



VSLogix

User Guide

Document Revision 1.27

(Updated April 9th, 2025)

© 2025 Vital Systems Inc

Buford, GA USA

For more information, please visit the product web page:

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

Contents

License Agreement for VSLogix Software	4
I. Introduction	5
II. Basic Concepts	6
III. User Interface.....	8
Menu Bar	9
Project Explorer Window	10
Ladder Library Window.....	11
Ladder Editor.....	12
Example Ladder.....	15
Toolbox Window	16
Debug Window	17
Output Window	18
Conveyor Layout Window.....	18
Project Settings Window.....	21
Init Params Editor	22
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	34
Register Files	34
Float Files	35
Timer Files	35
Interlock Files	37
Device File Types.....	39
Inputs	40
Outputs	41
Extended Inputs	42

Extended Outputs	43
Global File Types	44
Control Words.....	44
Control Bits.....	45
Global Binary Files.....	45
Global Register Files.....	46
VI. Ladder Commands	46
Input Commands.....	47
Normally Open Command	47
Normally Closed Command	48
Compare Command	48
Output Commands.....	50
Output Coil Command	50
Timer/Counter Command.....	52
Move Command.....	54
Move Range Command.....	54
Math Command	55
Reset Command.....	56
Send Message Command.....	57
VII. Protocols	58
Ethernet/IP.....	59
VIII. Example	60

License Agreement for VSLogix Software

PLEASE READ THIS AGREEMENT CAREFULLY BEFORE USING THE SOFTWARE.

By using the VSLogix software and any associated hardware, you agree to the terms outlined in this License Agreement. Your use of the software constitutes acceptance of these terms.

1. Intellectual Property Rights

VSLogix software and accompanying tools are protected by copyright laws and international intellectual property treaties. Unauthorized reproduction, distribution, or modification of this software is strictly prohibited and may result in civil and criminal penalties. Vital Systems reserves the right to pursue legal action to the fullest extent permitted by law.

2. Acknowledgement of Risk and Responsibilities

You acknowledge that all mechanical and electrical systems inherently involve hazards. By using the VSLogix software and associated hardware in conjunction with any machine or device, you agree to the following:

- You, as the user, accept full responsibility for its operation and any associated risks.
- You agree that neither Vital Systems nor its affiliates, distributors, or partners will be held liable for any damages, injuries, losses, or other misfortunes resulting from the use or misuse of the VSLogix software or any associated hardware.
- You acknowledge that while every effort is made to ensure the software operates correctly and reliably, no software is completely free of bugs or errors. You accept that any errors, faults, or unintended behavior of the software—whether due to system bugs, hardware malfunctions, or operator errors—are your responsibility.

3. Waiver of Liability

By agreeing to this License Agreement, you explicitly waive any claims against Vital Systems and its affiliates for:

- Any direct or indirect damages, including but not limited to material loss, injury, or secondary damages, caused by the use of the software.
- Any errors or operational faults that arise from the use of VSLogix or its associated hardware.

You agree to indemnify and hold harmless Vital Systems and its affiliates from any and all claims, damages, or liabilities arising from the use or inability to use the software.

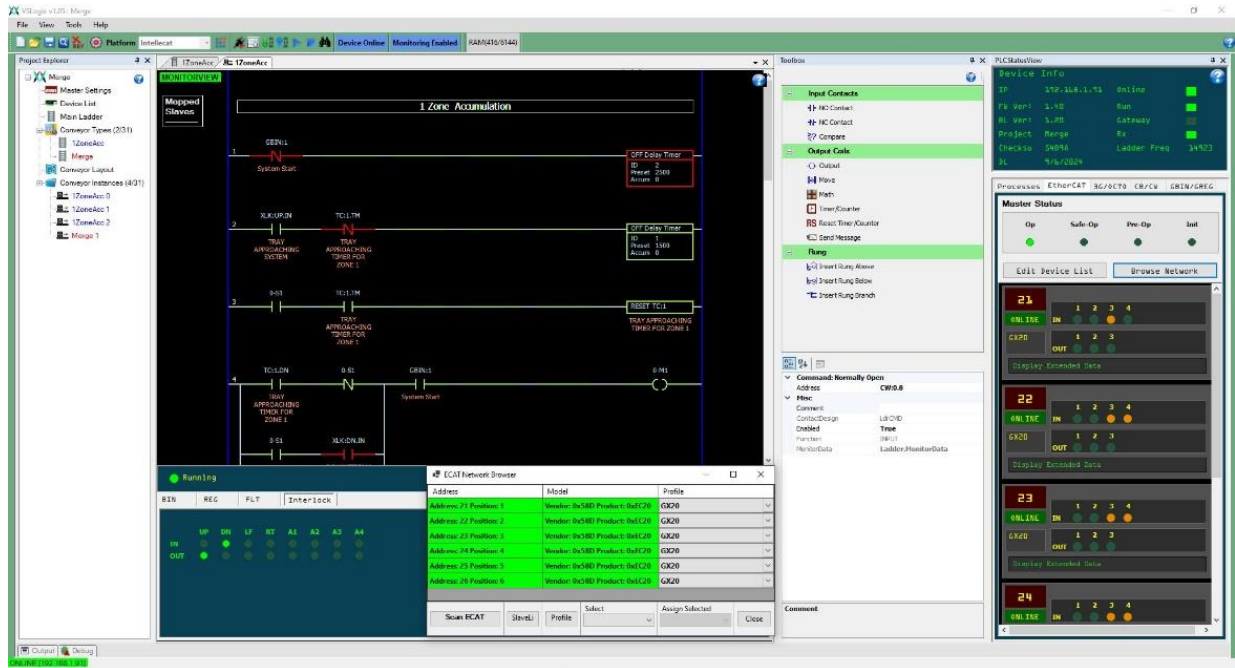
4. User Responsibilities

- You are solely responsible for ensuring the safe operation of the software and associated hardware, both by yourself and by any other individuals who may use it.
- It is your responsibility to inform any operators or personnel working with or near machines controlled by VSLogix of any limitations or potential risks associated with the software.
- You agree to comply with all applicable safety procedures and regulations in your jurisdiction, including but not limited to standard operating procedures, industry safety guidelines, and any local laws related to the safe operation of machinery and software.

5. Acceptance of Terms

By using VSLogix, you acknowledge that you have read, understood, and agree to the terms of this License Agreement. You fully accept all responsibility for the operation of the software and associated hardware and agree to ensure that all users of the system are aware of these terms.

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 serves as a direct interface to the IntellecAT gateway and its connected network. It allows for the writing, configuration, and real-time monitoring of ladder programs, as well as the current status and I/O of all connected GX20 control cards.

VSLogix Features:

- Create robust, powerful, and reusable ladder-logic programs and download them to devices.
- Upload and modify existing ladder-logic programs directly from the IntellecAT gateway.
- Scan and Configure GX20 network through the IntellecAT gateway.
- Monitor program execution and debug programs by changing values during runtime.
- Drag-and-drop Graphical User Interface for simple yet extensive ladder-logic programming.

IntellecAT Features:

- In Gateway mode provides Ethernet access to a network of up to 128 GX20 control cards.
- In PLC mode, can run up to 32 ladder programs controlling up to 64 GX20 control cards or 10 Smart3G cards.

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 control card 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).
Smart3G	Useful where you need screw terminals for 24V digital I/O (8 in, 8 out)
Octo24	IP67 Hardware for liquid-proof I/O control 24V digital I/O (8 in, 8 out)

II. Basic Concepts

The IntelleCAT gateway can control up to 128 GX20 control cards and up to 10 Smart3G/OCTO cards. How these devices will be controlled depends on which of two modes the IntelleCAT is in.

Gateway Mode

In Gateway Mode, devices will be controlled directly by a connected Master PLC. No ladder programs will run on the IntelleCAT gateway itself. The IntelleCAT will simply act as a gateway to the GX20 network for a connected Ethernet/IP Master. Depending on the model of IntelleCAT, you may still need to configure the IntelleCAT's device list and download the VSLogix 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 network are controlled by Ladder programs running inside the IntellecAT Gateway. These are created using the [Ladder Editor](#). The IntellecAT gateway 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)

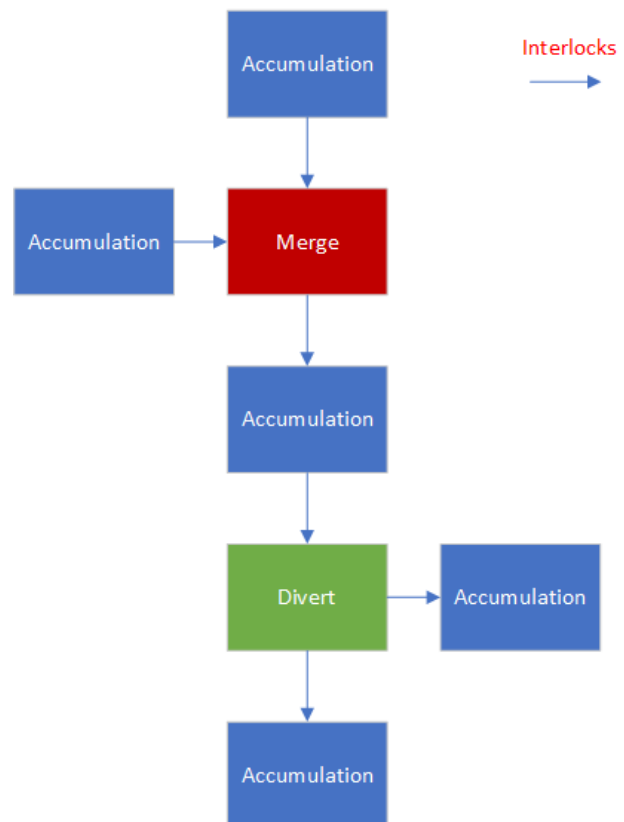
Conveyor Types are Ladder Programs that can be instantiated multiple times as **Conveyor Sections**. Instead of referring directly to device addresses, Conveyor Types use placeholders. These placeholders are later mapped to actual addresses when a Conveyor Section is created from that type. This allows 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.

Types



Sections



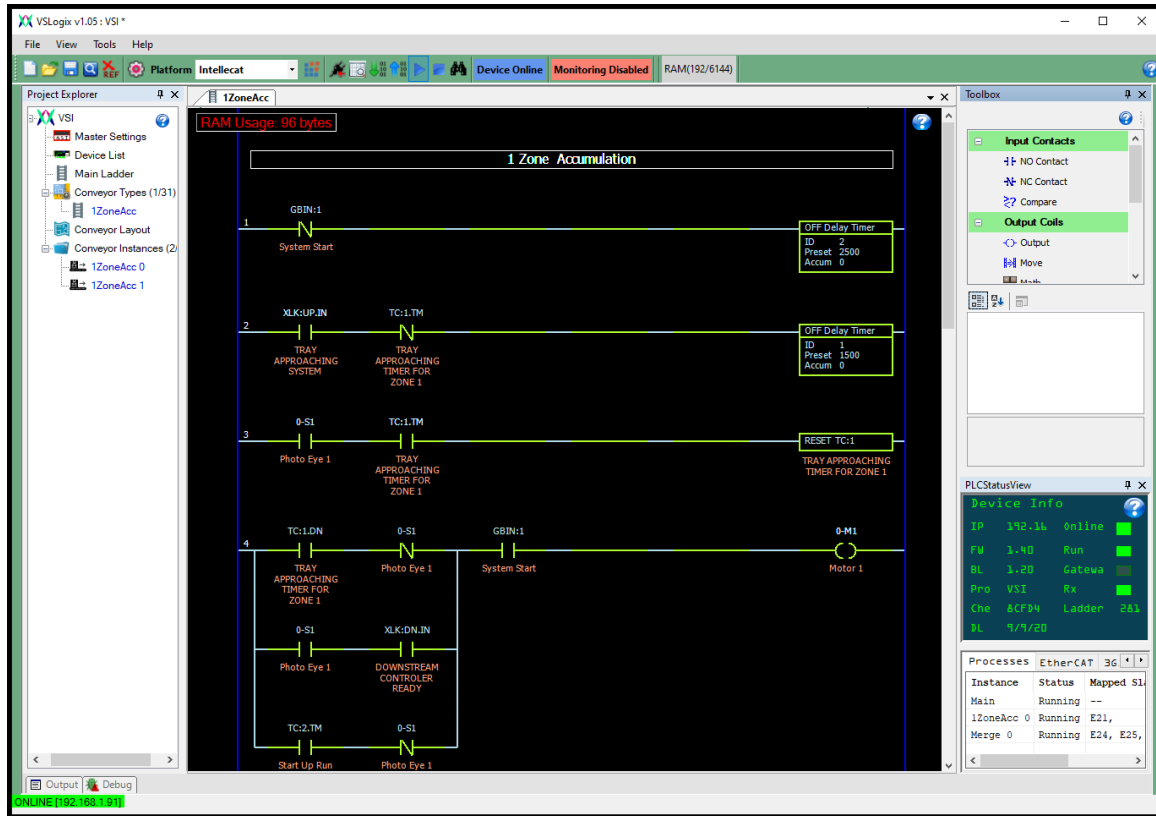
Conveyor Sections

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 control cards. However, you cannot exceed the total max number of devices controllable by the IntellecAT (64 GX20, 10 Smart3G/OCTO).

Conveyor Layout

In the Conveyor Layout window, you can drag and drop conveyor types to create conveyor sections. You can then assign GX20 control cards to those sections. These sections can also be connected to one another using interlocks, establishing the overall flow of the conveyor system.

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. [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. Cross Reference Window | 14. Stop All Ladder Programs on Device |
| 6. Master Settings Window | 15. Enable Monitoring |
| 7. Select Platform | 16. Device Connection State |
| 8. Compile Ladders | 17. Device Monitoring State |
| 9. Connect To Device | 18. Memory Consumption |

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

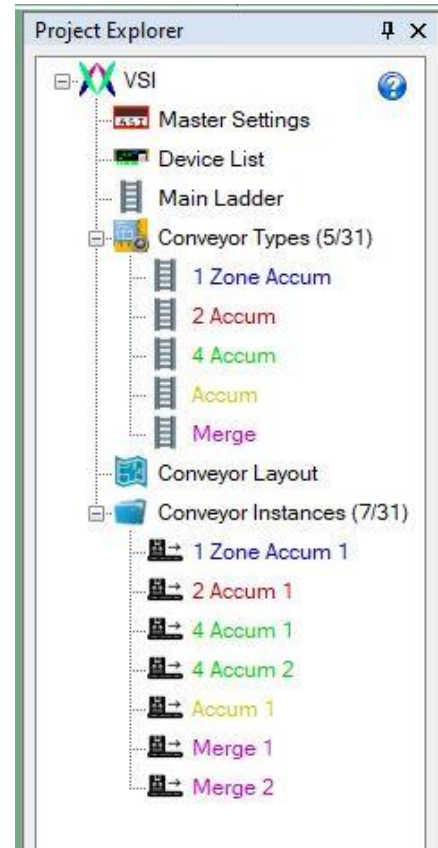
- 1-3 allow creating/opening/saving projects.
- 4 and 5 are used to search through the ladder programs in the project.
- 6 opens the [Master Settings Window](#) for editing global settings.
- 7 is the Platform Selection. The Platform is used to select the intended device the project will be running on.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

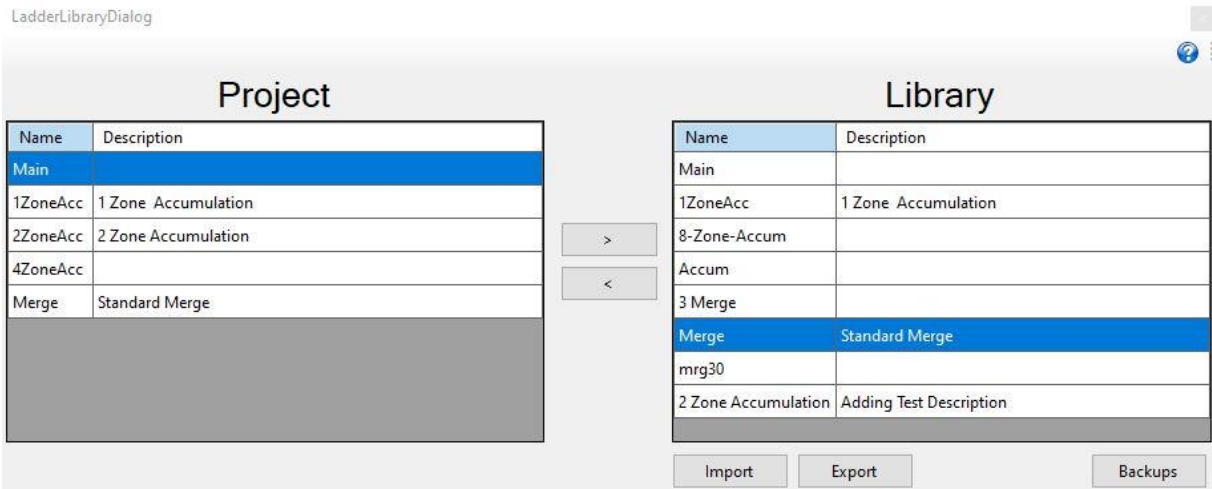
The Project Explorer Window is the basic navigation and management window for the application. It is open by default, but can be re-opened from the View Menu if closed.

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 an IntellecAT is [connected to VSLogix](#).
- Master Settings Node – Opens [Master Settings window](#), allowing editing of global project settings.
- Conveyor Layout Node – Double clicking or 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.



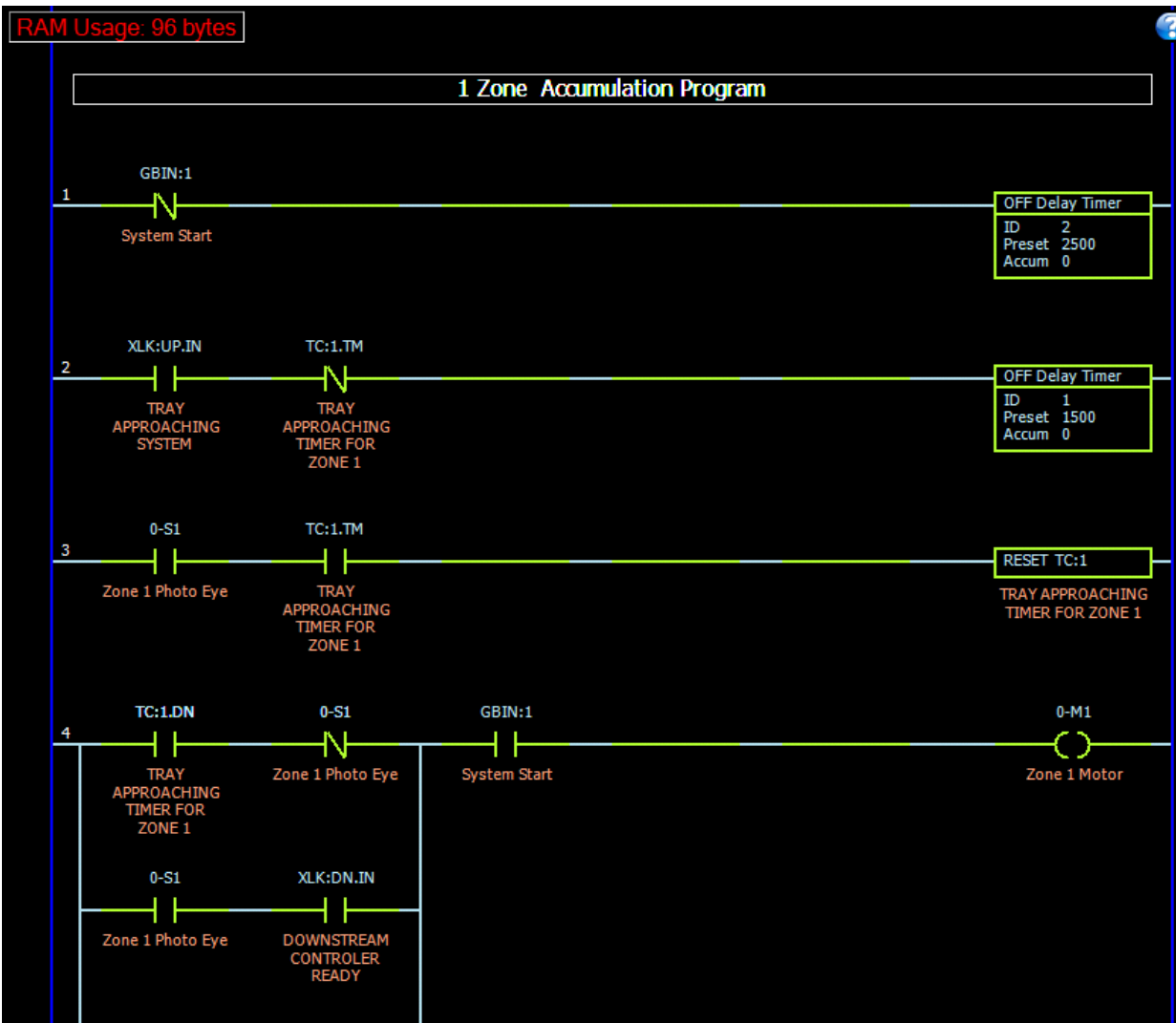
Ladder Library Window



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.

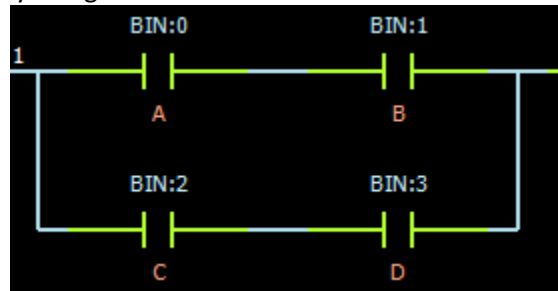
Ladder Editor



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:

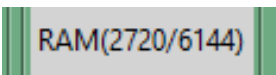
1. Main Ladder
2. [Conveyor Sections](#), ordered first by Ladder Program, then by Name

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.



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. If the value starts to approach the maximum, you may want to consider reducing the number of unique addresses used in your ladders.

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

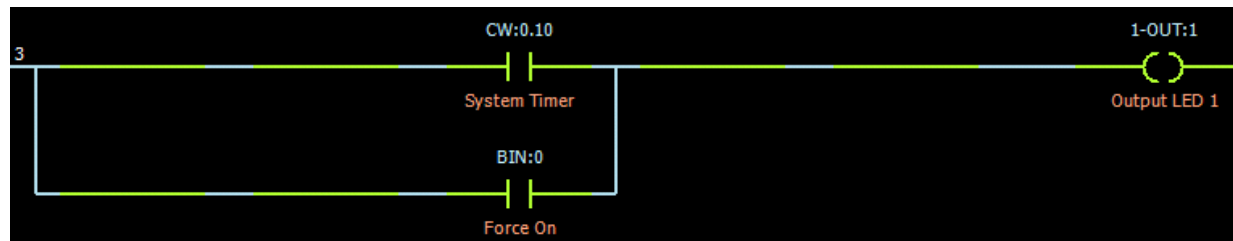
Using the above information of how the Ladder Editor works, we will create an example Ladder to demonstrate the creation process. Let's start by creating a simple flashing LED:



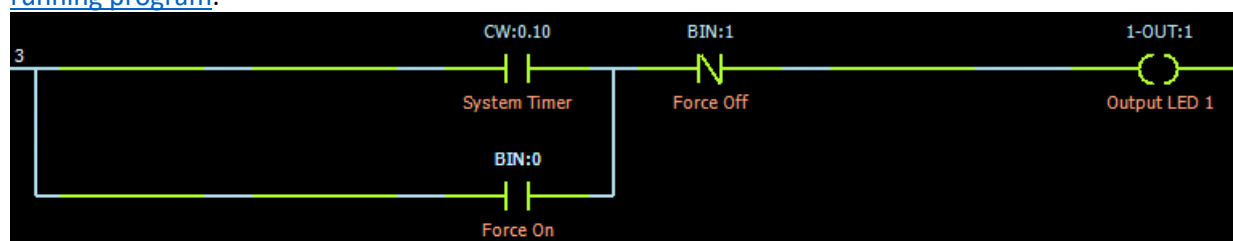
On the left we have a [Normally Open](#) command with the address CW:0.10, and on the right we have an [Output](#) command with the address 1-OUT:1. A more detailed explanation of these addresses can be found in the [File Address](#) section, but simply put:

- The Address CW:0.10 refers to the 10th bit of Control Word 0, the System Timer. Since the system timer counts up in milliseconds, the 10th bit will toggle every 2¹⁰ (aka 1024) milliseconds, or roughly once a second. The [Timer command](#) exists for more complex timing needs.
- The Address 1-OUT:1 refers to digital output 1 of Device 1. 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.

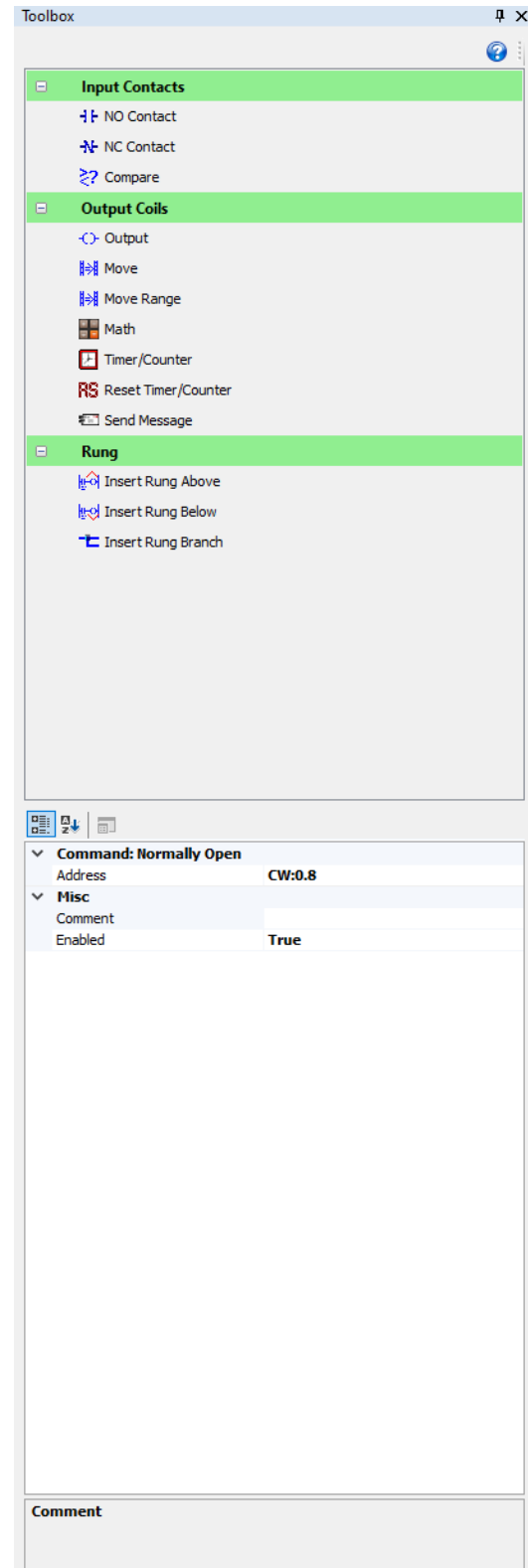
Toolbox Window

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

Cross Reference

Ladder Program: [All Sources]

File Type ALL


Refresh

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 Approaching 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.

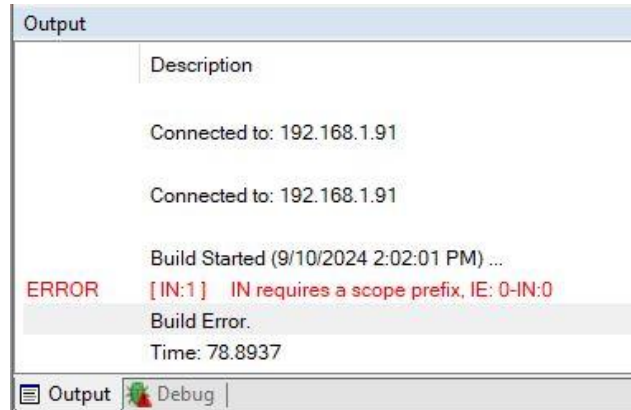
Debug Window

Debug					
	Description	File	Rung	Position	Level
	[IN:1] IN requires a scope prefix, IE: 0-IN:0	4ZoneAcc	3	1	1

Output
Debug

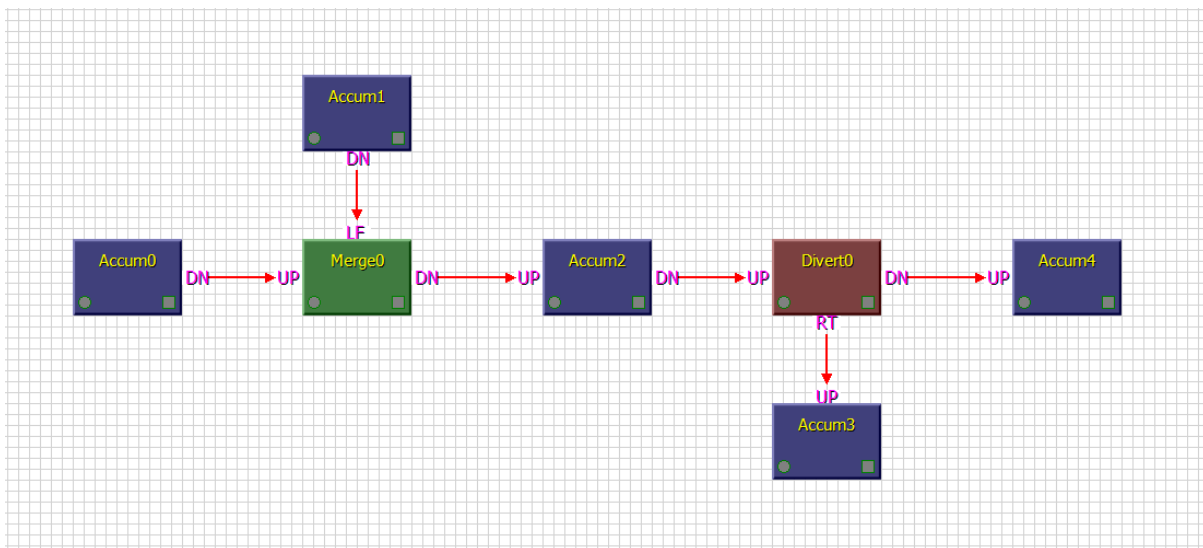
The Debug Window displays compilation errors for the ladder program (if there are any) after the compilation process. Error entries can be clicked to jump to the source of the error. The Debug Window can be accessed by clicking the Debug Window Icon under the menu bar, otherwise it pops-up when errors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

Output Window

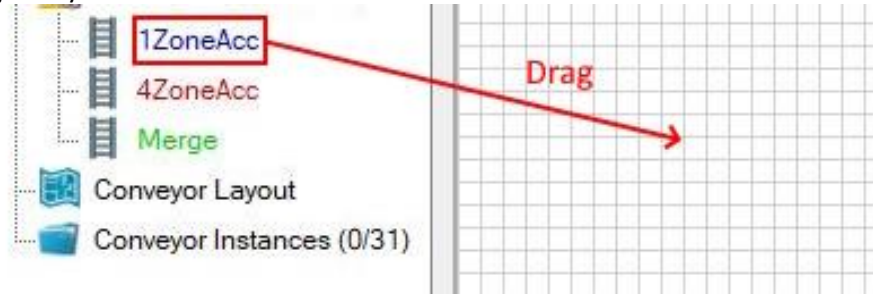


The Output Window provides detailed and technical feedback such as application log messages, compile messages, and errors. The Output Window can be accessed by clicking the Output Window Icon under the menu bar, otherwise it pops-up automatically when necessary.

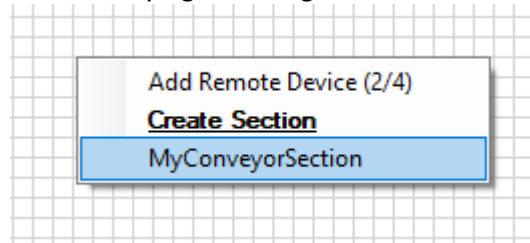
Conveyor Layout Window



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.



Alternatively, you can create a section by right clicking into a blank area on the grid:

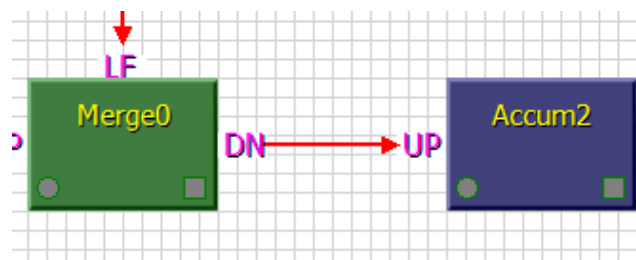


After creating more than one section, you can create interlocks between them. These allow the Ladder Sections to pass data directly to one another. To do this, first select one section, then press a key to choose a port. Then select a different section you want to connect it to, and press a key to choose its port.

The available port choices are:

U	Upstream
D	Downstream
L	Left
R	Right
0-9	Auxiliary 0-9

Note the direction of the arrows. Arrows will point away from Downstream ports and towards Upstream ports to signify the flow of packages.

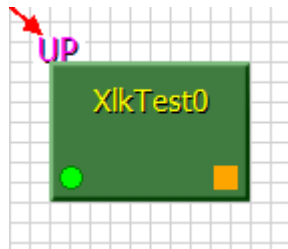


Note that interlocks simply allow the transfer of data from one section to another. It is up to the designer of the ladder program to choose what data to send, and how to process it. Within the Ladder Program, this Interlock data is accessed using the [XLK Filetype](#).

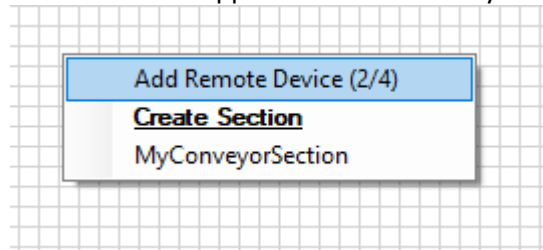


You can also right click the Conveyor Sections to [edit their Device Maps](#), to edit their [Initial Parameters](#), or to delete them. You can also select monitor, or double click the section to open its monitoring window. Note that Monitoring the section's execution in real-time will require an active connection to the IntellecAT gateway.

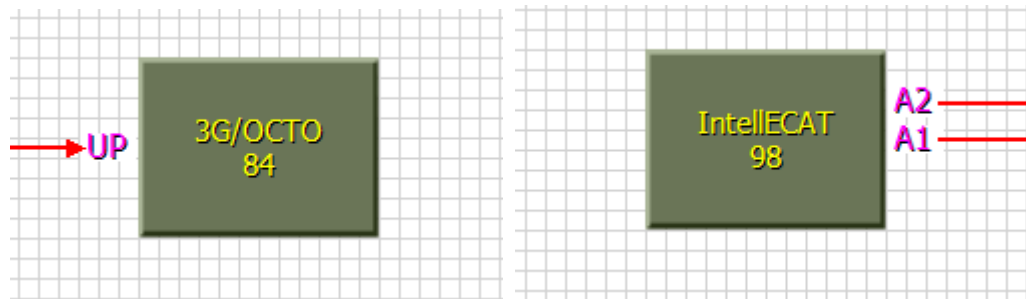
When you monitor a conveyor section, Monitoring Mode will be activated. This will be indicated at the top of the screen. You can also manually enable or disable Monitoring Mode with the binoculars button. While Monitoring Mode is enabled, the Conveyor Layout window will display very basic information 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. Note that you cannot edit anything in the Conveyor Layout window while Monitoring mode is enabled.



In addition to adding Conveyor Instances, you can also create Remote Devices in the Conveyor layout Window. These appear similar to Conveyor Sections, with some key differences:



There are two types of Remote Devices that can be created, 3G and IntellecAT.



3G Remote Device

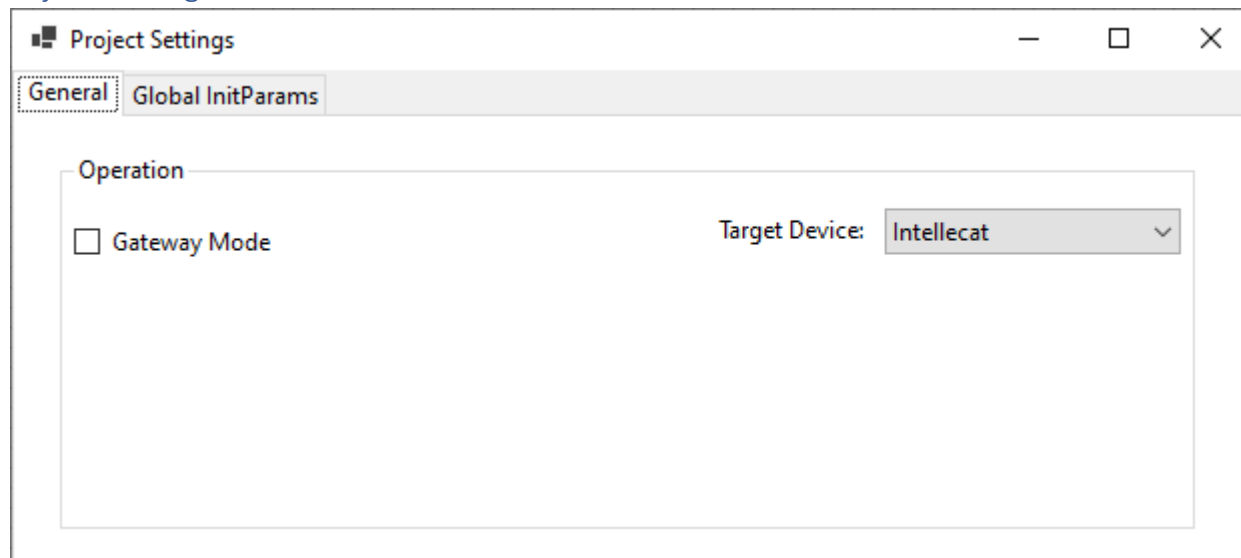
By adding a 3G remote device, the IntellecAT card can communicate with an existing Smart3G/OCTO network through interlocks as if it were a 3G/OCTO card itself. This allows Smart3G and IntellecAT networks to coexist and interop together.

IntellecAT Remote Device

By adding an IntellecAT remote device, the network of Conveyor Sections in the current IntellecAT Project can be connected seamlessly with those in another IntellecAT. Interlocks are made with the Remote Device both on the sending and receiving side as if the Remote Device were just another conveyor Section.

The primary difference between these two is that 3G Remote Devices only support 1 interlock at a time, and do not support [interlock messaging](#). IntellecAT Remote Devices support up to 14 Interlocks and do support [interlock messaging](#).

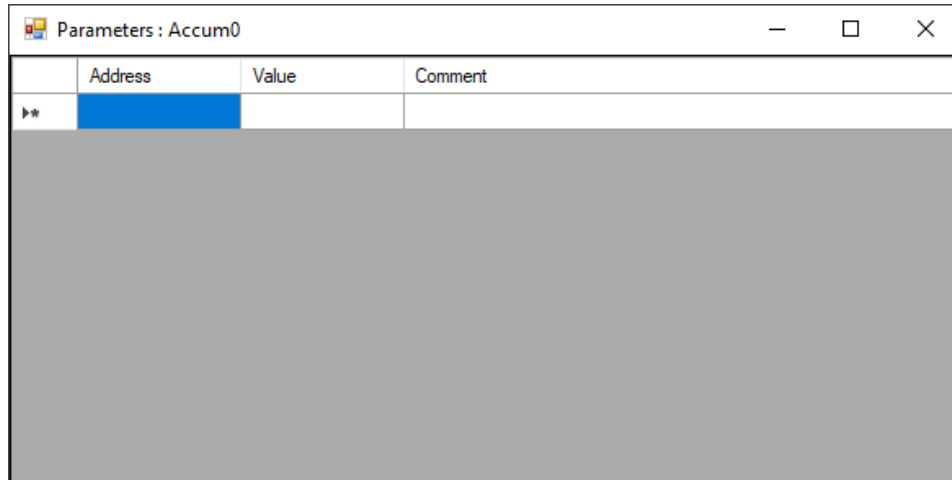
Project Settings Window



The Project Settings window is used to configure global settings for the device. This includes:

- Toggling Gateway Mode Gateway Mode on or off
- Selecting the target hardware the project is intended for
- Setting [Global Init Parameters](#), effectively default values for global files

Init Params Editor



The Initialization Parameters Editor (Init Params for short) allows you to set the values for files on startup of the device. These values will be set whenever a ladder is first started. Simply enter the file 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.

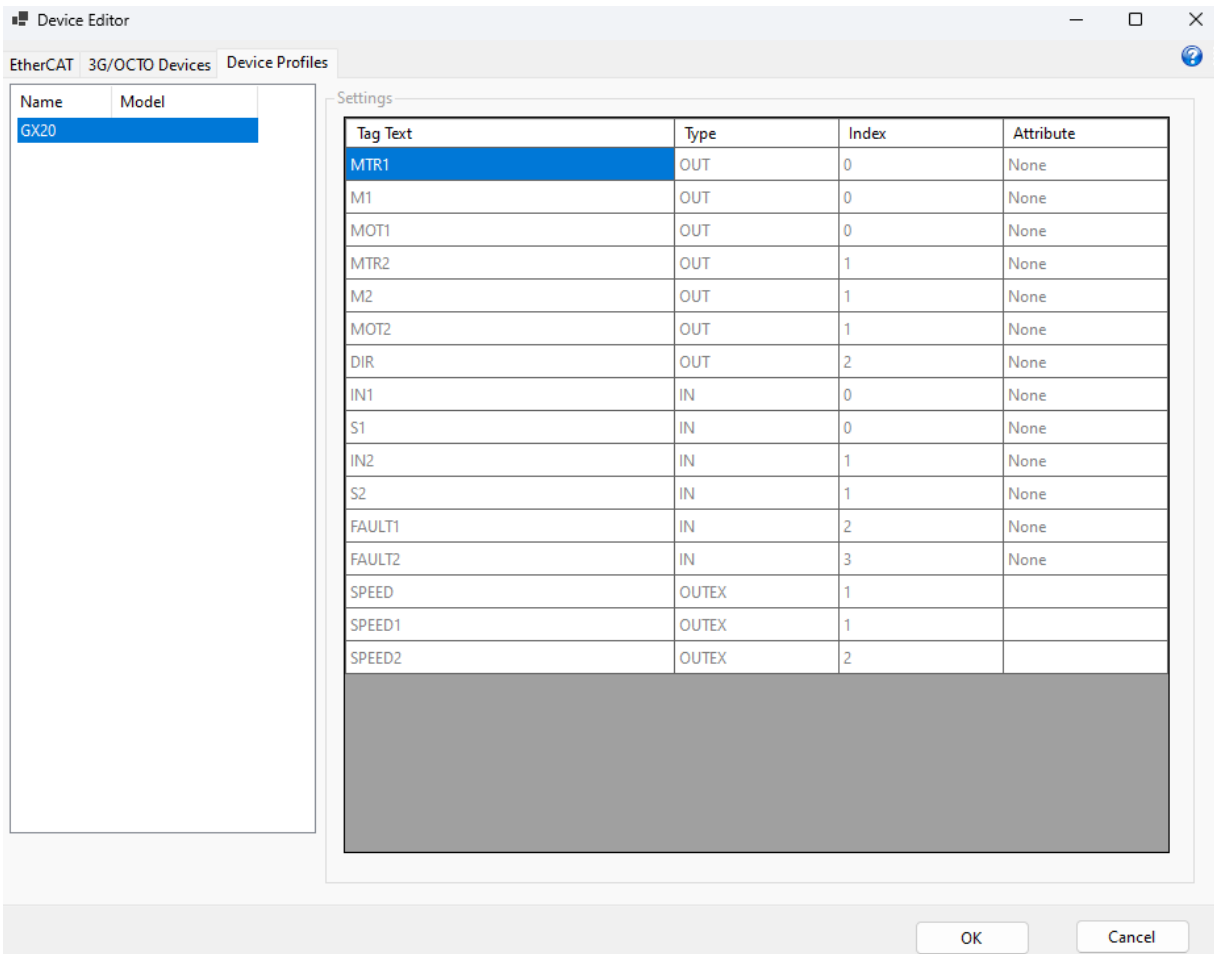
There are two different Init Param editors available:

- One specifically for [Global File Types](#), accessible from the [Project 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

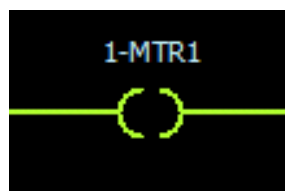
You can open this window from the Tools dropdown on the menu bar, by right clicking a Conveyor Section and clicking "Edit Device Map" or from the GX20 Status Tab. This window is used to configure the devices in your project, and will be used in every project. Refer to the [Device File](#) section to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

Device Profiles Tab



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

When using a Device File, you can use any of the listed tags instead of needing to manually determine the proper Index and Attribute. For example, I can turn on Motor 1 on a GX20 card like so:



EtherCAT Tab

Device Editor

EtherCAT
3G/OCTO Devices
Device Profiles

Device Report

☐ Hide Unmapped Devices

Clear All

	Profile	Description	
1	GX20	Accum 1 Card	Clear
2	GX20	Accum 2 Card	Clear
3	GX20	Merge 1 Card	Clear
4	GX20	Merge 2 Card	Clear
5	GX20	Transfer Card	Clear
6	GX20	Not Used	Clear
7	GX20	Not Used	Clear
8			Clear
9			Clear
10			Clear
11			Clear
12			Clear
13			Clear
14			Clear
15			Clear
16			Clear
17			Clear
18			Clear

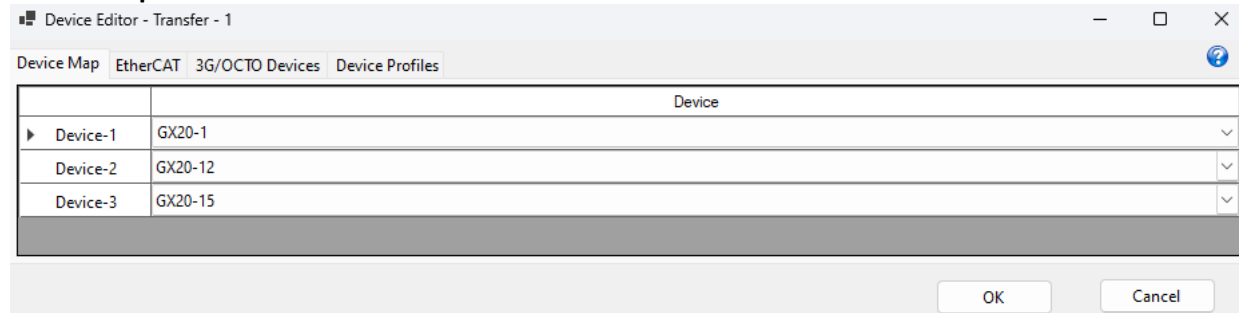
OK

Cancel

The EtherCAT tab allows you to configure which GX20 control cards 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 should match the type of GX20 control card you intend to connect to. On the left of the tab, you can see the device IDs.

For example in the above image we have set ID 1-7 to GX20, so communication will be performed with these cards on the network. If a configured device is not found on the network, it will still be scanned for and communication will begin when it is found.

Device Map Tab

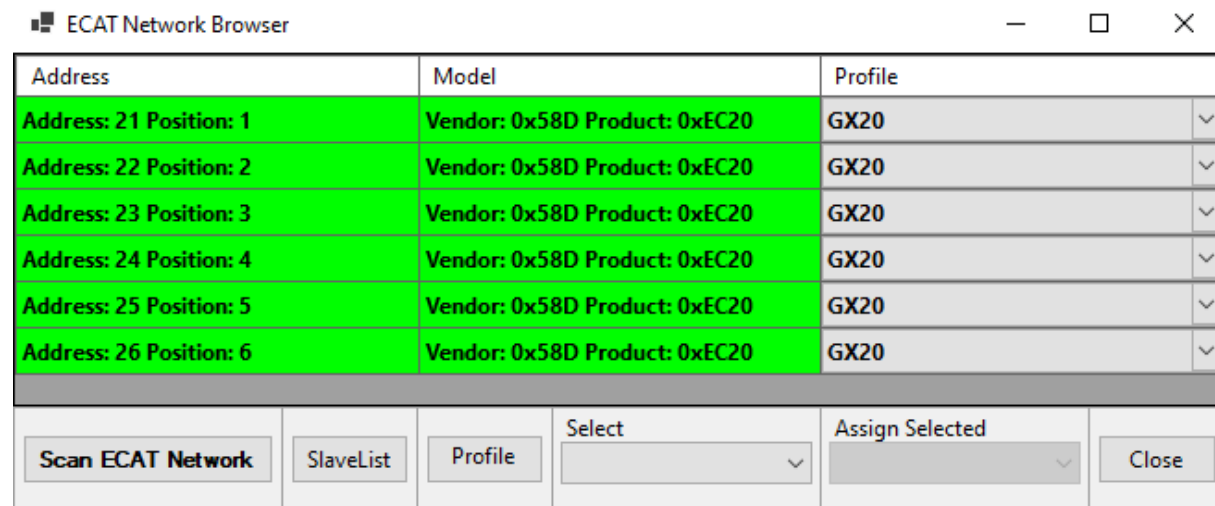


The Device Map tab can be opened by right clicking a conveyor section and selecting “Edit Device Map.” This can be done from either the Project Explorer or the Conveyor Layout window. Within the window [Device Placeholders](#) will appear in the left column. In the right column you can map to these placeholder devices that were configured in either the EtherCAT or 3G/OCTO tabs.

This functionality exists so that Ladder programs can be written once and be reused multiple times in the same network, for different sets of GX20s. For example, I may create an accumulation Conveyor Type that controls the motors of a GX20 card. From this Conveyor Type I could create two Conveyor Sections and map two different GX20s to these sections.

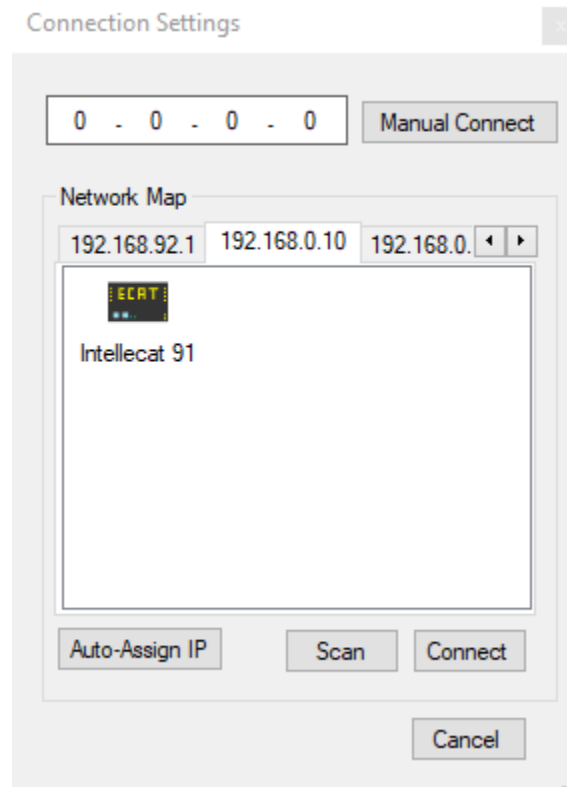
Refer to the [Device File](#) section to see how Devices should be accessed within ladder programs.

EtherCAT Network Browser



The EtherCAT Network Browser can only be accessed after an IntellecAT gateway is connected to VSLogix via the [Connection Dialogue](#). The EtherCAT 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).

Connection Dialogue



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

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. However, we would advise against performing auto-assign on production networks as resetting all device IPs could cause issues on complex network configurations.

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).

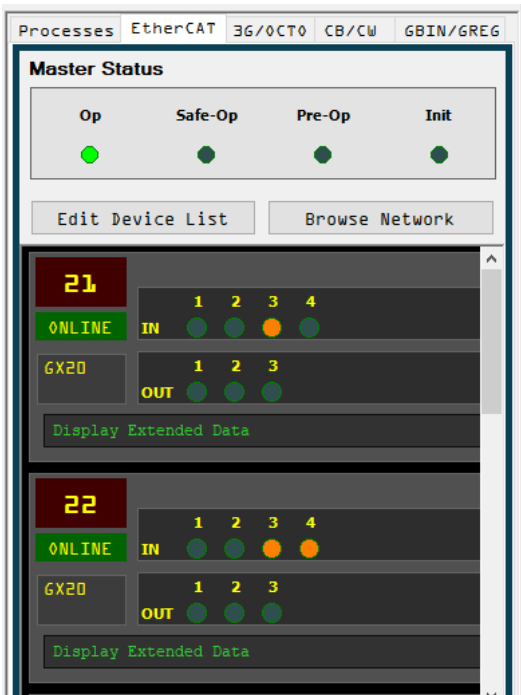
Device Status Window

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:

PLCStatusView			
Device Info			
IP Addr:	192.168.1.91	Online	■
FW Ver:	1.40	Run	■
BL Ver:	1.20	Gateway	■
Project:	VSI	Rx	■
Checksum	F0E32	Ladder Freq	16323
DL Time:	9/10/2024		
Processes			
EtherCAT 3G/0CT0 CB/CW GBIN/GREG			
Instance	Status	Mapped Slaves	Ladder
Main	Running	--	Main / 46454
1ZoneAcc 0	Running	E21,	1ZoneAcc / EF9DC
1ZoneAcc 1	Running	E22,	1ZoneAcc / EF9DC
1ZoneAcc 2	Running	E23,	1ZoneAcc / EF9DC
1ZoneAcc 3	Running	E24,	1ZoneAcc / EF9DC
1ZoneAcc 4	Running	E25,	1ZoneAcc / EF9DC
1ZoneAcc 5	Running	E26,	1ZoneAcc / 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 devices 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. Finally, in the far right column is listed the execution time in microseconds each section is taking during runtime.



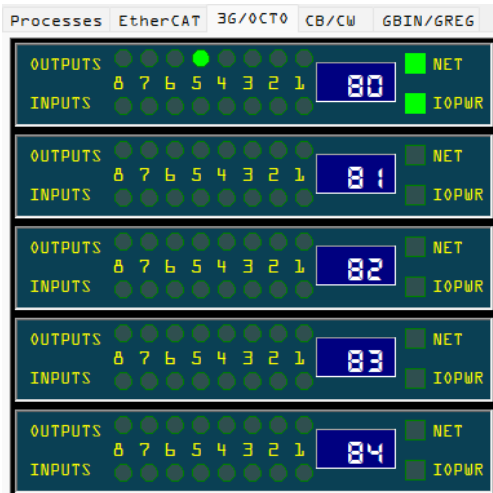
EtherCAT

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:

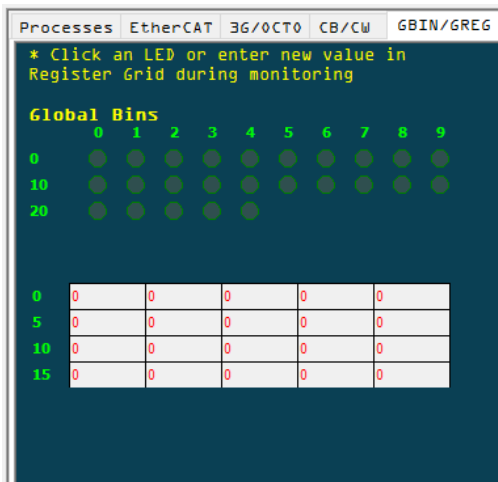
- **OP** – In this state, full process data exchange is possible between the master and its connected devices. Both inputs and outputs are active, and the system is fully functional. This is the state in which normal EtherCAT communication and control occur.
- **Safe-Op** – The device is ready to exchange 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 Smart3G and OCTO cards, which were configured from the [Device Editor](#). 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.

Here devices with ID 80 - 84 are displayed in the status window, which means, these devices have been used in the ladder program. The device with ID80 is connected to the IntelleCAT as we can see the NET and IOPWR lights are on.



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 control card which has ID 22, select profile 22 and assign GX20 control card there. Selecting a matching profile for a device will add it to the IntellecAT'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, type and index 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 in 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 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 .

You can rename any conveyor section by Right Click > rename option. You can quickly go to the ladder editor of any conveyor section by Right Click > Edit Ladder. You can also run, monitor or delete any specific conveyor section by Right Click and selecting Run Ladder, Monitor or Delete 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 control cards 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.

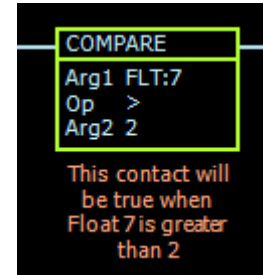
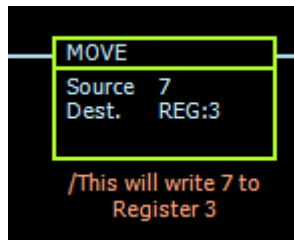
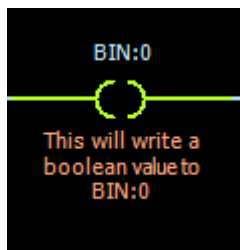
Once online, you will also now be able to open the [EtherCAT network browser](#) from the EtherCAT tab of the Status window. This will allow you to view all GX20 control cards on the EtherCAT network, whether they have been configured in the project or not.

V. Files Addresses

IntellecAT 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:

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, Device, or Global.
Attributes	<p>The data attributes that are present for the given filetype. The data type of the attribute will be enclosed in parentheses (Boolean, Integer, Float).</p> <p>The individual bits of all filetypes can be accessed by supplying a number for the attribute. For example, an attribute of 3 will access bit 3 of that file.</p> <p>Some filetypes also allow a syntax to read/write multiple bits at once. An attribute with the format 'b0000' will specify a bitmask in binary with the bits you wish to read/write. Hexadecimal can also be used with the format 'h00'. FileTypes where bitmask reading/writing are allowed will have this specified in their attribute section.</p>
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.

Binary Files



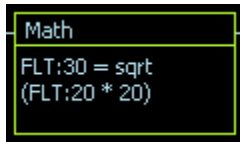
Description	A bit value that can be 0 or 1.
Keyword	BIN
Scope	Local
Data Type	Boolean
Usable Attributes	<ul style="list-style-type: none"> None Full – Directly access the BIN's containing integer (becomes Integer type)
Format	BIN:<file index>.<optional attribute>
Usage	Read, Write

Register Files



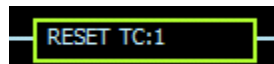
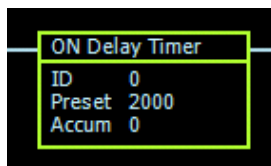
Description	A 32-bit signed integer. Referred to plainly as “Register” in VSLogix. <div style="border: 1px solid black; background-color: #FFD700; padding: 5px; margin-top: 10px;"> NOTE: Although numerical values with decimal point precision can be assigned to these files, the digits following the decimal point are dropped. For numerical values with a decimal point, consider using the float registers (FLT). </div>
Keyword	REG
Scope	Local
Data Type	32-bit Signed Integer
Usable Attributes	<ul style="list-style-type: none"> 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	<ul style="list-style-type: none"> None
Format	FLT:<file index>
Usage	Read, Write

Timer Files

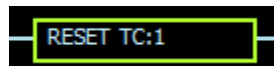


Description	<p>Timers that can keep time in a resolution of milliseconds. The timer keeps time until the user-defined preset value is reached.</p> <p>Unlike other FileTypes, each Timer/Counter File corresponds to an associated Timer/Counter command in the Ladder program. You cannot use a Timer/Counter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

Usable Attributes	<ul style="list-style-type: none"> • 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)

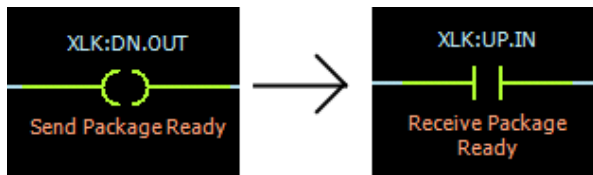
Counter Files

UP Counter
ID 1
Preset 300
Accum 0

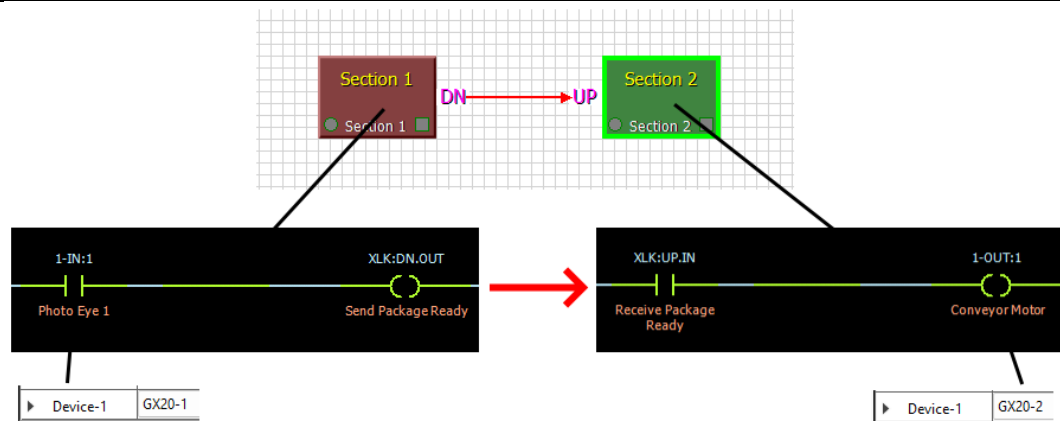


Description	<p>Counters that increment their accumulated value by 1 every time the rung state transitions to true. Directly usable only with the Reset and Counter Command.</p> <p>Unlike other FileTypes, each Timer/Counter File corresponds to an associated Timer/Counter command in the Ladder program. 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>
Keyword	TC
Scope	Local
Data Type	32-Bit Signed Integer / Boolean
Usable Attributes	<ul style="list-style-type: none"> • 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)

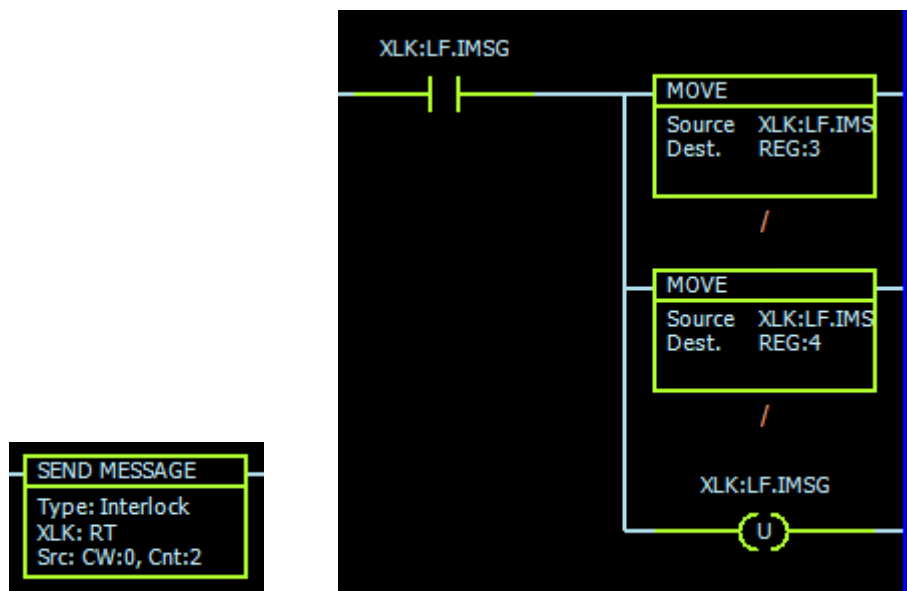
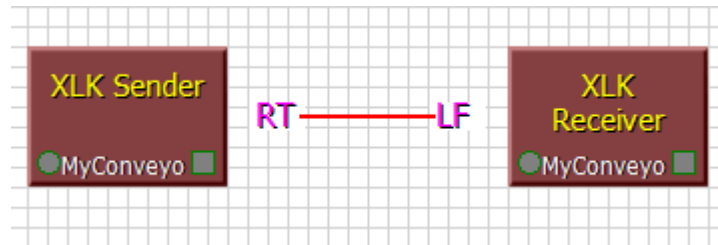
Interlock Files



Description	Used to send or receive data 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, A0, A1, A2, A3, A4, A5, A6, A7, A8, or A9. Use of one of the Attributes IN, OUT, IMMSG, IMMSG1 or IMMSG2 is required.
Keyword	XLK
Scope	Local
Data Type	Boolean, Integer
Usable Attributes	<ul style="list-style-type: none"> • IN – Read bit coming <i>from</i> the connected Section (bool value) • OUT – Read/Write bit going <i>to</i> the connected Section (bool value) • IMMSG – This bit is set to true when a message is received from another Conveyor Section using the SendMessage command (bool value) • IMMSG1 – The first integer file received from a SendMessage Command (int value) • IMMSG2 – The second Integer file received from a SendMessage Command (int value)
Format	XLK:<interlock channel>.<attribute>
Usage	Read, Write (OUT, IMMSG 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.



In this example, the XLK Sender Section is using the SendMessage command to send a message to the XLK Receiver Section.



The XLK Receiver is checking the IMSG bit to look for a received message. Once it finds it, it uses Move commands to copy over the message data from the IMSG1 and IMSG2 attributes. Then it unlatches the IMSG bit to indicate the message has been processed.

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:

E1-M1

This would access Motor 1 of EtherCAT Device 1.

There are two ways to address a device, Absolute and Placeholder:

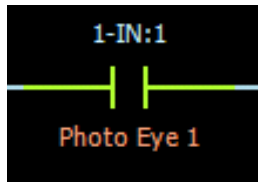
Absolute E#, 3G#, OCTO#, V#	<p>The scope of each device File must start with a prefix:</p> <ul style="list-style-type: none"> • E – EtherCAT Device • 3G, OCTO, V – 3G or OCTO Device <p>Using the prefix followed by a number will refer directly to that device address. E10 will refer to the EtherCAT Device (GX20) with address 10. If you use V10/3G10/OCTO10, it will refer to a Vital Systems device (Smart3G/OCTO) with that address.</p>
Placeholder E(#), 3G(#), OCTO(#), V(#)	<p>If you instead wrap the number in parentheses, the address will be a Placeholder Address. As the name suggests, placeholder addresses do not directly refer to the device to be controlled, but will later be mapped to the desired device using the Device Mapping window.</p> <p>This is usually preferred over Absolute addresses as it allows you to reuse a conveyor type multiple times, mapped to a different set of Devices each time. Even if no reuse is desired, it still allows you to easily change the mapped device without needing to edit the ladder itself.</p> <p>For example, I might use the File Address E(1)-IN:1 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 1 to GX20 with address 5, while in the other I could map it to a GX20 with address 7.</p>

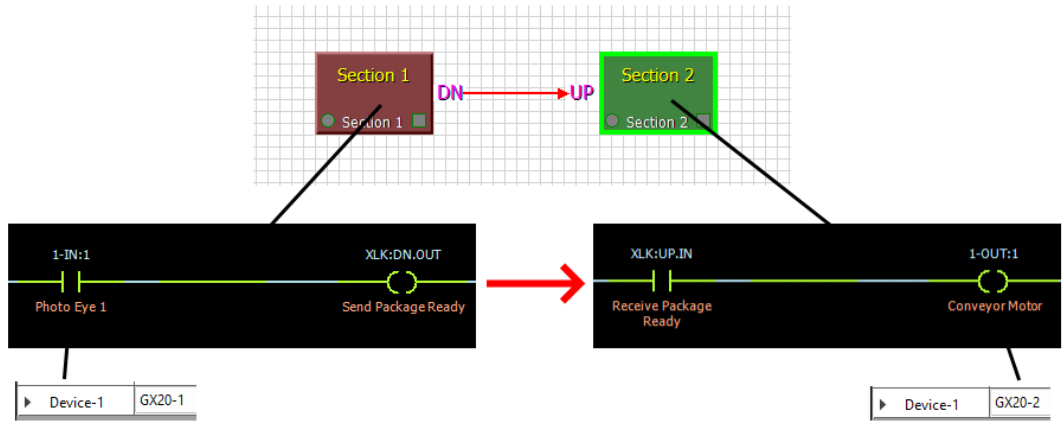
Refer to the Profile tab of the [Device Editor](#) for an explanation of Short Tags such as M1, S1, MOT1, etc. that will help you write more readable ladder programs.

Device Files:

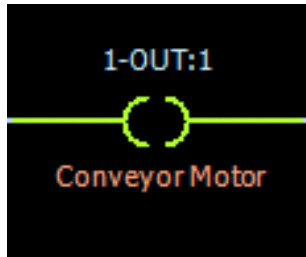
- [IN](#) – Digital Input on specified Device
- [OUT](#) – Digital Output on specified Device
- [INEX](#) – Extended Input
- [OUTEX](#) – Extended Output

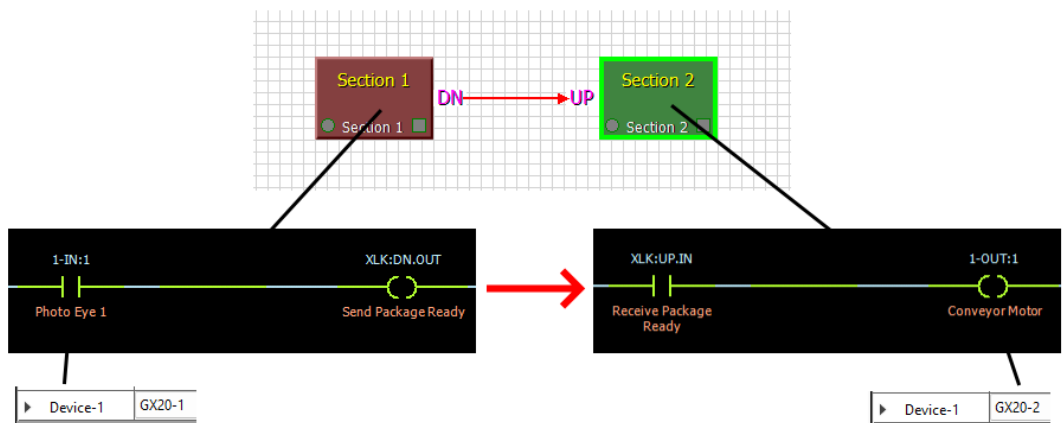
Inputs



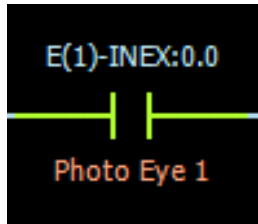
Description	Input state of the selected device (similar to BIN values in usage).
Keyword	IN
Scope	Device
Data Type	Boolean
Usable Attributes	<ul style="list-style-type: none"> None Full - Directly access the File's containing Byte (becomes Byte type)
Format	<device>-IN:<fileIndex>
Usage	Read
Example	<p>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 Device Map window for Section1, Device-1 has been mapped to GX20-1. Therefore, 1-IN:1 will correspond to Input 1 of GX20 control card 1.</p> 

Outputs



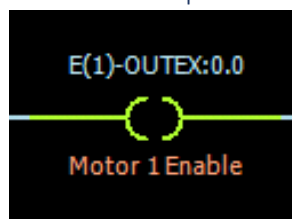
Description	Output state of the selected device(similar to BIN values in usage).
Keyword	OUT
Scope	Device
Data Type	Boolean
Usable Attributes	<ul style="list-style-type: none"> None Full - Directly access the File's containing Byte (becomes Byte type)
Format	<device>-OUT:<file index>
Usage	Read, Write
Example	<p>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 Device Map window for Section2, Device-1 has been mapped to GX20-2. Therefore, 1-OUT:1 will correspond to Output 1 of GX20 Control card 2.</p> 

Extended Inputs



Description	Input object available to the Device		
Keyword	INEX		
Scope	Device		
Data Type	32-Bit Signed Integer / Boolean		
Usable Attributes	<ul style="list-style-type: none"> None Bit Indexing [0 – 31] (becomes Boolean type) 		
Format	<device>-INEX:<file index>.<attribute>		
Usage	Read		
Example	Some Devices may have more data available to them past the standard inputs. The INEX filetype allows you to read from the full range of outputs available to the device. For example, here are the extended outputs for the GX20:		
	Address	Name	Notes
	INEX:0	Sensors/Motor Faults	This is also the data written to when using IN:
			Bit 0 Sensor 1
			Bit 1 Sensor 2
			Bit 2 Motor Fault 1
			Bit 3 Motor Fault 2
	INEX:1	Motor1 Current	Current value of Motor1 in mA
	INEX:2	Motor2 Current	Current value of Motor2 in mA
	INEX:3	Power Voltage	Input voltage on the GX20 card from power supply
	INEX:4	Motor1 Peak Current	Highest current seen on the Motor1 output
	INEX:5	Motor2 Peak Current	Highest current seen on the Motor2 output
	As mentioned in the Device Profiles section, it may be recommended to use the short address tags available for devices instead of directly using the INEX syntax.		

Extended Outputs



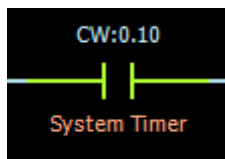
Description	Output object available to the Device			
Keyword	OUTEX			
Scope	Device			
Data Type	32-Bit Signed Integer / Boolean			
Format	<device>-OUTEX:<file index>.<attribute>			
Usable Attributes	<ul style="list-style-type: none">• None (becomes Integer type)• Bit Indexing [0 – 31] (becomes Boolean type)<ul style="list-style-type: none">○ Example: E(2)-OUTEX:0.5 will access the 5th bit of Extended Output 0, on EtherCAT device 2• Bit Mask [b0000, or h00] (becomes Boolean type. Read/Write multiple bits at once)<ul style="list-style-type: none">○ Example: E(3)-OUTEX:1.b1010 or E(3)-OUTEX:1.h0A will write to the 2nd and 4th bits of Extended Output 1, on EtherCAT device 3○ Note: Using this syntax on an Output Command will write to all specified bits. Using this syntax on an Input Command will perform an OR between all specified bits.			
Usage	Read, Write			
Example	Some Devices may have more data available to them past the standard outputs. The OUTEX filetype allows you to write to the full range of outputs available to the device. For example, here are the extended outputs for the GX20:			
	Address	Name	Notes	
	OUTEX:0	Motor Outputs/Direction	This is also the data written to when using OUT:	
			Bit 0	Motor 1 Enable
			Bit 1	Motor 2 Enable
			Bit 2	Motor1 Direction
			Bit 3	Motor2 Direction
OUTEX:1	Motor Speed 1	Value from 0-100 defining speed of Motor1		
OUTEX:2	Motor Speed 2	Value from 0-100 defining speed of Motor2		
As mentioned in the Device Profiles section, it may be recommended to use the short address tags available for devices instead of directly using the OUTEX syntax.				

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

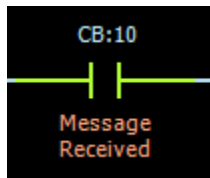
Control Words



Description	Special purpose 32-bit signed integers for device control. Each one has a unique predefined function.																				
Keyword	CW																				
Scope	Global																				
Data Type	32-Bit Signed Integer																				
Usable Attributes	<ul style="list-style-type: none">• None• Bit Indexing [0 – 31] (Becomes Boolean type)																				
Format	CW:<file index>.<attribute>																				
Usage	<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, IntelECAT : 80)</td></tr></table>			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, IntelECAT : 80)
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, IntelECAT : 80)																			

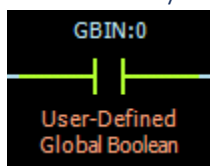
	32-39	R	Receive Message Data
	40-49	R	Data from Ethernet/IP Master (Refer to Protocols section)
	50-59	R/W	Data to Ethernet/IP Master (Refer to Protocols section)
<i>*Unmentioned CW indexes are currently reserved</i>			
Example	The example is using a Normally Open command with the 10 th bit of CW:0, 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	<ul style="list-style-type: none"> None Full - Directly access the File's containing Integer (becomes Integer type) 							
Format	CB:<file index>							
Usage	<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



Description	A global bit value that can be 0 or 1. The IntellectCAT has a maximum limit of 24 GBINs at once.
Keyword	GBIN
Scope	Global
Data Type	Boolean
Usable Attributes	<ul style="list-style-type: none"> 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 IntellectCAT has a maximum limit of 20 GREGs at once.
Keyword	GREG
Scope	Global
Data Type	32-Bit Signed Integer
Usable Attributes	<ul style="list-style-type: none"> 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.

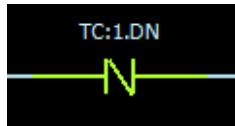
- [Normally Open](#)
- [Normally Closed](#)
- [Compare](#)

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, XLK:UP.IN, etc.)

Normally Closed Command

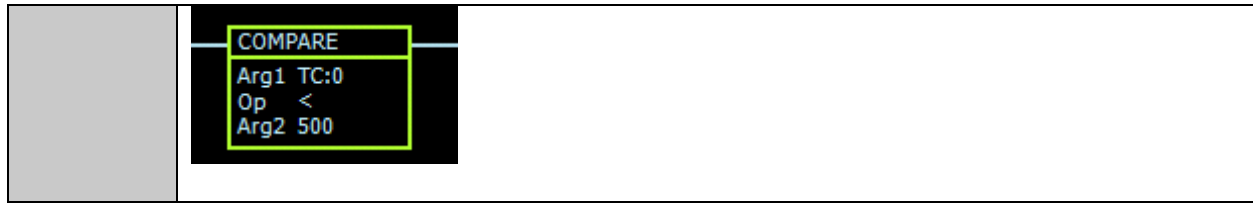


Description	An input command whose condition is true when the addressed bit value is inactive.
Type	Input
Parameters	Address – the referenced bit to read from (Ex. <i>IN:1</i> , <i>OUT:2</i> , <i>TC:2.TM</i>).
File Types	All Boolean Types. (BIN, IN, OUT, CB, Register Bits, TC Attributes, XLK:DN.IN, etc.)

Compare Command



Description	A command whose condition depends on the logical comparison of the values of two referenced files.
Type	Input
Parameters	<ul style="list-style-type: none"> • Arg1 – The first argument (referenced file or constant) for comparison. (Ex. <i>10</i>, <i>REG:5</i>, <i>"text"</i>, <i>BAR:8,10</i>). • Arg2 – The second argument (referenced file or constant) for comparison. (Ex. <i>10</i>, <i>REG:5</i>, <i>"text"</i>, <i>BAR:8,10</i>). • Operation – The logical comparison to make between the 2 arguments. <ul style="list-style-type: none"> ➤ 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	



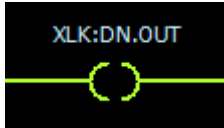
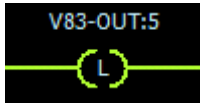
Output Commands

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.

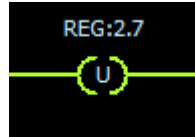
- [Output Coil](#)
- [Move](#)
- [Math](#)
- [Timer/Counter](#)
- [Reset](#)
- [Send Message](#)

Output Coil Command

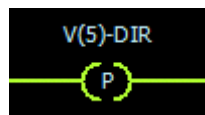


Description	This command sets the addressed bit to true or false. When the rung condition is true, the addressed bit or output is set to true (1 or high) and with rung condition false, the bit or output is set to false (0 or low).
Type	Output Coil
Options	<ul style="list-style-type: none"> • Address – the referenced bit value to write to (Ex. Out:1, REG:5.3, etc). • Output Type – Controls the behavior of the command. <ul style="list-style-type: none"> ➤ Normal () – If the rung is true, it writes 1, otherwise it writes 0.  ➤ Latch (L) – If the rung is true, it writes 1 every time the rung is executed, otherwise if the rung condition is false, it does nothing. 

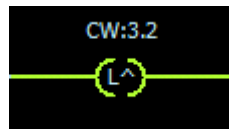
- Unlatch (U) – If the rung is true, it writes 0 every time the rung is executed, otherwise if the rung condition is false, it does nothing.



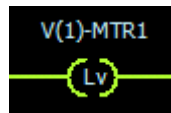
- Pulse – If the rung condition change from false to true, the output is set to 1. The next time the command is executed, the output will be set to 0. If no change in rung condition, it does not do anything.



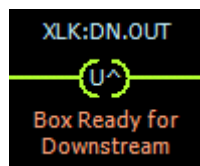
- Latch on Rising Edge – If the rung condition change from false to true, the output is set to 1. If no change in rung condition, it does not do anything.



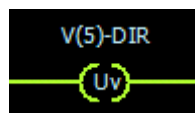
- Latch On Falling Edge – If the rung condition change from true to false, the output is set to 1. If no change in rung condition, it does not do anything.




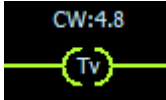
- Unlatch On Rising Edge – If the rung condition change from true to false, the output is set to 0. If no change in rung condition, it does not do anything.



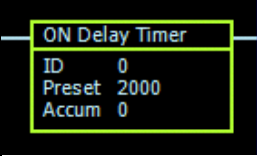
- Unlatch On Falling Edge – If the rung condition change from false to true, the output is set to 0. If no change in rung condition, it does not do anything.



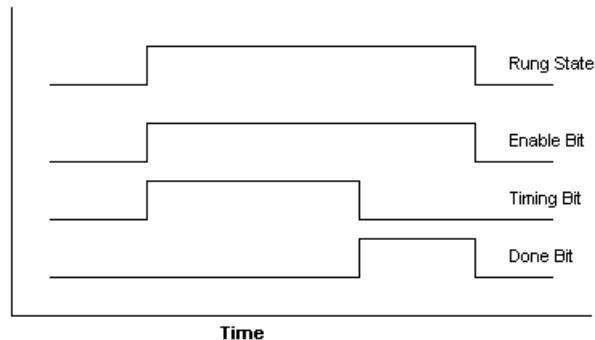
- Toggle On Rising Edge – If the rung condition change from false to true, the output is inverted (0 to 1, or 1 to 0). If no change in rung condition, it does not do anything.

	 <p>➤ Toggle On Falling Edge – If the rung condition change from true to false, the output is inverted (0 to 1, or 1 to 0). If no change in rung condition, it does not do anything. This type of command only effects the output when the rung condition changes.</p> 
File Types	Boolean files (BIN, OUT, CB, GBIN), Register Bits (REG:#.x, CW:#.x, GREG:#.x), Interlock (eg XLK:DN.OUT XLK:UP.OUT ...)

Timer/Counter Command

Description	<p>The Timer/Counter command makes use of either a Timer or Counter register for its functionality</p> <ul style="list-style-type: none"> • 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	<div>  <ul style="list-style-type: none"> • 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. <i>For timers, this value is in milliseconds.</i> • Type – Controls the behavior of the command. <div style="background-color: #ffcc00; padding: 10px; border: 1px solid black;"> <p>NOTE: 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>Types:</p> <p>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</p> </div>

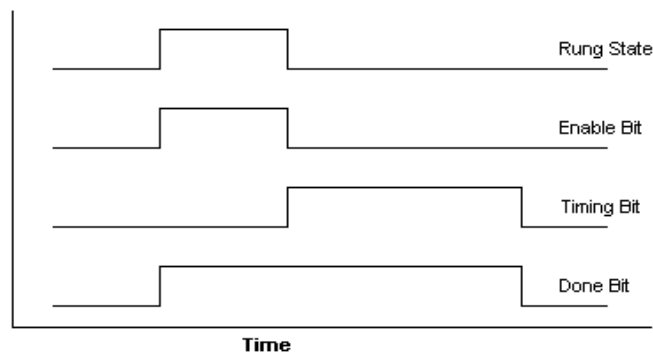
and Enable Bit are reset as soon as the rung state becomes false. The Accumulator is reset to 0 until the rung condition becomes true again.



ON Delay Timing Diagram

OFF Delay Timer	
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

UP Counter	
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.

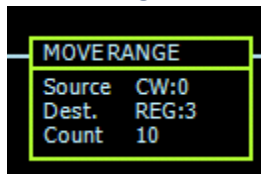
File Types	TC
-------------------	----

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	<ul style="list-style-type: none"> • 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)

Move Range Command



Description	Copies a range of Files starting with the specified source address to the specified Destination address. The amount copied is specified by the Count.
Type	Output
Parameters	<ul style="list-style-type: none"> • 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)
Example	In the listed example, 10 files are being copied from the address starting at CW:0 to the address starting at REG:3. Below is shown the Property window for this command. The range of values from CW:0 to CW:9 will be copied to the range of REG:3 to REG:12. This command is highly useful when many values need to be moved at once.

	▼ Command: MoveRange	
	SourceStart	CW:0
	DestinationStart	REG:3
	Count	10
	SrcRange	CW:0 - CW:9
	DestRange	REG:3 - REG:12

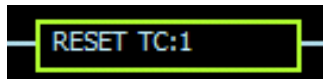
Math Command

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

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	<ul style="list-style-type: none"> • 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. <div style="border: 1px solid black; background-color: #FFD700; padding: 5px; margin: 10px 0;"> <p>NOTE: 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"> ➤ <i>None</i> – the result is the value of Arg1. Arg2 is ignored. ➤ <i>Addition</i> ➤ <i>Subtraction</i> ➤ <i>Multiplication</i> ➤ <i>Division</i> ➤ <i>Power/Exponentiation</i> ➤ <i>BitAND</i> – Bitwise AND ➤ <i>BitOR</i> – Bitwise OR ➤ <i>BitXOR</i> – Bitwise Exclusive OR ➤ <i>BitShiftLeft</i> – Shifts Arg1's bit value by a specified number of digits (Arg2's value) to the left. ➤ <i>BitShiftRight</i> – Shifts Arg1's bit value by a specified number of digits (Arg2's value) to the right.

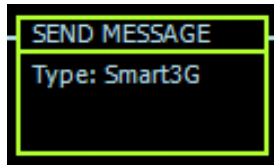
	<ul style="list-style-type: none"> • Unary Operation – operation to perform on the result of the Binary Operation. <ul style="list-style-type: none"> ➤ <i>None</i> ➤ <i>Negative</i> – Negates the value. ➤ <i>Bitwise Inversion</i> – Invert the bit value. ➤ <i>Absolute</i> – Absolute value. ➤ <i>Square Root</i> ➤ <i>Sine</i> ➤ <i>Cosine</i> ➤ <i>Tangent</i> ➤ <i>Cosecant</i> ➤ <i>Secant</i> ➤ <i>Cotangent</i> ➤ <i>Natural Logarithm</i> ➤ <i>Common Logarithm</i>
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	<ul style="list-style-type: none"> • Address – The Timer/Counter to reset. (Ex. TC:3, TC:7 etc)
File Types	TC

Send Message Command



Description	<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><p>NOTE: This command only triggers when the rung state transitions to true.</p></div>																								
Type	Output																								
Parameters	<div><div><ul style="list-style-type: none">Type<ul style="list-style-type: none">➤ Interlock – This type sends the data specified by the source address field over an interlock. This allows Conveyor Sections to send up to 2 Integers over Interlocks. Refer to the XLK FileType section for more information on this SendMessage Type.</div><table><tr><td>Interlock Channel</td><td>Which Interlock Channel to send the Message over</td></tr><tr><td>Source Address</td><td>The start address of the data we want to send</td></tr><tr><td>Count</td><td>How many Files we want to send in this message (Max 2)</td></tr></table><div><ul style="list-style-type: none">➤ IntelleCAT – This type sends an explicit message to another IntelleCAT Gateway. 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 Card. 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.</div><table><tr><td></td><td>Control Word</td><td>Expected Value</td></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><ul style="list-style-type: none">IntelleCAT – Any integerSmart3G – 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:<ul style="list-style-type: none">IntelleCAT – 80Smart3G – 67</td></tr><tr><td>Receive Buffer</td><td>CW:32 - CW:39</td><td><ul style="list-style-type: none">IntelleCAT – Any integerSmart3G – 0-255</td></tr></table></div>	Interlock Channel	Which Interlock Channel to send the Message over	Source Address	The start address of the data we want to send	Count	How many Files we want to send in this message (Max 2)		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">IntelleCAT – Any integerSmart3G – 0-255	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">IntelleCAT – 80Smart3G – 67	Receive Buffer	CW:32 - CW:39	<ul style="list-style-type: none">IntelleCAT – Any integerSmart3G – 0-255
Interlock Channel	Which Interlock Channel to send the Message over																								
Source Address	The start address of the data we want to send																								
Count	How many Files we want to send in this message (Max 2)																								
	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">IntelleCAT – Any integerSmart3G – 0-255																							
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">IntelleCAT – 80Smart3G – 67																							
Receive Buffer	CW:32 - CW:39	<ul style="list-style-type: none">IntelleCAT – Any integerSmart3G – 0-255																							

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

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

Add Class1 Connection

Originator->Target (O->T) Connection Parameters

☒ Connection Point: 100

☐ Connection Tag:

Data Size (bytes): 40 ☒ Run/Idle Header

Target->Originator (T->O) Connection Parameters

☒ Connection Point: 101

☐ Connection Tag:

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).

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

OK Cancel

EIP RX Format (O->T)		
BYTE 0	Control Word	
BYTE 2	Control Word	
...	...	
BYTE 38	Control Word	
BYTE 40	Outputs	Device 1
BYTE 41	Speed	
BYTE 42	Outputs	Device 2
...	...	
BYTE 43	Speed	
BYTE 44	Outputs	Device 3
BYTE 45	Speed	
BYTE 46	Outputs	Device 4
BYTE 47	Speed	
...	...	
BYTE 166	Outputs	Device 64
BYTE 167	Speed	

EIP TX Format (T->O)		
BYTE 0	Control Word	
BYTE 2	Control Word	
...	...	
BYTE 38	Control Word	
BYTE 40[3:0]	Inputs	Device 1
BYTE 40[7:4]	Outputs	
BYTE 41[6:0]	Speed	
...	...	
BYTE 41[7:7]	Online Status	
...	...	
BYTE 166[3:0]	Inputs	Device 64
BYTE 166[7:4]	Outputs	
BYTE 167[6:0]	Speed	
...	...	
BYTE 167[7:7]	Online Status	

Here is an explanation of the values in the above table:

Control Word	Values written to and read from the IntellectCAT for use in ladder programs. These bytes are unused in Gateway Mode.	
Speed	Percentage value from 0 to 100 command the speed of this the Device's motors	
Outputs	Bit	Usage
	0	M1 Enable
	1	M2 Enable
	2	Dir M1, M2
	3	Dir M2 (Future)
Inputs	Bit	Usage
	0	PE1
	1	PE2
	2	Fault 1
	3	Fault 2
	4	M1 Enable
	5	M2 Enable
	6	Dir M1, M2
	7	Dir M2 (Future)

VIII. Example

Getting Started:

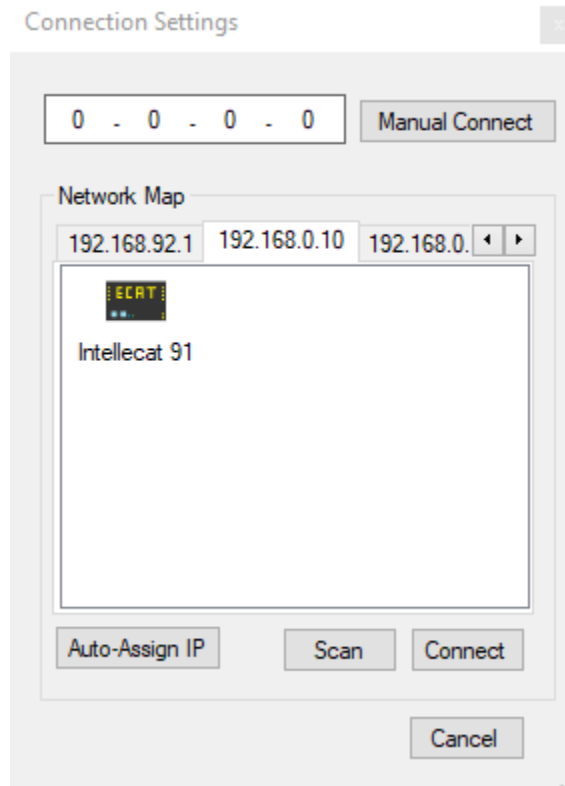
Step 1:

Connect IntellectCAT with your PC using EtherNet cable > Power it up with 24v > Connect GX20 control cards with IntellectCAT 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 IntellectCAT 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 control cards 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 Control Cards if the ID is correct. You will be able to see the status of all the GX20 control cards 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

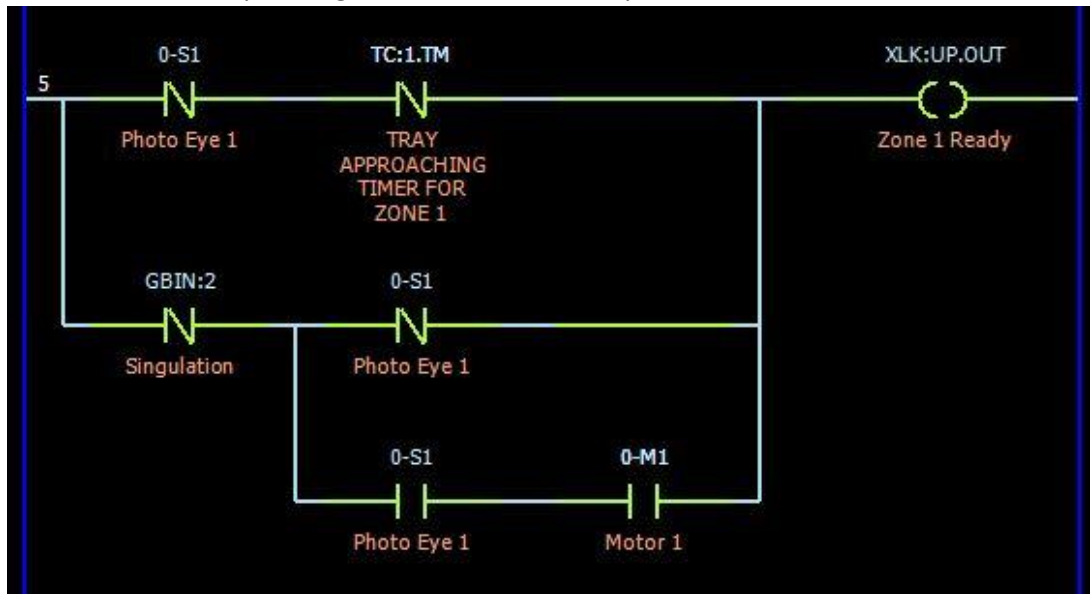
This file type is the state of box available from the previous conveyor section and ready to pull in into the current conveyor section.



XLK:UP.OUT

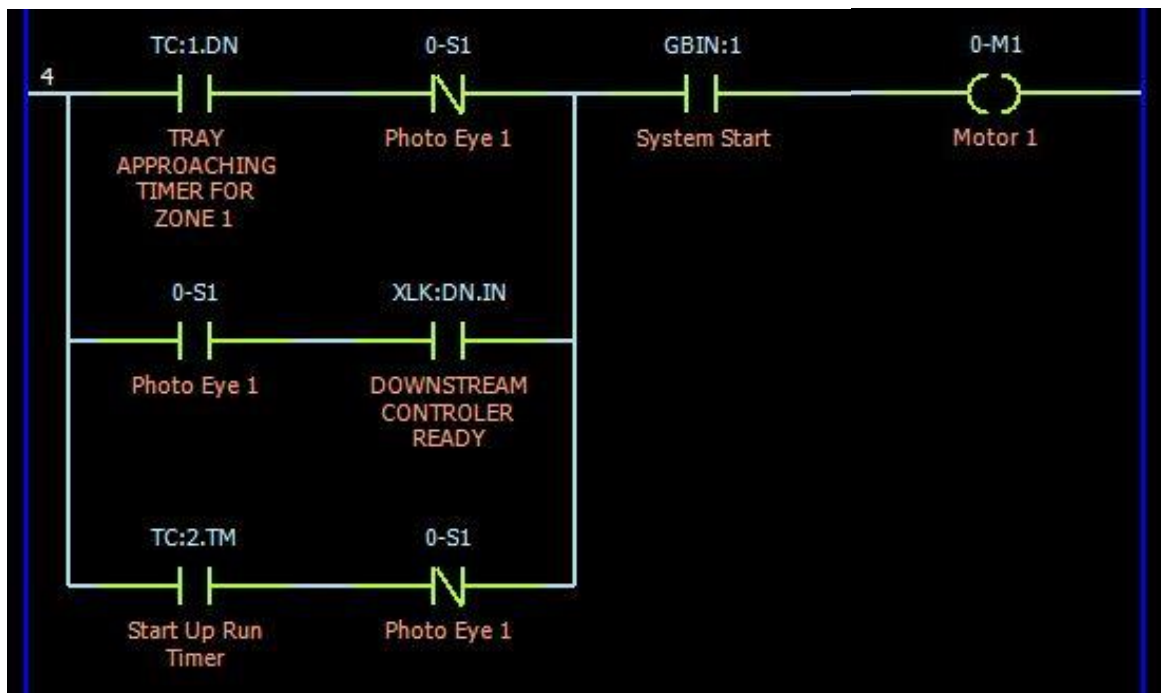
This file type sends the ready status of current conveyor section to the previous conveyor section. When both XLK:UP.IN which is box available to bring in to the current conveyor section and XLK:UP.OUT which is ready to accept box from the previous section are active, package should move from previous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

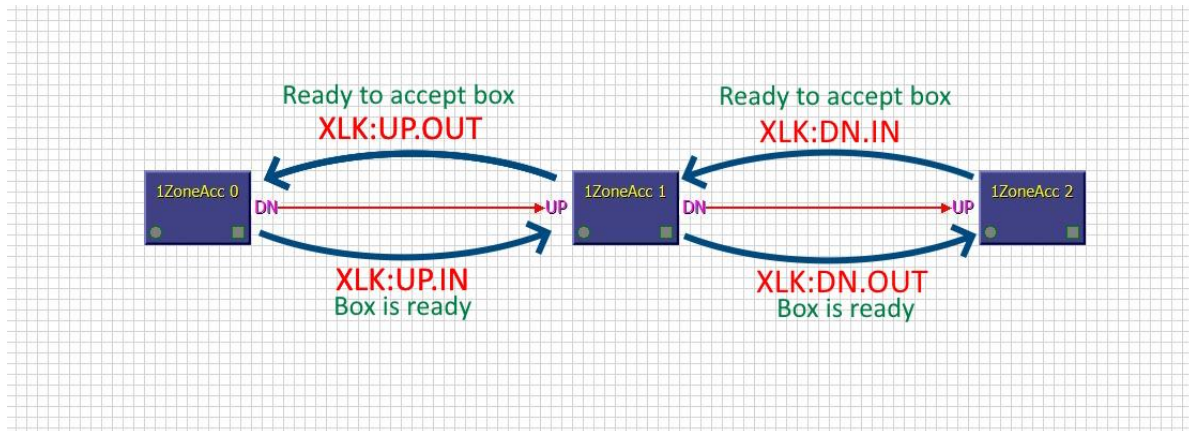


XLK:DN.OUT

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.

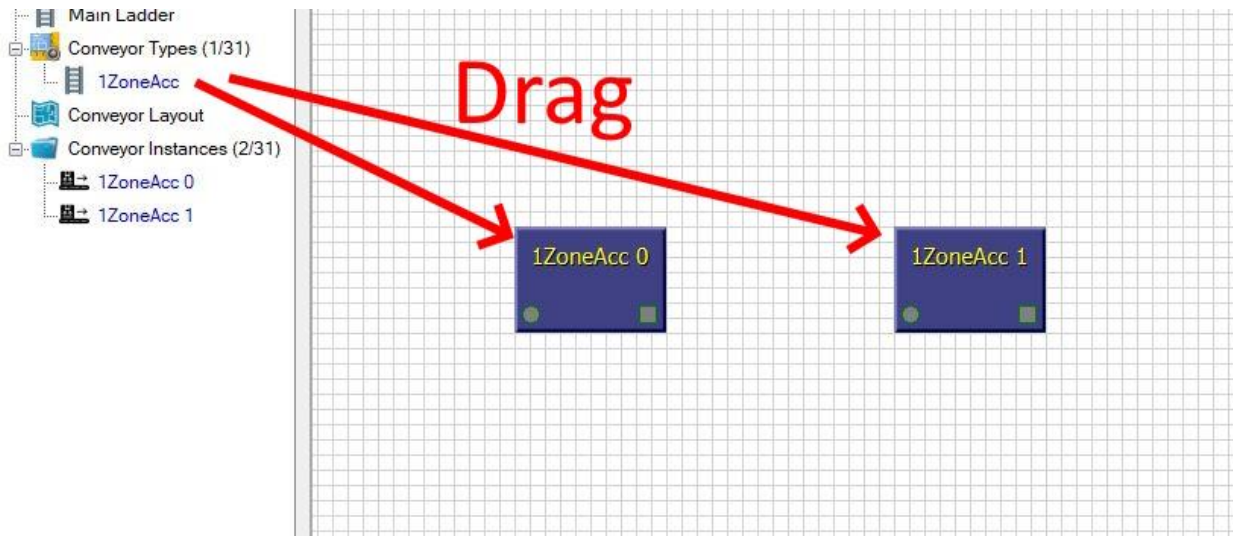


When both XLK:DN.IN and XLK:DN.OUT are in sync, the package should move from current section to the next section.



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.

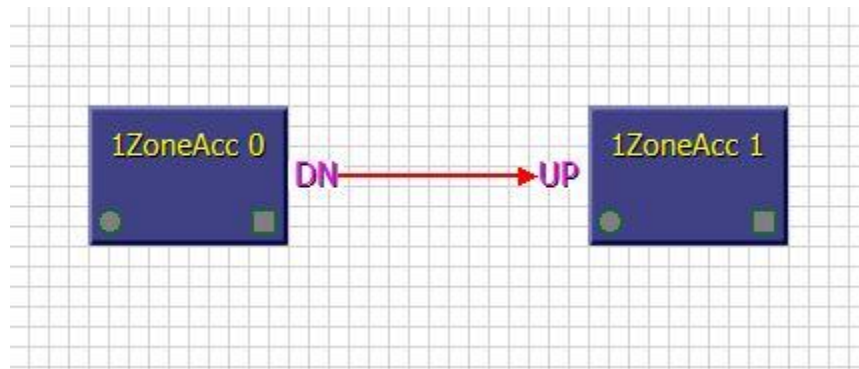


Step 4:

Right click on each Conveyor Sections > Edit Device Map > Select which GX20 control cards 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 sections 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.