

Duplicating Tags and Code for ControlLogix Rev 35+

Add-on instructions (AOIs), user-defined types (UDTs) and code libraries will be used to facilitate project programming with the Logix Designer programming software. AOIs allow one to encapsulate code into function blocks. UDTs allow data to be organized into structures. A library file is an L5X file that contains portions of code. A text editor will be used to modify the L5X file to change tags and descriptions.

You may want to make backup project files (with a different name) frequently while you get comfortable with the procedure. You cannot "undo" an import.

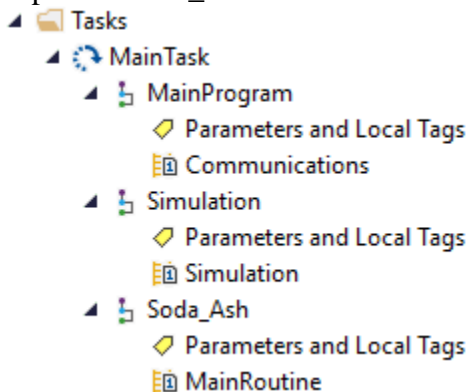
These directions assume you are using NotePad, but any ASCII editor should be fine. DO NOT use a word processor since it will insert extra characters into the file. Logix Designer inserts extra characters in the exported L5X file. WordPad interprets these characters as a "font change" and subsequently Logix Designer does not import the file saved from WordPad. NotePad seems to ignore the special characters and so it is used to generate the files referenced below.

Download the archived library (CLogix20.zip) and extract the L5X files into a folder.

Steps:

1. Start a blank project file – (rev 35 for this semester). Check the processor type for your project. If you do not know, define as an L71 and change later if needed. One can convert an L71 to an L81E, but not vice versa.
2. Create the programs/routines needed for the project.
 - Change "MainRoutine" in "MainProgram" to "Communications" (Right-click on "MainRoutine", select **Properties**, and change name).
 - Create "Simulation" program (Right-click "MainTask", select **Add**, then **New Program**. Enter "Simulation" as the name. Click **OK**).
 - Create "Simulation" routine in "Simulation" program (Right-click "Simulation" program, select **Add**, then **New Routine**. Enter "Simulation" as the name. Make sure "Type" is "Ladder Diagram" and "Assignment" field is "Main". Click **OK**).
 - Create the program for your unit. Create the MainRoutine for your unit.

Example for Soda_Ash below



3. Import the Valve_Disc AOI (includes the Valve_Disc_Type UDT) into your project. Steps to import the Valve_Disc AOI

- In the project tree, expand "Assets", right-mouse the "Add-On Instructions" folder.
 - Select **Import Add-On Instruction ...**
 - Select the Valve_Disc.L5X file and click **Open**.
 - An import configuration window appears. The entry in the "Operation" column indicates what will happen if you click on the "OK" button.
 - If this is a new AOI, the Operation should be "Create." If there are any yellow triangles in the tree in the left pane of the import configuration window, check them and make sure you understand what will happen if you import the AOI. You may want to check the Errors/Warnings. Click on "OK" if no errors.
 - If you want to replace the AOI with an updated version, click on the "Collison Details..." button. The "Property Compare" tab allows you to compare the revision of the existing AOI versus the one being imported. The "Imported References" tab shows the locations in the project that already use the AOI. If you want to use the new AOI, click on the "Overwrite" button below. You will then be taken back to the import configuration window and the "Operation" column has been changed to "Overwrite". Click on the "OK" button.
 - The AOI should appear in the "Add-On Instruction" folder.
4. Import the other device AOI's even if your project does not contains that type or device:
 - Motor_Std AOI (includes Motor_Std_Type UDT)
 - Motor_Conv AOI (includes Motor_Conv_Type UDT)
 - Gate_Slide AOI (includes Gate_Slide_Type UDT)
 - Gate_Flop AOI (includes Gate_Flop_Type UDT)
 5. Import the sequence-associated UDTs and AOIs into your project:
 - In the project file, import the Unit_Type UDT:
 - In the project tree, expand "Assets", right-mouse "User-Defined" in the "Data Types" folder. You may need to expand the data types by clicking on the "►" in front of the "Data Types" folder.
 - Select **Import Data Type ...**
 - Select the Unit_Type.L5X file and click **Open**.
 - An import configuration window appears. Make sure the Operation Field is "Create". If the Operation Field shows "Use existing," it means the data type already exists. Set the Operation Field to "Overwrite" only if the definition truly needs to be corrected. You may want to check the Errors/Warnings. Click on "OK" if no errors.
 - The UDT should appear beneath the "User-Defined" folder in the "Data Types" folder.
 - Import the Seq_Type UDT.
 6. Define the tag for your unit information (of Unit_Type). For example, if your unit is called "Package", then the name of this tag should also be called "Package". The name may be shortened if there is a defined shortened name. Create it as a Controller Tag.
 7. Four sequence library routines are available:
 - Unit1_Sample.L5X: 40-step sequence named "Unit1_Sample"
 - Unit1_Hold.L5X: Hold sequence named "Unit1_Hold"
 - Unit1_Shutdown.L5X: 20-step sequence named "Unit1_Shutdown"
 - Unit1_Eshutdown.L5X: 10-step sequence named "Unit1_EShutdown"; Step 9 has 15-sec delay

These four libraries are meant to be imported as separate routines. The auto-starts are cross-linked. Every unit has at least 4 sequences. Modify each of the four L5X files with the new unit and sequence tags and descriptions, and then import these new L5X files into the project. For the first four unit sequences (hold, shutdown, e-shutdown, and one more):

- Modify **each of the four** L5X files listed above, making the following changes to the text (in the following order, back to the top of the text file for every change):
 - Replace “Sample” with your first sequence name, e.g., “Startup”
 - Replace “Unit1” with your unit name, e.g., “Package”
 - Replace “Unit Desc1” with the description line you want to appear in the first line of your tag description, usually the unit name, not shortened as used for the tags. [Note: there is a space between “Unit” and “Desc1”]
- Save each changed L5X file as a different name, preferably the name of the routine into which the rungs will be imported (one of the sequence routines).
- The three changes must be identical for each of the four files. For the example above, “Sample” is replaced by “Startup” for **all four** files. “Unit1” is replaced by “Package” for **all four** files.
- In each case, **do not start** with a previously fixed L5X file. Start with the L5X file from the library.
- Import the four fixed L5X files as a routine. Procedure:
 - Right-click the program that is your unit name, select *Add*, then *Import Routine...*
 - Select the appropriate L5X file and click *Open*
 - An import configuration window appears.
 - Click on the Tags entry in the tree in the left pane. Normally, this is your opportunity to change tags, but that is what you did with NotePad. The 4th column indicates whether the tag already exists or if it will be created. Even if there are differences, use the existing tag.
 - Click on "OK." Any tag collisions are also noted in the "Errors" window at the bottom of the screen. The Errors window will also indicate if one or more data types were not imported and if the Seq63[127]_Step_Trans add-on instruction was not imported. If you receive these warning messages, fix the problem.
 - Change the routine name to include the Msg offset number, e.g., “Package_040Hold”

If your unit has more than 4 sequences, modify the Unit1_Sample.L5X file again:

- Make the following changes to the text (in the following order):
 - Replace “Sample” with another sequence name, e.g., “Flush”
 - Replace “Unit1” with your unit name, e.g., “Package”
 - Replace “Unit Desc1” with the description line you want to appear in the first line of your tag description, usually the unit name, not shortened as used for the tags.
- Save the changed L5X file as a different name, preferable the name of the sequence.
- Import the L5X file into the appropriate program.
- For all of the sequences, you will need to fix-up the start rung (rung 0) and the auto-start rung (rung 1) to add the xxx.Auto_Start Booleans and .Step_Num clears for the new sequences.

You will need to add your own routines for the abnormal conditions, handling maintenance mode, and un-AFI appropriate auto-starts.

8. Repeat step 7 as needed.

9. You can check for errors. There will be warnings that AFI instructions are detected. These are normal for this stage of the process.
10. Save your project.
11. Add the routines in your unit program that will hold the device AOI's: Gates, Motors, PID_Loops, Valves