, 0-IN:0 will correspond to Input 0 of GX20 Device 21.</p>

## Outputs



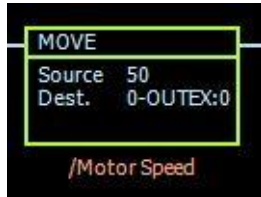
<b>Description</b>	Output state of the selected device device (similar to BIN values in usage).
<b>Keyword</b>	OUT
<b>Scope</b>	Device
<b>Data Type</b>	Boolean
<b>Usable Attributes</b>	<ul style="list-style-type: none"> <li>• None</li> <li>• Full - Directly access the File's containing Byte (becomes Byte type)</li> </ul>
<b>Format</b>	<device>-OUT:<file index>
<b>Usage</b>	Read, Write
<b>Example</b>	<p>In the below diagram from the <a href="#">Interlock page</a>, 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 Device <a href="#">Map</a> window for Section2, Device-0 has been mapped to GX20-23. Therefore, 0-OUT:0 will correspond to Output 0 of GX20 Device 23.</p> <p>The diagram illustrates the interlock logic between two sections. Section1 (left) contains a Photo-Eye Input (0-IN:0) and a Send Package Ready output (XLK:DN.OUT). Section2 (right) contains a Receive Package Ready input (XLK:UP.IN) and a Conveyor Motor output (0-OUT:0). A red arrow labeled 'DN' points from Section1 to Section2, and another red arrow labeled 'UP' points from Section2 to Section1. Below the diagram are two device mapping boxes: 'Device-0 GX20-21' and 'Device-0 GX20-23'.</p>

## Input Parameter



<b>Description</b>	A read-only value that reflects the actual confirmed parameter value of an GX20 Device. For GX20, INEX 1 represents Motor 1 current in MilliAmps.
<b>Keyword</b>	INEX
<b>Scope</b>	Device
<b>Data Type</b>	Byte
<b>Usable Attributes</b>	<ul style="list-style-type: none"> <li>• None</li> <li>• Bit Indexing [0 – 31] (becomes Boolean type)</li> </ul>
<b>Format</b>	<device>-INEX:<fileIndex>
<b>Usage</b>	Read

## Output Parameter



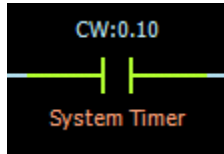
<b>Description</b>	Represents the commanded value of the parameter for an GX20 Device. For GX20, this corresponds to the Motor Speed. The Source value determines the percentage of full speed.
<b>Keyword</b>	OUTEX
<b>Scope</b>	Device
<b>Usable Attributes</b>	<ul style="list-style-type: none"> <li>• None</li> <li>• Bit Indexing [0 – 31] (becomes Boolean type)</li> </ul>
<b>Format</b>	<device>- OUTEX:<file index>
<b>Usage</b>	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.

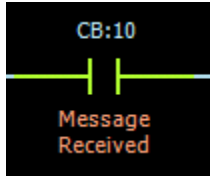
- [CB](#) – Control Bit (Boolean type with pre-defined function)
- [CW](#) – Control Words (Signed 32-bit Integer with pre-defined function)
- [GBIN](#) – Global Boolean value
- [GREG](#) – Global Signed 32-bit Integer

### Control Words



<b>Description</b>	Special purpose 32-bit signed integers for device control. Each one has a unique predefined function.																												
<b>Keyword</b>	CW																												
<b>Scope</b>	Global																												
<b>Data Type</b>	32-Bit Signed Integer																												
<b>Usable Attributes</b>	<ul style="list-style-type: none"> <li>• None</li> <li>• Bit Indexing [0 – 31] (Becomes Boolean type)</li> </ul>																												
<b>Format</b>	CW:<file index>.<attribute>																												
<b>Usage</b>	<table border="1"> <thead> <tr> <th>CW</th> <th>Usage</th> <th>Description</th> </tr> </thead> <tbody> <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, IntelleCAT : 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 <a href="#">Protocols</a> section)</td> </tr> <tr> <td>50-59</td> <td>R/W</td> <td>Data to Ethernet/IP Master (Refer to <a href="#">Protocols</a> section)</td> </tr> </tbody> </table> <p><i>*Unmentioned CW indexes are currently reserved</i></p>		CW	Usage	Description	0	R	System Millisecond Timer. Counts up from 0 starting when entering run mode	20	R/W	Send Message Address (1-250)	22-29	R/W	Send Message Data	30	R	Receive Message Address (1-250)	31	R	Receive Message Type (Smart3G : 67, IntelleCAT : 80)	32-39	R	Receive Message Data	40-49	R	Data from Ethernet/IP Master (Refer to <a href="#">Protocols</a> section)	50-59	R/W	Data to Ethernet/IP Master (Refer to <a href="#">Protocols</a> section)
CW	Usage	Description																											
0	R	System Millisecond Timer. Counts up from 0 starting when entering run mode																											
20	R/W	Send Message Address (1-250)																											
22-29	R/W	Send Message Data																											
30	R	Receive Message Address (1-250)																											
31	R	Receive Message Type (Smart3G : 67, IntelleCAT : 80)																											
32-39	R	Receive Message Data																											
40-49	R	Data from Ethernet/IP Master (Refer to <a href="#">Protocols</a> section)																											
50-59	R/W	Data to Ethernet/IP Master (Refer to <a href="#">Protocols</a> section)																											
<b>Example</b>	The below example is using a Normally Open command with the 10 <sup>th</sup> bit of CW:0, aka the System Timer. This setup will toggle state once every 2 <sup>^</sup> 10 (1024) milliseconds.																												

## Control Bits



<b>Description</b>	Special purpose boolean values for device control. Each one has a predefined function.						
<b>Keyword</b>	CB						
<b>Scope</b>	Global						
<b>Data Type</b>	Boolean						
<b>Usable Attributes</b>	<ul style="list-style-type: none"> <li>None</li> <li>Full - Directly access the File's containing Integer (becomes Integer type)</li> </ul>						
<b>Format</b>	CB:<file index>						
<b>Usage</b>	<table border="1"> <thead> <tr> <th>CB</th> <th>Usage</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>10</td> <td>R/W</td> <td>Message Received Bit. Left to the user to clear it.</td> </tr> </tbody> </table> <p><i>*Unmentioned CB indexes are currently reserved</i></p>	CB	Usage	Description	10	R/W	Message Received Bit. Left to the user to clear it.
CB	Usage	Description					
10	R/W	Message Received Bit. Left to the user to clear it.					

## Global Binary Files



<b>Description</b>	A global bit value that can be 0 or 1. The INTELLCAT has a maximum limit of 24 GBINs at once.
<b>Keyword</b>	GBIN
<b>Scope</b>	Global
<b>Data Type</b>	Boolean
<b>Usable Attributes</b>	<ul style="list-style-type: none"> <li>None</li> <li>Full - Directly access the File's containing Integer (becomes Integer type)</li> </ul>
<b>Format</b>	GBIN:<file index>
<b>Usage</b>	Read, Write

## Global Register Files



<b>Description</b>	A global 32-bit signed integer. The IntelleCAT has a maximum limit of 20 GREGs at once.
<b>Keyword</b>	GREG
<b>Scope</b>	Global
<b>Data Type</b>	32-Bit Signed Integer
<b>Usable Attributes</b>	<ul style="list-style-type: none"><li>• None</li><li>• Bit Indexing [0 – 31] (Becomes Boolean type)</li></ul>
<b>Format</b>	GREG:<file index>.<bit index>
<b>Usage</b>	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



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

### Normally Closed Command

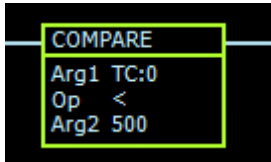


<b>Description</b>	An input command whose condition is true when the addressed bit value is inactive.
<b>Type</b>	Input
<b>Parameters</b>	<b>Address</b> – the referenced bit to read from (Ex. <i>IN:1</i> , <i>OUT:2</i> , <i>TC:2.TM</i> ).
<b>File Types</b>	All Boolean Types. (BIN, IN, OUT, CB, Register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



## Compare Command

```
Compare
Arg1 REG:5
Op >
Arg2 REG:15
```

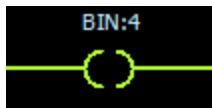
<b>Description</b>	A command whose condition depends on the logical comparison of the values of two referenced files.
<b>Type</b>	Input
<b>Parameters</b>	<ul style="list-style-type: none"> <li>• <b>Arg1</b> – The first argument (referenced file or constant) for comparison. (Ex. 10, REG:5, "text", BAR:8,10).</li> <li>• <b>Arg2</b> – The second argument (referenced file or constant) for comparison. (Ex. 10, REG:5, "text", BAR:8,10).</li> <li>• <b>Operation</b> – The logical comparison to make between the 2 arguments. <ul style="list-style-type: none"> <li>➤ <b>Equal ( = )</b> – true if the 2 arguments are equal.</li> <li>➤ <b>Not Equal( != )</b> – true if the 2 arguments are not equal.</li> <li>➤ <b>Greater Than ( &gt; )</b> – true if Arg1 is greater than Arg2.</li> <li>➤ <b>Less Than ( &lt; )</b> – true if Arg1 is less than Arg2.</li> <li>➤ <b>Greater Than or Equal to ( &gt;= )</b> – true if Arg1 is greater than or equal to Arg2.</li> <li>➤ <b>Less Than or Equal to ( &lt;= )</b> – true if Arg1 is less than or equal to Arg2.</li> </ul> </li> </ul>
<b>File Types</b>	Numerical values (REG, FLT, CW, GREG). Timer/Counter .PR .AC Attributes.
<b>Examples</b>	 <pre>COMPARE Arg1 TC:0 Op &lt; Arg2 500</pre>

## 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](#)

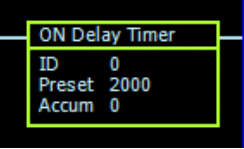
## Output Command



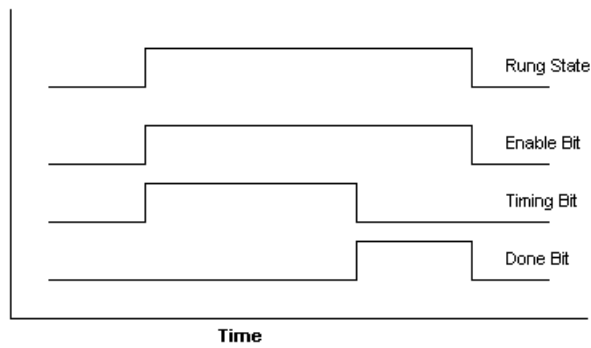
<b>Description</b>	This command sets the addressed bit to true or false. When the rung condition is true, the addressed bit or output is set to true (1 or high) and with rung condition false, the bit or output is set to false (0 or low).
<b>Type</b>	Output
<b>Parameters</b>	<ul style="list-style-type: none"> <li>• <b>Address</b> – the referenced bit value to write to (Ex. Out:1, REG:5.3, etc).</li> <li>• <b>Output Type</b> – Controls the behavior of the command. <ul style="list-style-type: none"> <li>➤ <b>Normal ( )</b> – If the rung is active, it writes 1, otherwise it writes 0. <div data-bbox="451 1436 662 1541" data-label="Image"> </div> </li> <li>➤ <b>Latch ( L )</b> – If the rung is active, it writes 1, otherwise it does nothing. <div data-bbox="451 1633 673 1724" data-label="Image"> </div> </li> <li>➤ <b>Unlatch ( U )</b> – If the rung is active, it writes 0, otherwise it does nothing.</li> </ul> </li> </ul>

	 <p>➤ <u>Latch Transition to True ( <b>U</b> )</u> – Writes 1 only when the current rung state becomes active and the previous rung state was inactive.</p>  <p>➤ <u>Latch Transition to False ( <b>V</b> )</u> – Writes 1 only when the current rung state becomes inactive and the previous rung state was active.</p>
<b>File Types</b>	Boolean files (BIN, OUT, CB, GBIN), Register Bits (REG:#.x, CW:#.x, GREG:#.x)

Timer/Counter Command

<b>Description</b>	<p>The Timer/Counter command makes use of either a Timer or Counter register for its functionality</p> <ul style="list-style-type: none"> <li>• <b>Timer</b> – The timer register keeps timing until the preset value (in milliseconds) is reached.</li> <li>• <b>Counter</b> – The counter register keeps counting until the reset value is reached.</li> </ul>
<b>Type</b>	Output
<b>Parameters</b>	 <ul style="list-style-type: none"> <li>• <b>ID</b> – The timer or counter that is bound to this command.</li> <li>• <b>Preset</b> – A 32-bit integer value that specifies when the command stops timing/counting. <i>For timers, this value is in milliseconds.</i></li> <li>• <b>Type</b> – Controls the behavior of the command.</li> </ul> <div style="background-color: #FFD700; padding: 5px; border: 1px solid black; margin: 10px 0;"> <p><b>NOTE:</b> Timers increment the “Accumulator” value every millisecond, while Counters increment the “Accumulator” value every time the rung state transitions from false to true.</p> </div> <p><b>Types:</b></p> <p><b>ON DELAY</b> – 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</p>

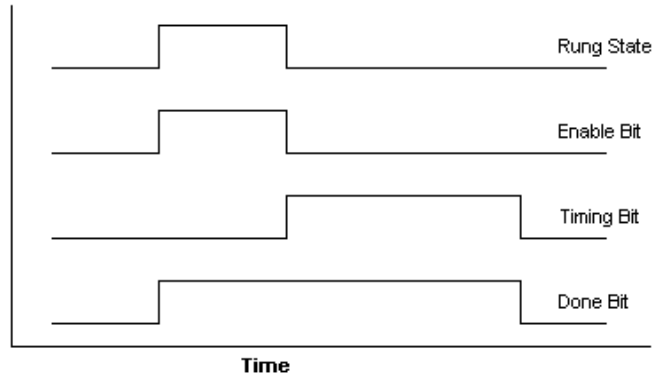
and Enable Bit are reset as soon as the rung state becomes false. The Accumulator is reset to 0 until the rung condition becomes true again.



**ON Delay Timing Diagram**

<b>OFF Delay Timer</b>	
ID	0
Preset	2000
Accum	0

**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.



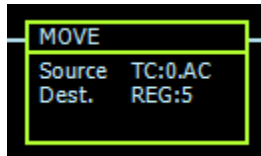
**OFF Delay Timing Diagram**

<b>UP Counter</b>	
ID	1
Preset	300
Accum	0

**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.

<b>File Types</b>	TC
-------------------	----

## Move Command



<b>Description</b>	Copies a specified source file's value (or a constant numerical value) into a specified destination file while the rung is active.
<b>Type</b>	Output
<b>Parameters</b>	<ul style="list-style-type: none"><li>• <b>Source</b> – The referenced file or constant value to be moved (Ex. 5000, 0.056, REG:9, CW:10, etc).</li><li>• <b>Destination</b> – The file where the source file's value is written (Ex. FLT:2, REG:9, CW:10, etc).</li></ul>
<b>File Types</b>	Numerical Values. (REG, FLT, CW, GREG)

## Math Command

```
Math
FLT:30 = sqrt
(FLT:20 * 20)
```

<b>Description</b>	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.
<b>Type</b>	Output
<b>Parameters</b>	<ul style="list-style-type: none"> <li>• <b>Arg1</b> – The first argument. (Ex. FLT:4, 5000, 4.556, REG:9, CW:10, etc)</li> <li>• <b>Arg2</b> – The second argument. (Ex. FLT:4, 5000, 4.556, REG:9, CW:10, etc)</li> <li>• <b>Destination</b> – Address of File to store the result. (Ex. FLT:4, REG:9, CW:10, etc)</li> <li>• <b>Binary Operation</b> – operation to perform between the two arguments. All Bit manipulation values must be done with REG (integer) values.</li> </ul> <div style="border: 1px solid black; background-color: #FFD700; padding: 5px; margin: 10px 0;"> <p><b>NOTE:</b> When using Float type for bitwise operations, the digits after decimal point are dropped, eg. FLT:4 BitAND 123.</p> </div> <ul style="list-style-type: none"> <li>➤ <i>None</i> – the result is the value of Arg1. Arg2 is ignored.</li> <li>➤ <i>Addition</i></li> <li>➤ <i>Subtraction</i></li> <li>➤ <i>Multiplication</i></li> <li>➤ <i>Division</i></li> <li>➤ <i>Power/Exponentiation</i></li> <li>➤ <i>BitAND</i> – Bitwise AND</li> <li>➤ <i>BitOR</i> – Bitwise OR</li> <li>➤ <i>BitXOR</i> – Bitwise Exclusive OR</li> <li>➤ <i>BitShiftLeft</i> – Shifts Arg1's bit value by a specified number of digits (Arg2's value) to the left.</li> <li>➤ <i>BitShiftRight</i> – Shifts Arg1's bit value by a specified number of digits (Arg2's value) to the right.</li> </ul> <ul style="list-style-type: none"> <li>• <b>Unary Operation</b> – operation to perform on the result of the Binary Operation. <ul style="list-style-type: none"> <li>➤ <i>None</i></li> <li>➤ <i>Negative</i> – Negates the value.</li> <li>➤ <i>Bitwise Inversion</i> – Invert the bit value.</li> <li>➤ <i>Absolute</i> – Absolute value.</li> </ul> </li> </ul>

	<ul style="list-style-type: none"> <li>➤ <i>Square Root</i></li> <li>➤ <i>Sine</i></li> <li>➤ <i>Cosine</i></li> <li>➤ <i>Tangent</i></li> <li>➤ <i>Cosecant</i></li> <li>➤ <i>Secant</i></li> <li>➤ <i>Cotangent</i></li> <li>➤ <i>Natural Logarithm</i></li> <li>➤ <i>Common Logarithm</i></li> </ul>
<b>File Types</b>	Numerical Types. (REG, FLT, CW)

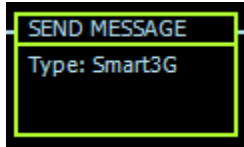
#### Reset Command



<b>Description</b>	Resets a timer or counter when the rung state transitions to true.
<b>Type</b>	Output
<b>Parameters</b>	<ul style="list-style-type: none"> <li>• <b>Address</b> – The Timer/Counter to reset. (Ex. TC:3, TC:7 etc)</li> </ul>
<b>File Types</b>	TC



## Send Message Command



<b>Description</b>	<p>Sends a message over Ethernet to another device 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.</p> <div style="border: 1px solid black; background-color: orange; padding: 5px; margin-top: 10px;"> <p><b>NOTE:</b> <i>This command only triggers when the rung state transitions to true.</i></p> </div>																									
<b>Type</b>	Output																									
<b>Parameters</b>	<ul style="list-style-type: none"> <li>• Type                         <ul style="list-style-type: none"> <li>➤ <b>IntelIECAT</b> – This type sends an explicit message to another <b>IntelIECAT Device</b>. Sends <b>all 8</b> Control Words in the Send Buffer to the destination device.</li> <li>➤ <b>Smart3G</b> – This type sends an explicit message to a <b>Smart3G Device</b>. Sends <b>only 6</b> 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.</li> </ul> </li> </ul> <table border="1" style="width: 100%; border-collapse: collapse; margin-top: 10px;"> <thead> <tr> <th style="width: 30%;"></th> <th style="width: 30%;">Control Word</th> <th style="width: 40%;">Expected Value</th> </tr> </thead> <tbody> <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> <ul style="list-style-type: none"> <li>• IntelIECAT – Any integer</li> <li>• Smart3G – 0-255</li> </ul> </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:                     <ul style="list-style-type: none"> <li>• IntelIECAT – 80</li> <li>• Smart3G – 67</li> </ul> </td> </tr> <tr> <td>Receive Buffer</td> <td>CW:32 - CW:39</td> <td> <ul style="list-style-type: none"> <li>• IntelIECAT – Any integer</li> <li>• Smart3G – 0-255</li> </ul> </td> </tr> </tbody> </table> <table border="1" style="width: 100%; border-collapse: collapse; margin-top: 10px;"> <thead> <tr> <th style="width: 30%;"></th> <th style="width: 30%;">Control Bit</th> <th style="width: 40%;">Expected Value</th> </tr> </thead> <tbody> <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> </tbody> </table>			Control Word	Expected Value	Send Address	CW:20	1-250. Last digit of IP Address.	Send Buffer	CW:22 - CW:29	<ul style="list-style-type: none"> <li>• IntelIECAT – Any integer</li> <li>• Smart3G – 0-255</li> </ul>	Receive Address	CW:30	1-250. Last digit of IP Address.	Receive Type	CW:31	Specifies type of sender: <ul style="list-style-type: none"> <li>• IntelIECAT – 80</li> <li>• Smart3G – 67</li> </ul>	Receive Buffer	CW:32 - CW:39	<ul style="list-style-type: none"> <li>• IntelIECAT – Any integer</li> <li>• Smart3G – 0-255</li> </ul>		Control Bit	Expected Value	Receive Flag	CB:10	Set to true when a new message is received. Clearing this bit is left to the user.
	Control Word	Expected Value																								
Send Address	CW:20	1-250. Last digit of IP Address.																								
Send Buffer	CW:22 - CW:29	<ul style="list-style-type: none"> <li>• IntelIECAT – Any integer</li> <li>• Smart3G – 0-255</li> </ul>																								
Receive Address	CW:30	1-250. Last digit of IP Address.																								
Receive Type	CW:31	Specifies type of sender: <ul style="list-style-type: none"> <li>• IntelIECAT – 80</li> <li>• Smart3G – 67</li> </ul>																								
Receive Buffer	CW:32 - CW:39	<ul style="list-style-type: none"> <li>• IntelIECAT – Any integer</li> <li>• Smart3G – 0-255</li> </ul>																								
	Control Bit	Expected Value																								
Receive Flag	CB:10	Set to true when a new message is received. Clearing this bit is left to the user.																								

## VII. Protocols

When the IntelIECAT is in Gateway Mode, it will not run any ladder logic. Instead all GX20 devices on the network will be commanded by a connected Master PLC.

When the IntelIECAT is in PLC Mode, it will run user defined ladder programs. The state of all GX20 devices on the network will be commanded by these ladder programs. A Master PLC may be connected to and exchange arbitrary data with the IntelIECAT 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:

The screenshot shows the 'Add Class1 Connection' dialog box with the following settings:

- Originator->Target (O->T) Connection Parameters:**
  - Connection Point: 100
  - Connection Tag: (empty)
  - Data Size (bytes): 40
  - Run/Idle Header:
- Target->Originator (T->O) Connection Parameters:**
  - Connection Point: 101
  - Connection Tag: (empty)
  - Data Size (bytes): 164
  - Run/Idle Header:
- Configuration:**
  - Configuration Instance: 1
  - Module Configuration Data - Each byte is a 2 char hex value, separated by a space (i.e. 0a 26 f9).
  - (Empty text box)
- Connection Rate:**
  - O->T Packet Rate (ms): 100
  - T->O Packet Rate (ms): 100
  - O->T Production Inhibit Timeout (ms): 0
  - T->O Production Inhibit Timeout (ms): 0
- Connection Type:**
  - O->T Transport Type: Point To Point
  - T->O Transport Type: Multicast
  - Transport Trigger: Cyclic
  - Timeout Multiplier: 16
  - T->O Priority: Scheduled
  - O->T Priority: Scheduled
- Keep TCP connection active during connection

Buttons: OK, Cancel

### Message Format

#### To IntelLECAT (Gateway: 31 bytes / PLC: 40 bytes)

(Gateway) Bytes 0 - 30	1 byte per GX20 Device: <ul style="list-style-type: none"> <li>• Bit 0-3: Device Output States</li> <li>• Bit 4-7: Device Parameter (Commanded Analog)</li> </ul>
(PLC) Bytes 0 - 39	Copied directly to Control Words 40-49 (4 bytes per CW)

#### From IntelLECAT (Gateway: 124 bytes / PLC: 164 bytes)

Bytes 0 – 123 (4 Bytes per Device)	2 byte status: <ul style="list-style-type: none"> <li>• Bit 0: Device Online (Detected)</li> <li>• Bit 1: Device Configured (Projected)</li> <li>• Bits 2-13: Out-of-range bits for device data (Ranges configurable in Device Profile)</li> </ul>
	2 bytes IO: <ul style="list-style-type: none"> <li>• Bit 0-3: Device Input States</li> <li>• Bit 4-7: Device Actual Parameter (Actual SpeedCode)</li> <li>• Bit 8-11: Device Output States</li> <li>• Bit 12-15: Device Commanded Parameter (Commanded SpeedCode)</li> </ul>
(PLC Only) Bytes 124 - 163	Copied directly from Control Words 50-59 (4 bytes per CW)

## VIII. Example

### Getting Started:

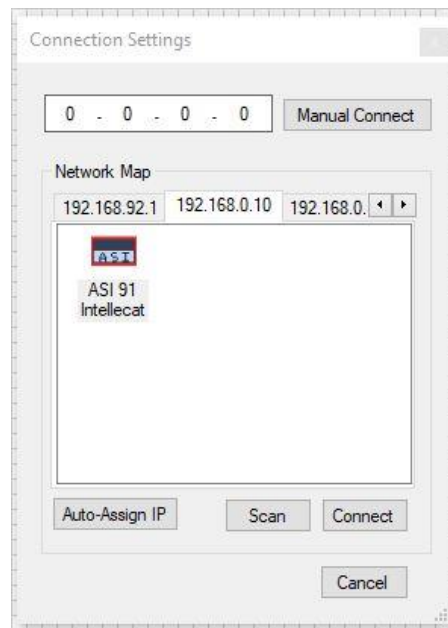
Step 1:

Connect IntelLECAT with your PC using EtherNet cable > Power it up with 24v > Connect GX20 devices with IntelLECAT Gateway/PLC > make sure they are all powered up.

Step 2:

Open VSLogix > Press the connect button located at the menu bar

This will find the IntelLECAT board in the network. Select it and press connect in the Connection Settings window.



Step 3:

When the IntelLECAT Gateway/PLC is connected, go to Device List in Project Explorer > Select the GX20 devices in their respective ID.

ID is the number you can see on the devices rotary switch. If the ID is 21, select GX20 next to 21 in EtherCAT Profile under Device Editor. You can add description of the device if necessary.

21	GX20	▼	
22	GX20	▼	Second GX20
23	GX20	▼	
24	GX20	▼	
25	GX20	▼	
26	GX20	▼	

Now, your IntelLECAT will be able to communicate with the GX20 MDR Control Devices if the ID is correct. You will be able to see the status of all the GX20 devices under PLC Status View section in EtherCAT tab.

### Taking advantage of Conveyor Types templates:

Here is a step-by-step example of designing a simple ladder program by creating a conveyor type and use it as a template.

Step 1:

Right click on Conveyor Types node>create new>give a suitable name to the Conveyor Type

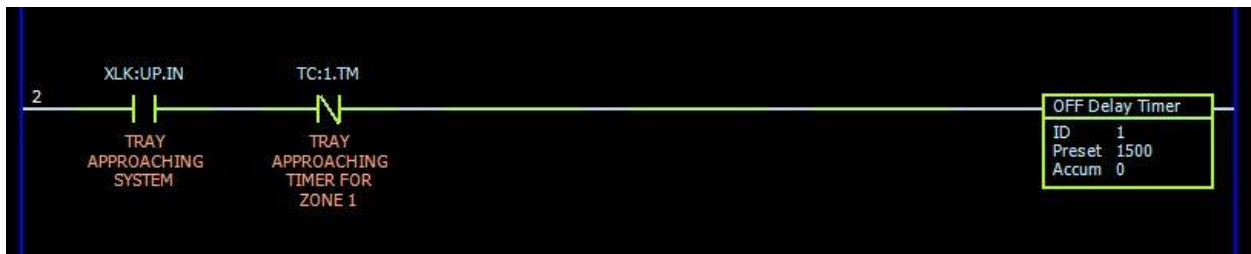
This will open a ladder editor where you can design your ladder program.

Step 2:

If you want to use your conveyor type as a template, it has to be able to communicate with the previous and/or next conveyor section. So, make sure your program has XLK file types.

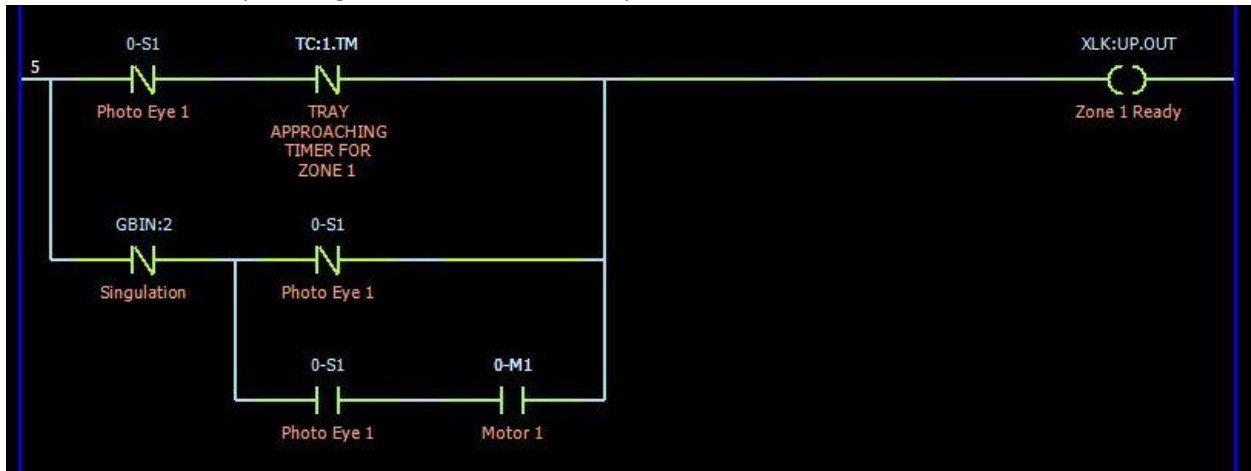
#### XLK:UP.IN

This file type is the state of box available from the previous conveyor section and ready to pull in into the current conveyor section.



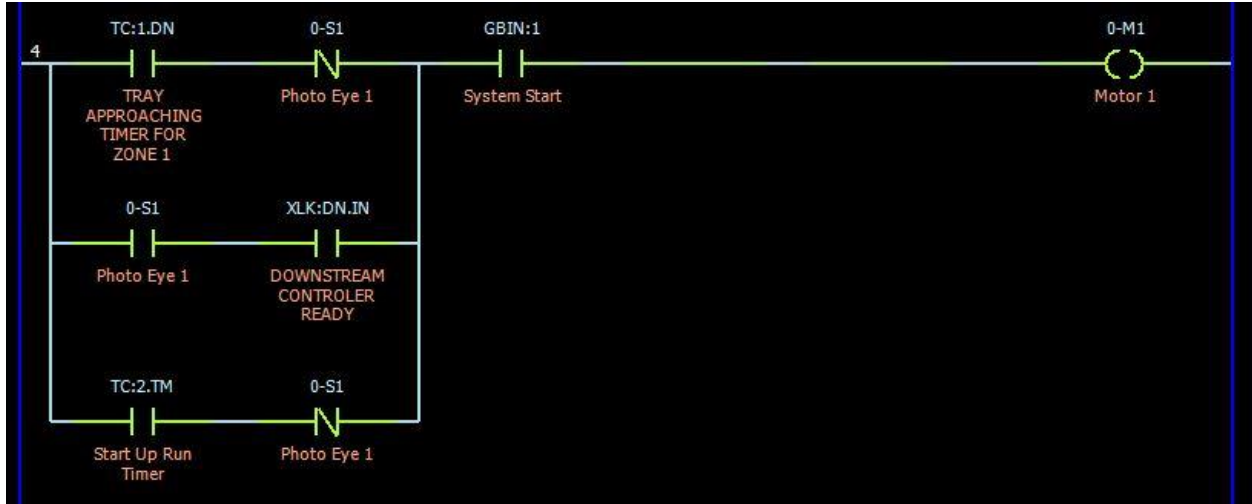
#### XLK:UP.OUT

This file type sends the ready status of current conveyor section to the previous conveyor section. When both XLK:UP.IN which is box available to bring in to the current conveyor section and XLK:UP.OUT which is ready to accept box from the previous section are active, package should move from previous section to current section by turning on the zone motor output.



### XLK:DN.IN

This file type is the state of next conveyor section. If there is a box in current zone and XLK:DN.IN is in ready state, the motor of current zone should turn on to deliver the package from current zone to next zone.

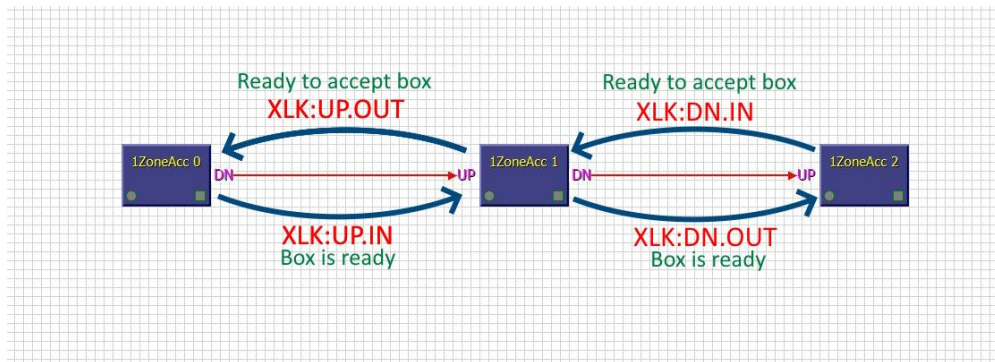


### XLK:DN.OUT

This file type sends current conveyor section status to next conveyor section. If there is a box in the current zone, XLK:DN.OUT status becomes active which means there is a box ready to deliver to the next section.

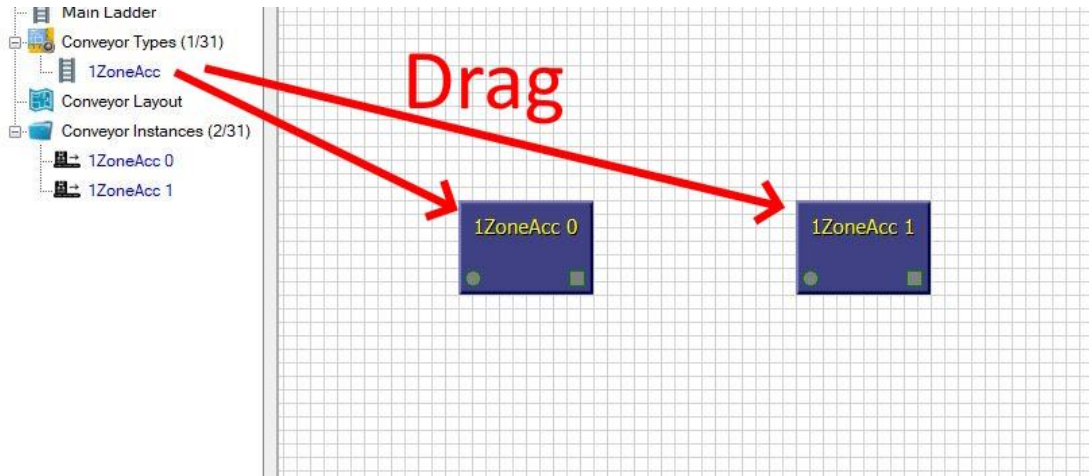


When both XLK:DN.IN and XLK:DN.OUT are in sync, the package should move from current section to the next section.



Step 3:

When the ladder programs for different conveyor types are ready, go to the Conveyor Layout page. Drag as many conveyor sections you need from Conveyor Types and name the sections accordingly.

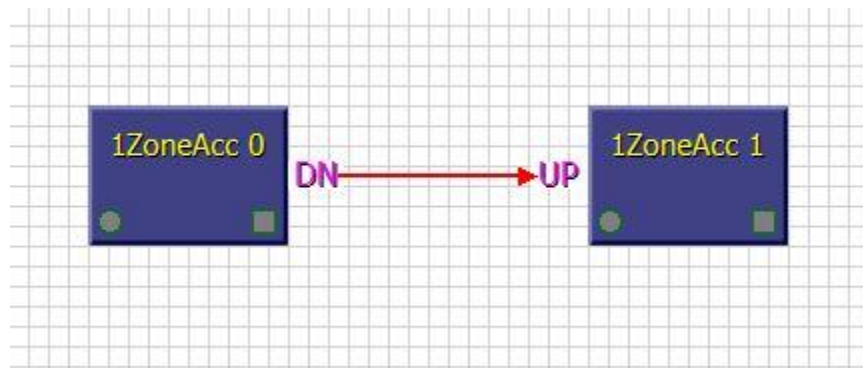


Step 4:

Right click on each Conveyor Sections > Edit Device Map > Select which GX20 devices will be used in that Conveyor Section.

Step 5:

If the Conveyor Flow is from left to right, select the left conveyor section > Press "D" in keyboard, then select the right conveyor section > Press "U" in keyboard. This step completes the interlock between two sections and shows the conveyor flow using an arrow.



Thus, you can add as many conveyor section you need for your entire conveyor plant and connect them using interlock so that they communicate with each other and keep the conveyor flow active.