

PLC Programming with RSLogix 5000



*How to Program an Allen-Bradley PLC or PAC
with Rockwell Automation's RSLogix 5000*

by Neal Babcock

PLC Programming with RSLogix 5000

*How to Program Allen-
Bradley
ControlLogix and
CompactLogix PLCs
with Rockwell
Automation's RSLogix
5000*

By Neal Babcock

Industrial Automation Series

engineer-and-technician.com

Introduction.....	5
PLCs.....	6
Hardware.....	6
ControlLogix Processor.....	7
I/O Modules.....	7
Software.....	8
Getting The RSLogix 5000 Software.....	8
Ladder Logic.....	8
The Dialect of PLCs.....	9
Equivalent Logic.....	13
Project Scope.....	15
Summarizing the Scope.....	22
Which PLC?.....	22
Lay Out The I/O.....	23
Assigning I/O Addresses.....	25
Running RSLogix.....	25
Tags.....	37
Adding Descriptors To Your I/O.....	39
Writing the Program.....	44
Ladder View.....	44
Setting Up An Overall Control Rung.....	44
Starting a Batch Cycle.....	59
Batching Steps.....	69
Step 1 – Adding City Water.....	69
The Tag Database.....	72
Analog Inputs.....	80
Setting up the Analog Input Card to Calculate Tank Weight.....	81
Setting up the Analog Input Card to Calculate Tank Level.....	83
Back to Batching – Step 1.....	85
Step 2 – Adding Chemical KM.....	91
Step 3 – Adding Chemical KM.....	94
Step 4 – Blending.....	95
Step 5 – Pump to Filling Lines.....	100
Faults.....	102
Valve Position Faults.....	103
Console Status Indicators – Pilot Lights.....	110
Adding Rung Comments.....	117
Connecting To The PLC And Going Online.....	121
RSLogix Emulate 5000.....	124
RSLinx.....	124
Emulator.....	126
Editing in Run Mode.....	150
Run Mode on the Plant Floor.....	154
Add-On Instructions & Function Block Diagram Programming.....	156
Creating a Function Block Diagram.....	158

A Final Note About Our Program.....	163
How Do I . . . ?.....	164
Tips, Shortcuts and Warnings.....	166
Conclusion.....	169

Introduction

The purpose of this book is to teach you how to set up, program and use an Allen-Bradley ControlLogix or CompactLogix. It will also familiarize you with the parts required for a common application.

It will tell you how to use RSLogix 5000 and how to write a ladder logic program.

Since I feel the best way to learn any programming language is by using a real-world example, there is a sample project included in this book. This sample project, which involves a chemical batching process, also contains a Project Scope. The Project Scope, or Functional Specification, or whatever your company might call it, defines in detail how the system is to operate when the project is finished.

You will learn, step by step, how to take a Project Scope and turn it into a working PLC program.

The book will show you how to go online with your PLC to monitor your program to verify your ladder logic and make sure it is functioning properly.

It will show you how to make changes to your program while you are online.

It will show you the keystrokes and mouse movements that you need to know to use RSLogix 5000.

Finally, it provides a number of tips and a Frequently Ask Questions section that will save you hours of frustration.

This book assumes you have a little background with PLCs – perhaps you have worked with other PLCs from other manufacturers or you have helped to install and wire PLCs. Perhaps you are a Mechanical, Chemical or Process Engineer and you need to learn how to use RSLogix 5000.

If you need a more thorough understanding of basic PLC concepts, you might want to try the [Beginner's Guide to PLC Programming](#) *How to Program a PLC (Programmable Logic Controller)*. This ebook, along with the online tutorial, provides an example of how to automate a drill press, while explaining all the basic concepts of PLC programming that are necessary to write a solid PLC program.

The [Beginner's Guide to PLC Programming](#) works well in conjunction with this book, in that it concentrates on basic PLC programming methods that are common to all types of PLCs. In addition, it provides an example of machine operation, whereas this book uses the example of a chemical batching process. Go to engineer-and-technician.com if you would like to learn more about this book.

Nearly all the industrial equipment that you find in a modern manufacturing facility shares one thing in common - computer control. The most commonly used controller is the PLC, or the Programmable Logic Controller, using a programming language called Ladder Logic. The language was developed to make programming easy for people who already understood how switches, relay contacts and coils work. Its format is similar to the electrical style of drawing known as the “ladder diagram”.

The most popular and most widely used manufacturer of PLCs is Rockwell Automation, who produces the Allen-Bradley ControlLogix and CompactLogix series of PLCs. The ControlLogix and CompactLogix families of processors and I/O modules are all programmed using Rockwell’s proprietary software known as RSLogix 5000.

When you are finished with this book, you will be able to sit down in front of any computer running RSLogix 5000 and create a new program. You will be able to edit existing programs. You will be able to professionally document any changes you have made.

Rockwell Automation Technical Support

Unfortunately, we can’t anticipate all the problems you might face as you are troubleshooting a program on the factory floor. There are just too many variables. This is why you must establish a relationship with your local Rockwell Automation technical support team. Get to know them *before* you are in the final stages of a start-up and you run into a problem. They are very helpful and they can save you hours of frustration.

The Rockwell reps are not just technical support personnel; they are skilled engineers that are responsible for running their own projects and writing and troubleshooting their own programs. If you run into a problem, more than likely they have already seen it and have come up with a solution.

Hardware

One of the nice things about Allen-Bradley’s smaller PLCs is the relative simplicity of assembling the hardware to create a system.

First, let’s see what it takes to assemble a ControlLogix system. You only need to have a few components: a processor, a power supply, a rack and some I/O modules.

ControlLogix Processor

At the time of this writing, there are 15 ControlLogix processors available. For our application, the 1756-L55 processor will be fine.

For your future projects, you will have to consider a number of factors before you make the choice of your processor. Utilize your Rockwell representative and Rockwell's website (www.ab.com) to help you in your choice.

All the processors use RSLogix 5000, so any program you write for one processor could be adapted to run any other 1756 processor.

I/O Modules

For our system, we need discrete inputs, discrete outputs and analog inputs. These modules will work fine for our application:

1756-IA16 Digital AC Input Module (16 discrete inputs)

1756-OA16 Digital AC Outputs (16 discrete outputs)

1756-IF8 Analog Modules (8 single-ended analog inputs)

Project Scope

We will use a batching operation as an example. Batching, as you may know, is the term that describes the mixing of assorted ingredients to make a finished product.

There are techniques that are common to batching, whether you are making soap or cake mix. We are going to write a program that mixes a hypothetical window cleaner.

Someone has to define the batching procedure. Usually, this is done by a process engineer or a chemical engineer. If the job of defining the project is done well, a document called a Project Scope (or something similar) is generated.

It is extremely important that you clearly understand the entire process that is defined in the scope. If you have any questions or concerns, you need to resolve those before you begin programming. If you don't, then the responsibility of errors and omissions, and perhaps the blame, may be placed on you.

If you bring up questions that result in changes to the defined sequence of operations, ask the originator to revise the Project Scope. In fact, it is not uncommon for a Project Scope to undergo a number of revisions.

If there is a change that is not documented in the scope, you should document it by getting an email from the originator that explains the change. If nothing else, you want to make sure you understand what the change involves.

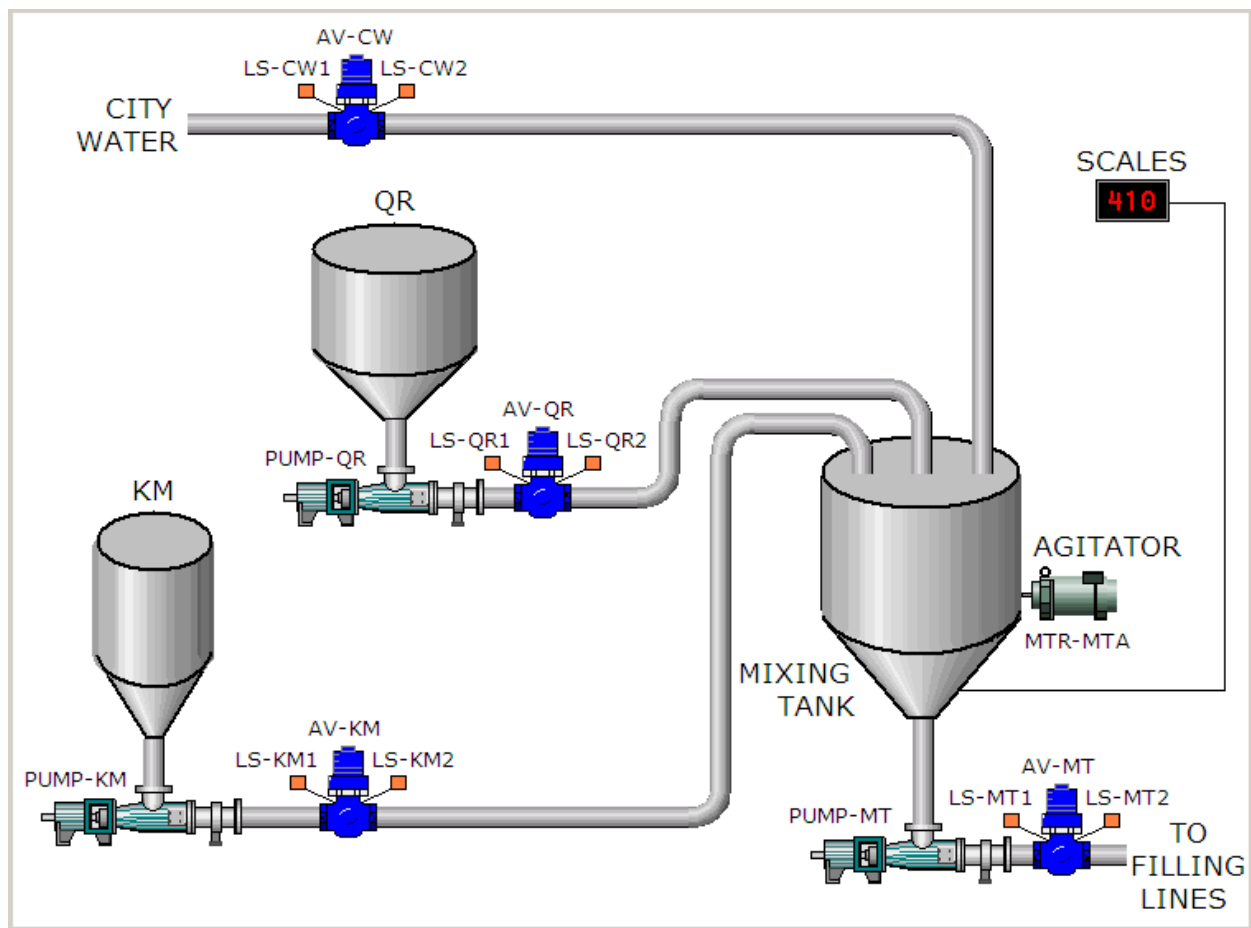
For our project, the project scope is as follows.

Hyper-Glass Cleaner Batching Project Scope

Goal

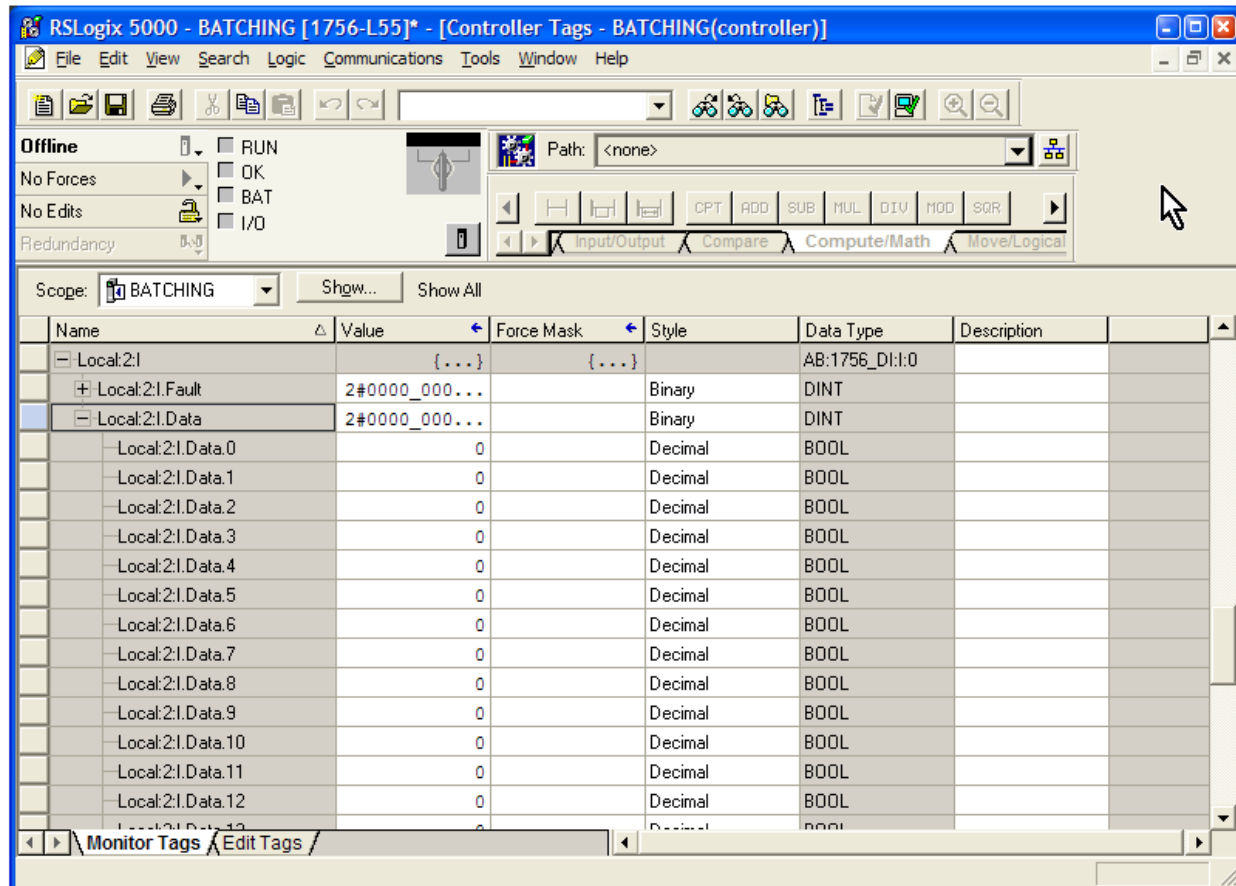
The goal of this project is to install a new automated batching system for mixing Hyper-Glass Cleaner.

Overview



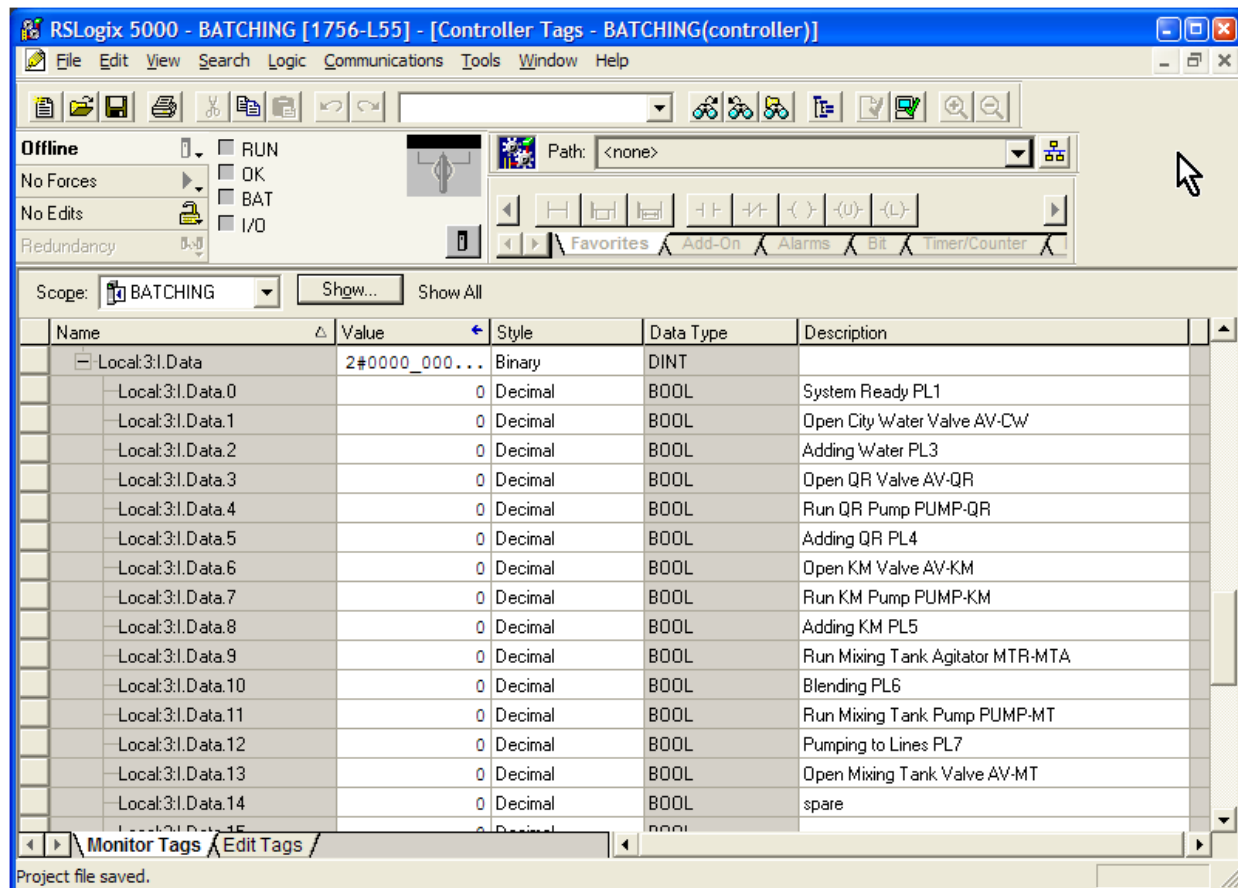
Three ingredients (city water, ingredient QR and ingredient KM) are added in specified amounts by weight to the Mixing Tank. After all the ingredients have been added to the Mixing Tank, the mixture is blended by running the agitator for a given time. When the blending time is complete, the finished product is pumped to the Filling Lines for bottling and final packaging.

Let's do the same with the descriptors for the first discrete input card. Expand "Local:2I". Expand Local:2I.Data.



These are the tags that define the actual inputs on the card. Like with the analog card, this is where you look to see if you have a signal on an input.

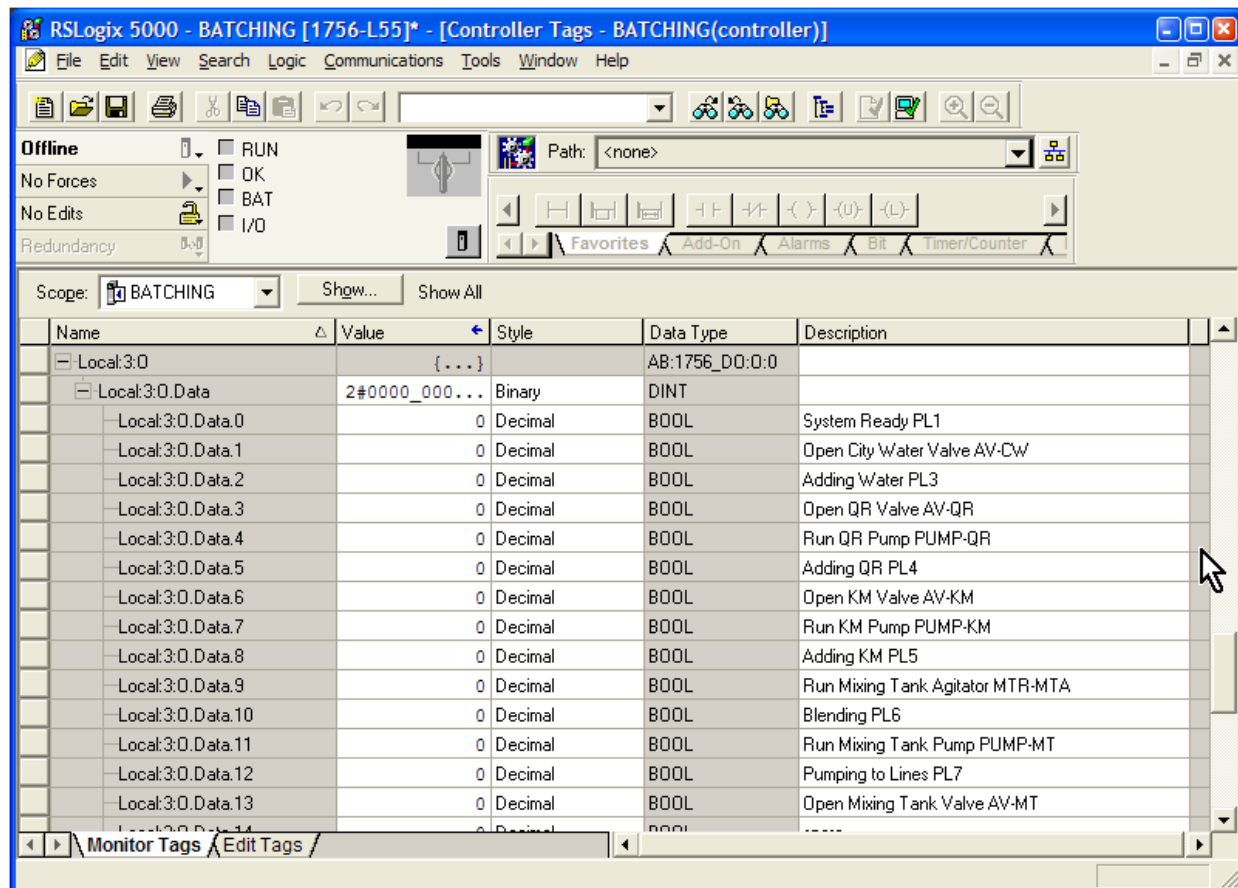
Copy the descriptions from the spreadsheet for this card.



I have hidden the Force Mask column and stretched the Description column to see the full description.

You can switch columns on or off in the Controller Tags window by selecting View > Toggle Column

Copy the descriptions from the spreadsheet for the output card.



This completes the descriptions for the I/O.

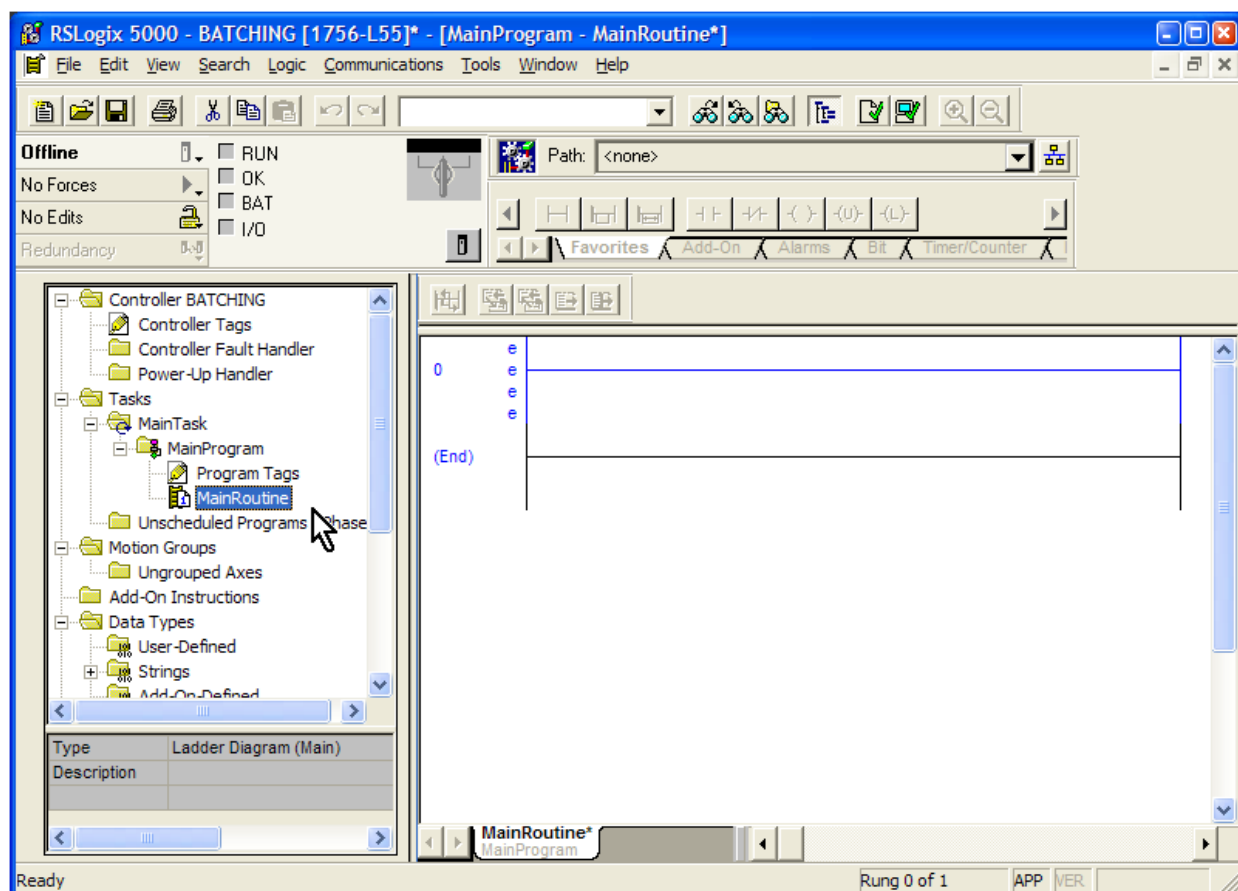
It is a good idea to save your work frequently. This is done in RSLogix like it is in any other Windows program (CTRL-S, or File > Save or).

Writing the Program

Ladder View

Open the Controller Organizer, expand the “Tasks” folder, and expand the “Main Program” folder.

Click on “Main Routine” and you should see this.



Setting Up An Overall Control Rung

Typically, a program will start with some kind of overall or master control rung. This rung will define a bit that must be on for the entire system to operate, and we include bits that we know must be true for the whole system to run.


In this project, we certainly want the E-Stop to be part of this logic. Our E-Stop (or, emergency stop) pushbutton switch is wired in such a way that the input must be on for the system to operate.

We
to use
Stop
in this
Find
XIC

You can add a rung by right-clicking on the rung number and selecting “Add Rung” from the dropdown menu.

You can also press CTRL-R.

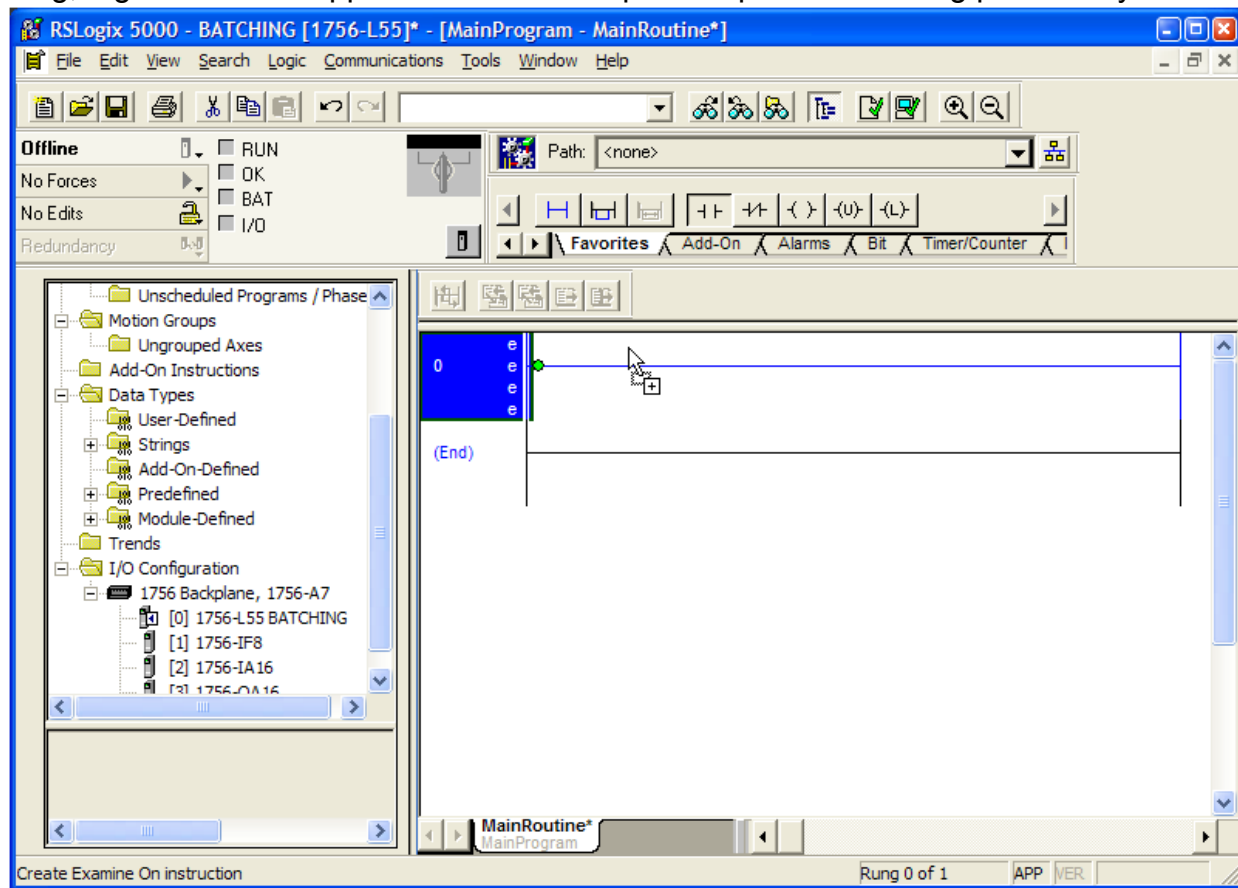
want
the E-
input
rung.
the

(examine if closed)  tool button in the User menu.

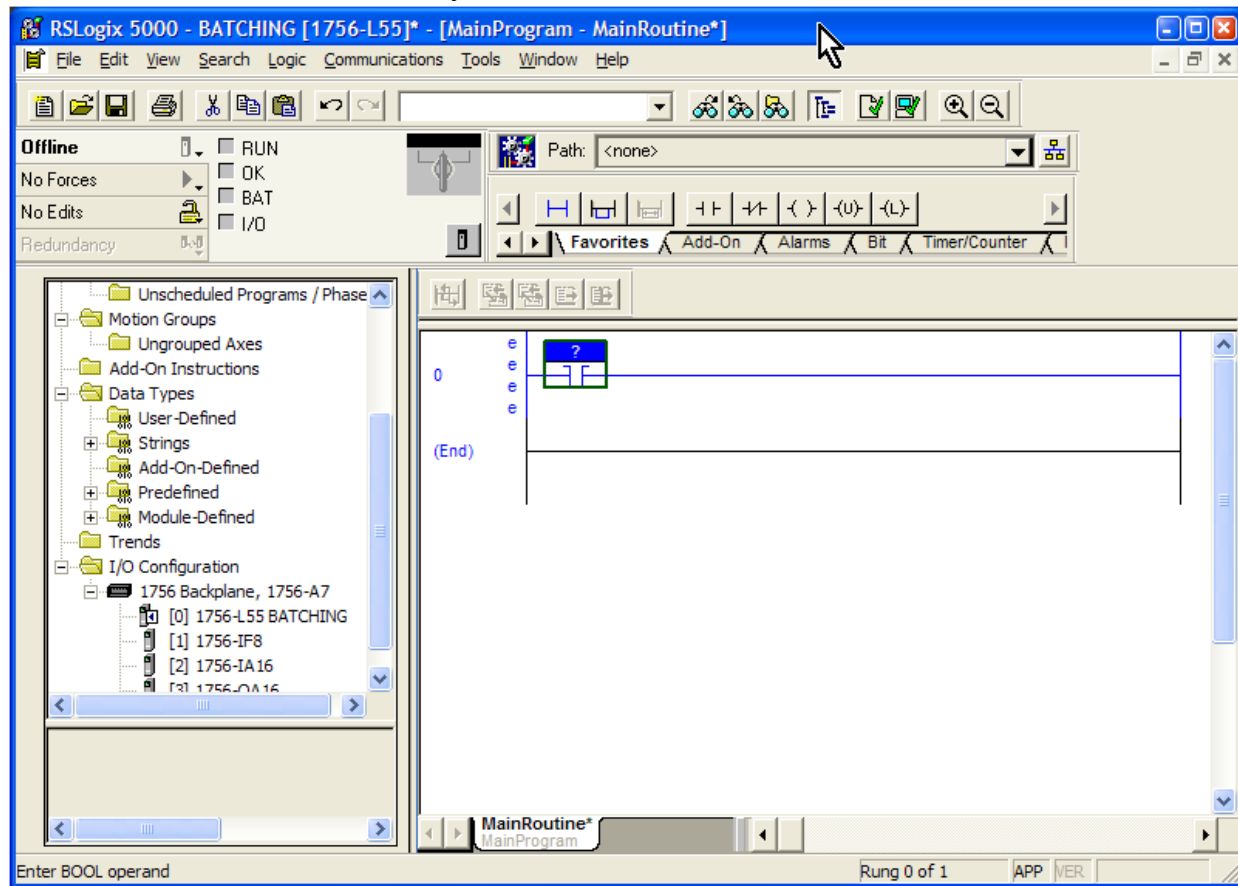
You can insert the instruction in a couple of ways. Simply clicking on the XIC icon will add it to the rung.

You can also click, hold and drag the tool to the spot in the rung where you want it inserted.

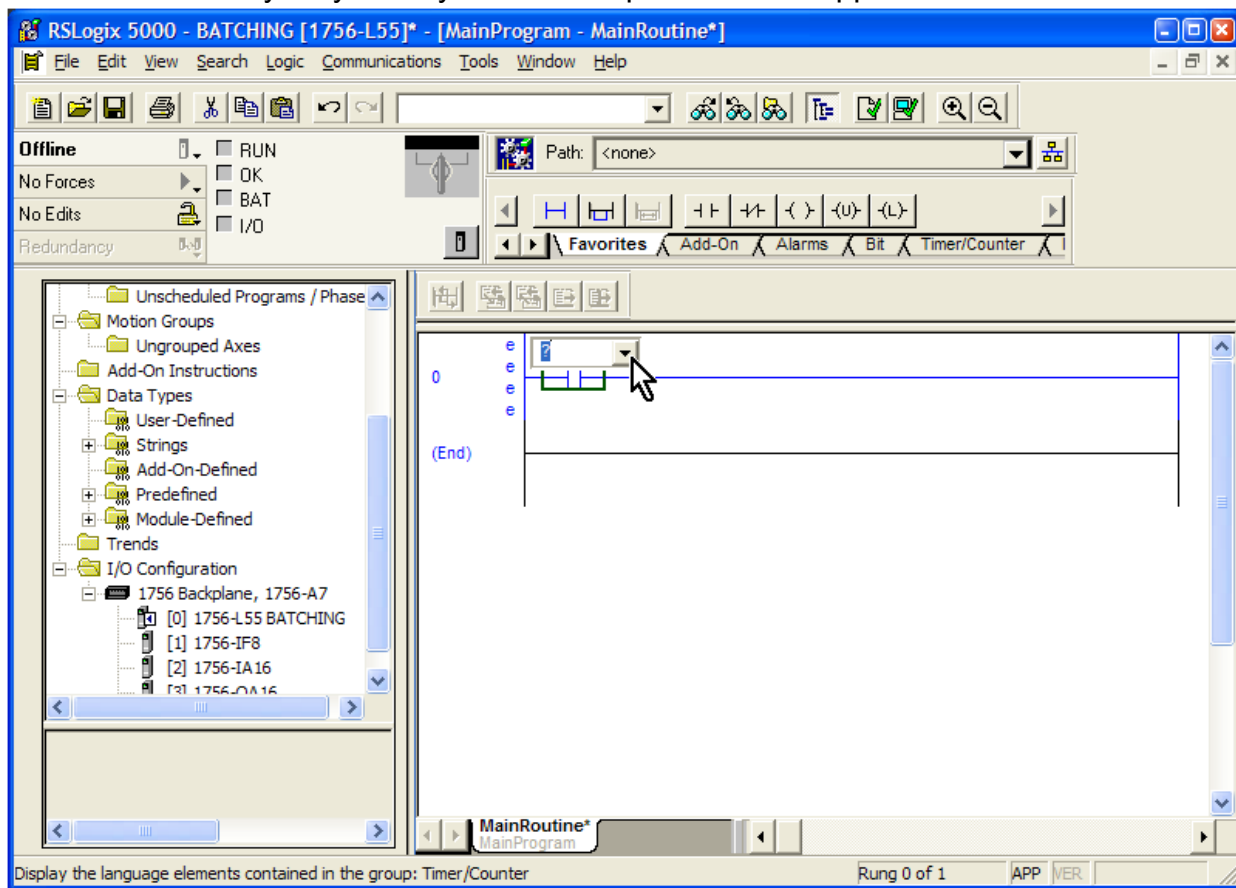
Click and drag it toward the new rung you just created. You will see that as you get near the rung, a green dot will appear. Green dots represent possible landing points for your instruction.



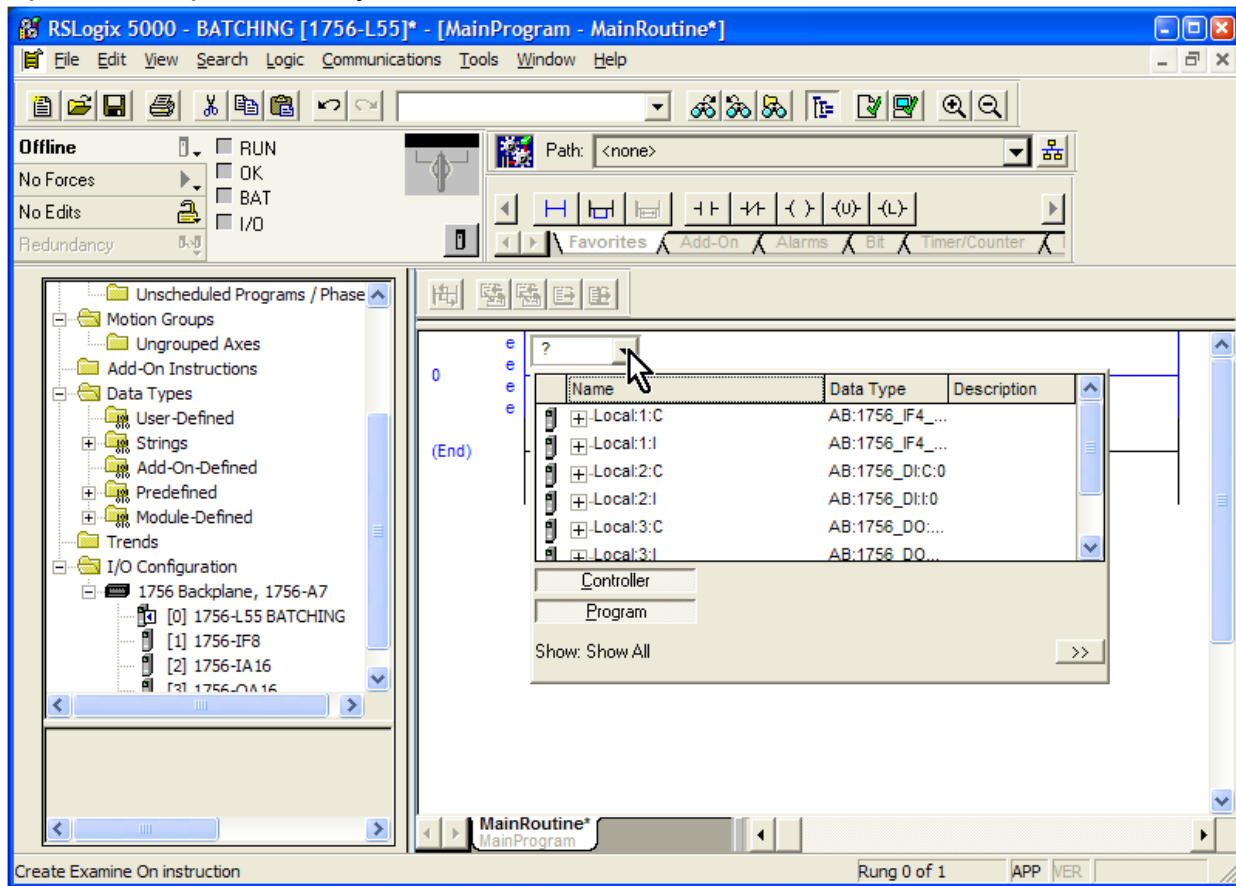
Release the mouse button and your screen should look like this.



Press the enter key on your keyboard. A dropdown menu appears above the instruction.



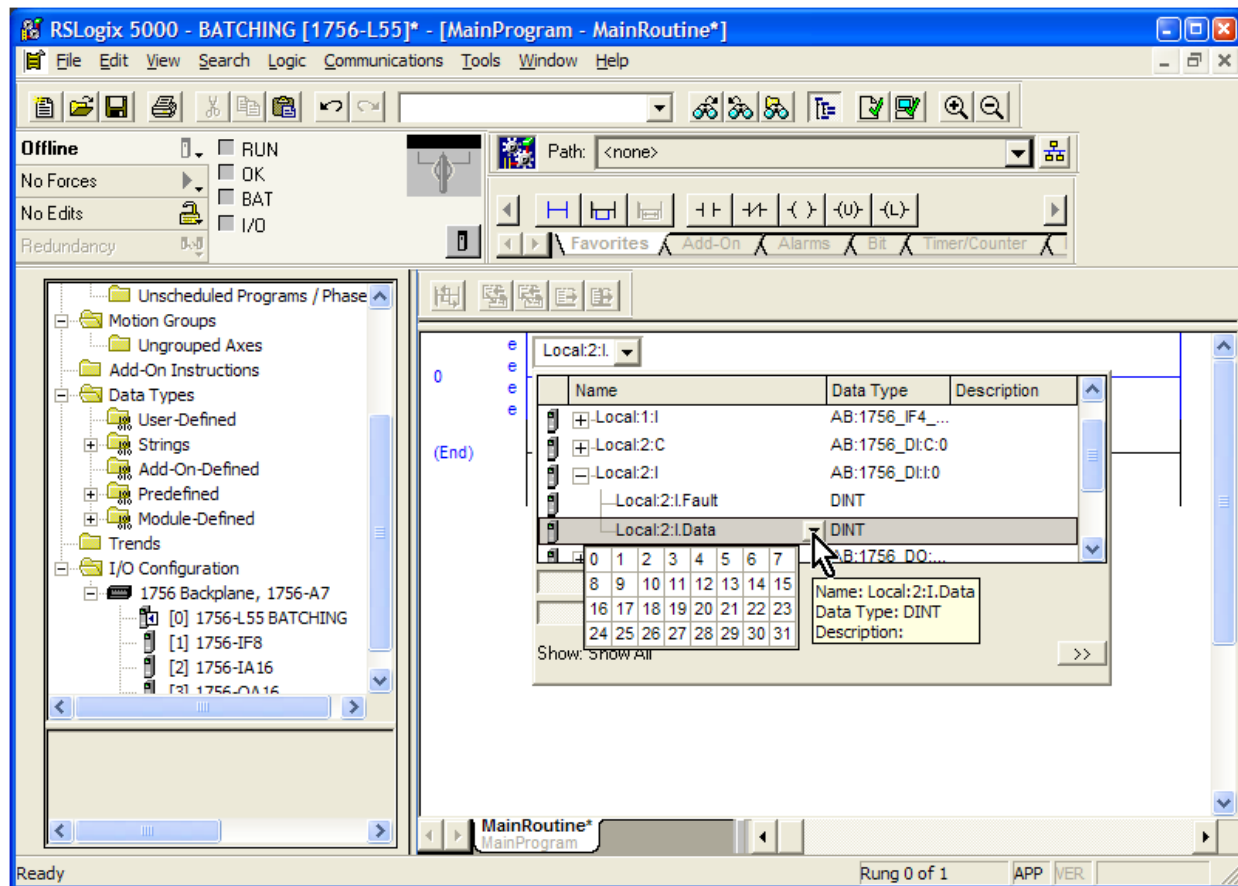
Open the dropdown and your screen now looks like this.



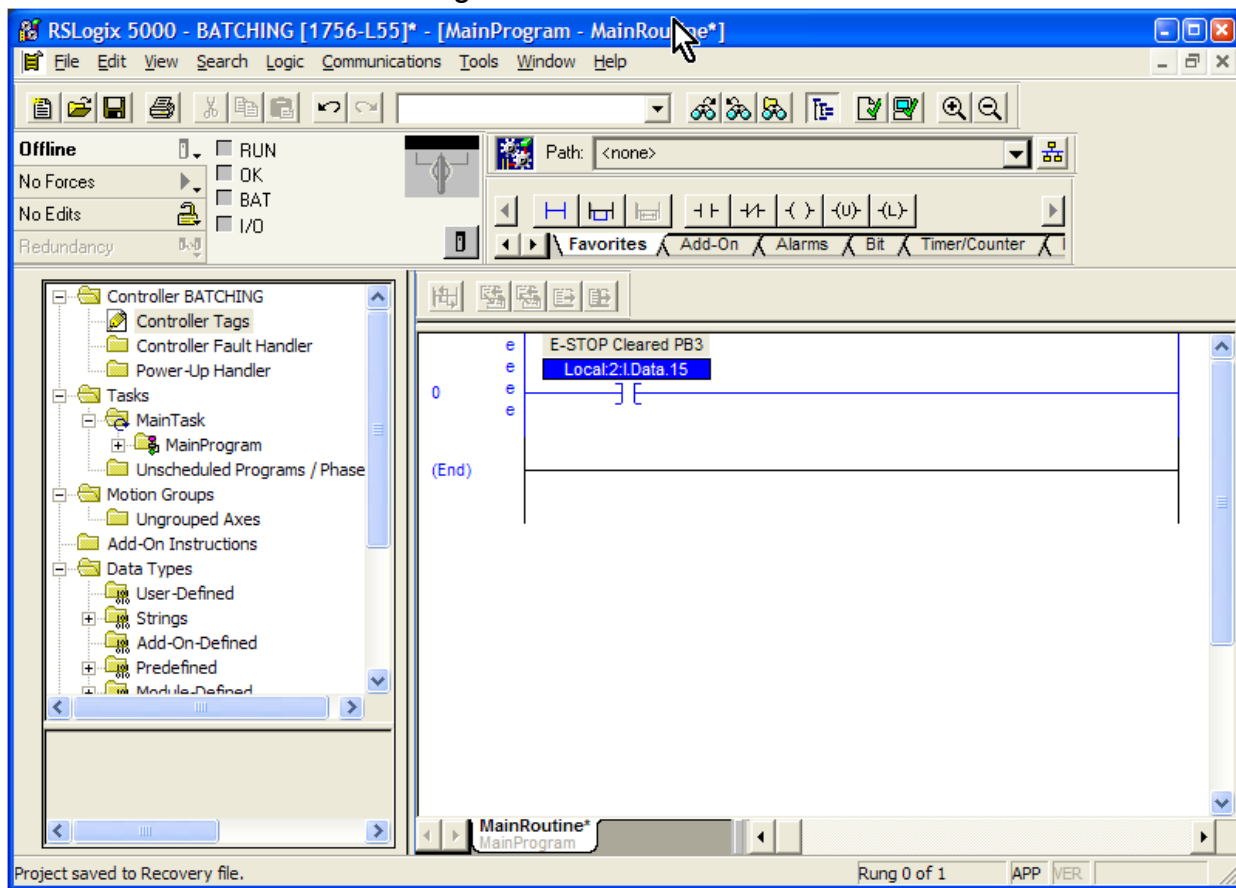
Notice that all the tag groups from our I/O are now showing.


From the I/O list, we see that the E-Stop switch is wired to the last position on the input card, giving it an address of Local:2:I.Data.15. We want to find that tag in this window.

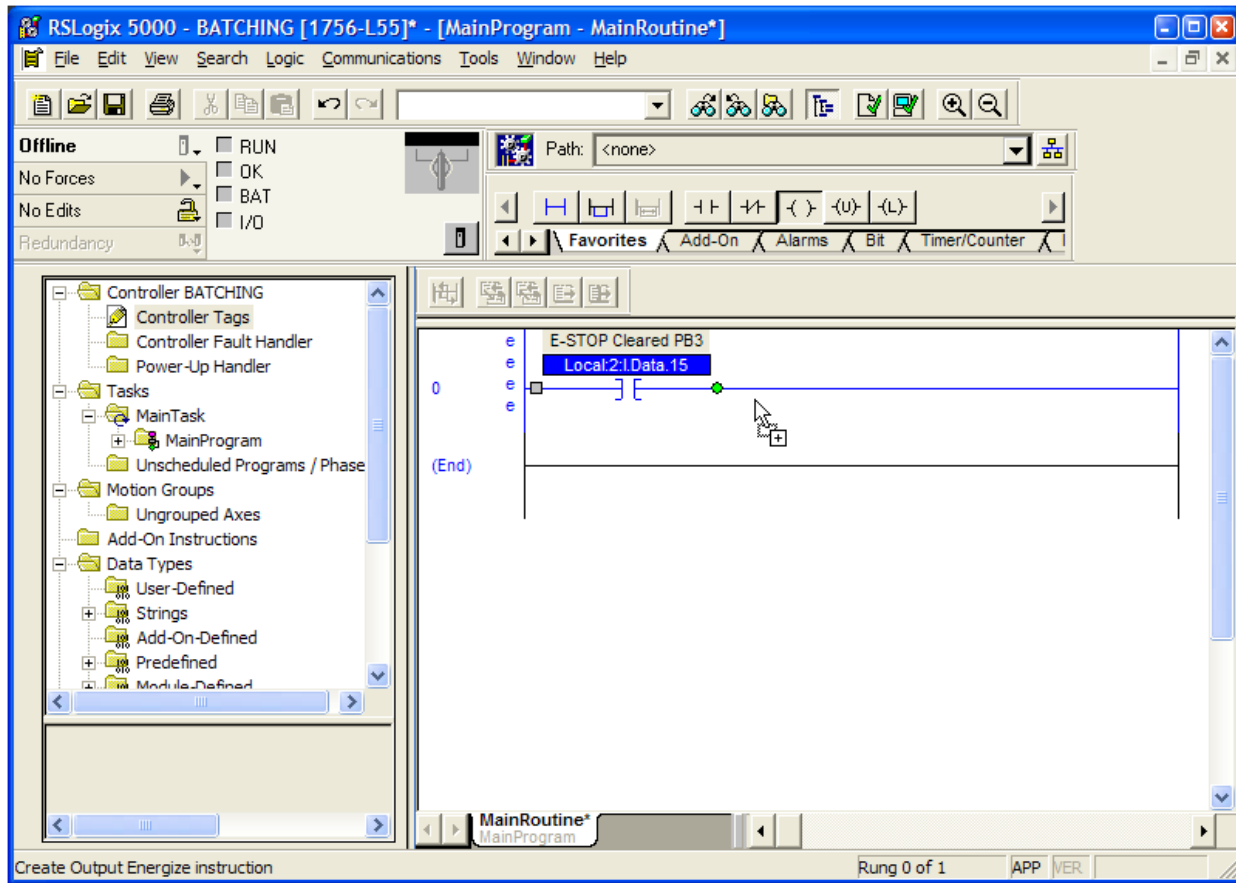
Expand the tag group Local:2:I, then expand the tag group Local:2:I.Data.



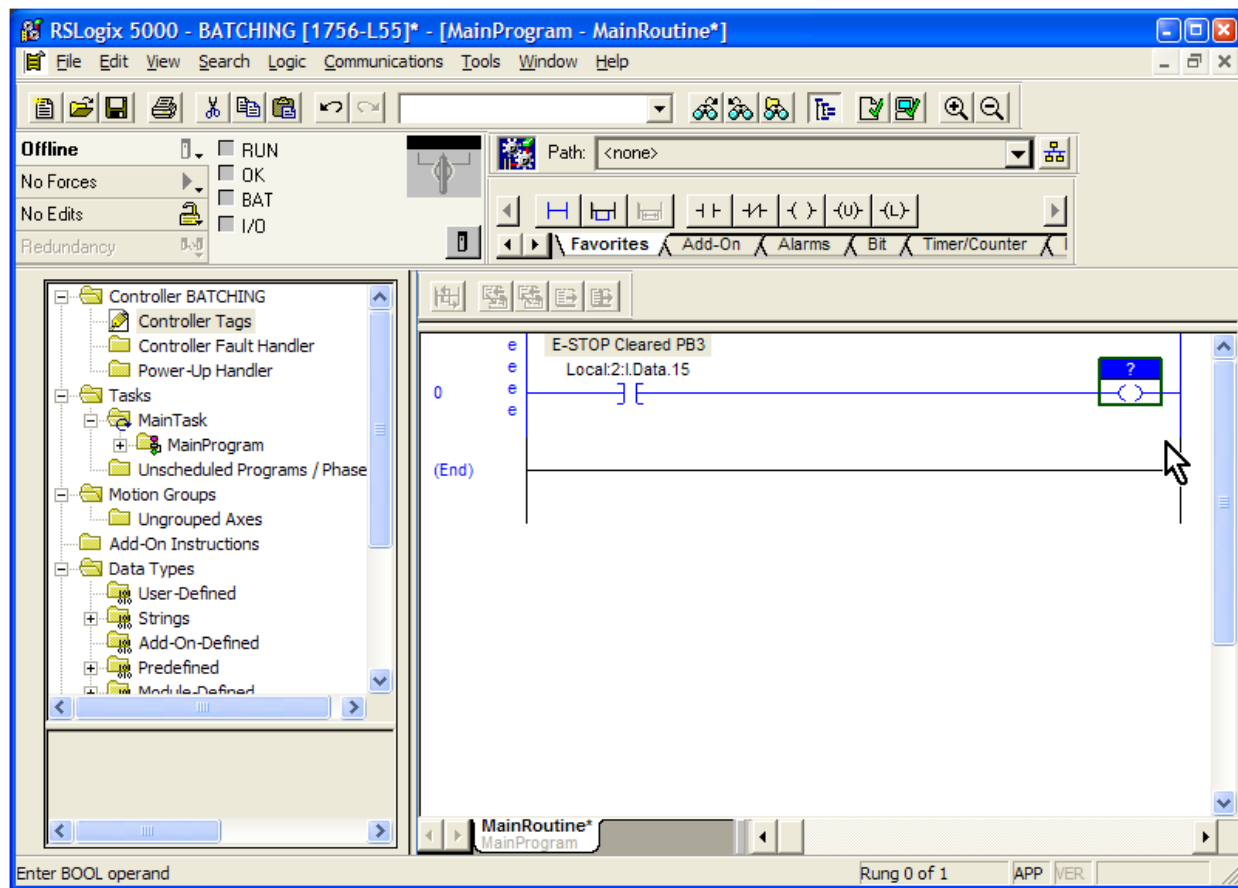
Click on bit 15 in the box to assign that address to the instruction.



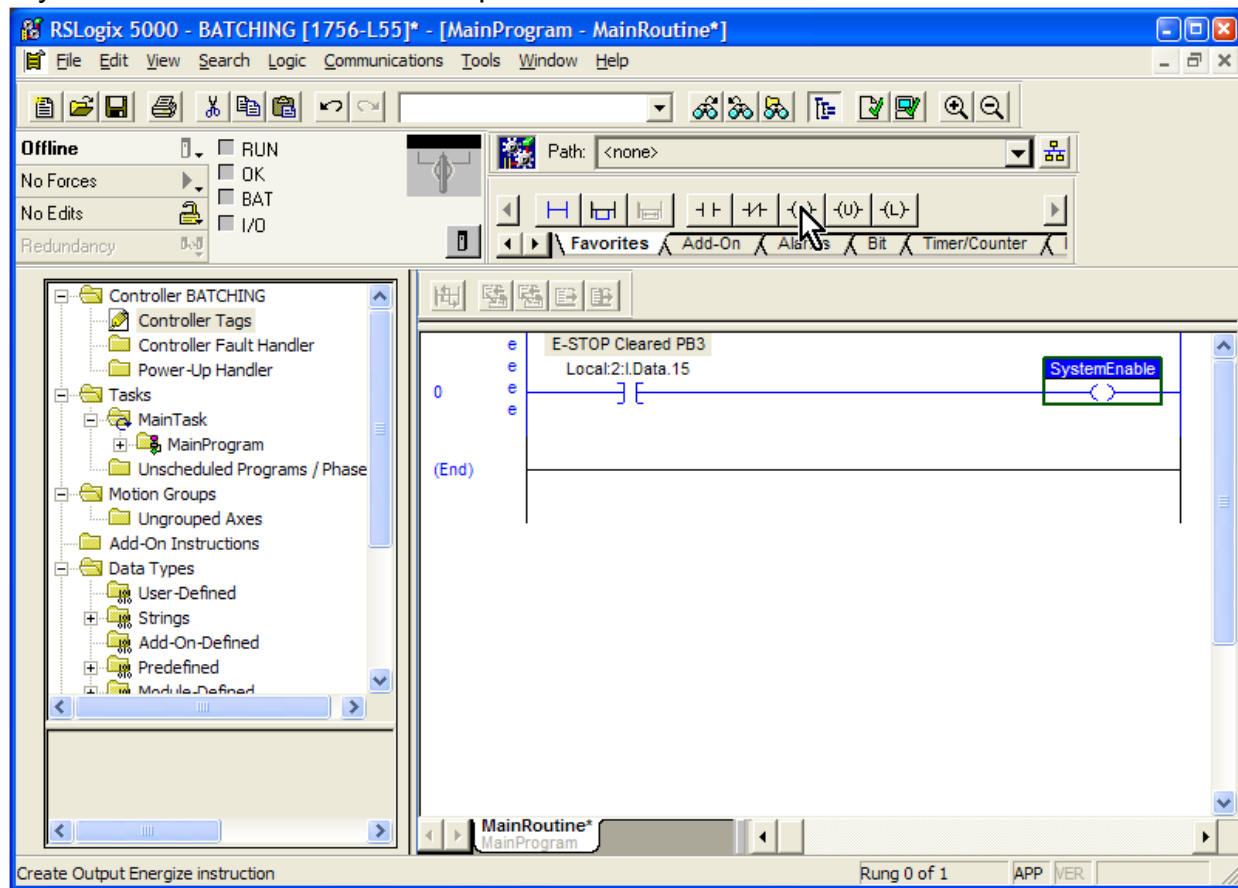
Click and drag the OTE (output energize) tool button  from the User menu down to the new rung. Place it on the marker at the right.



The screen looks like this.



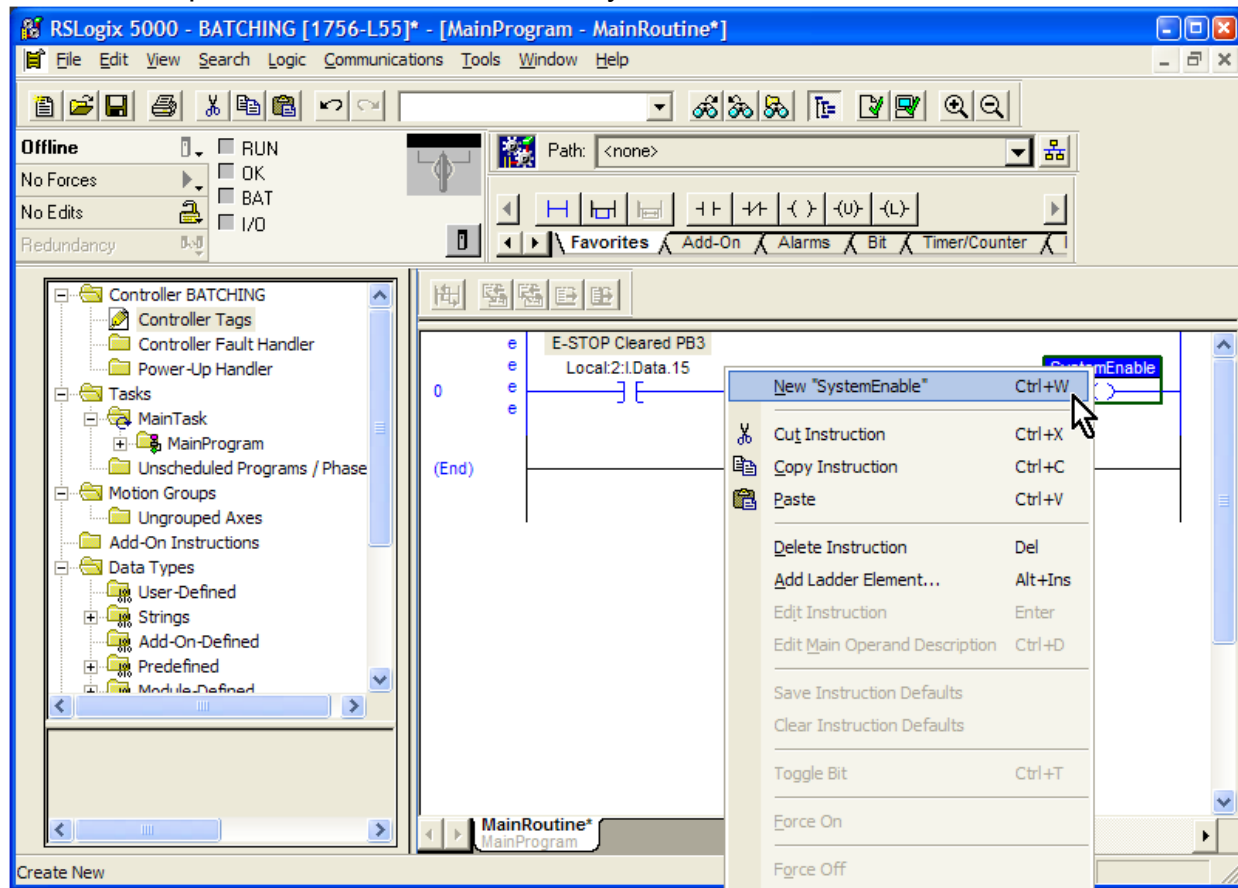
Click on the blue box (the tag name field) above the instruction. Type the phrase “SystemEnable” into the box and press enter.



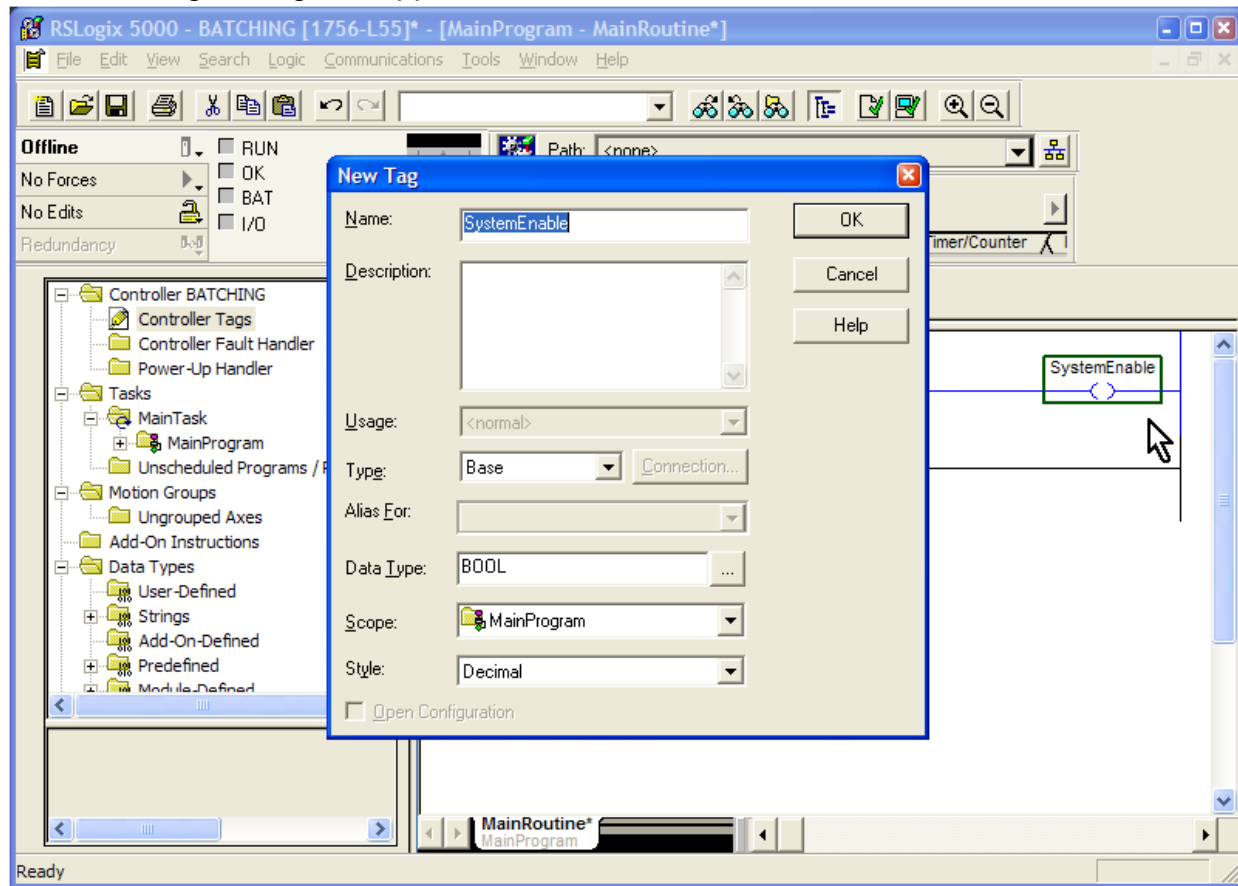
Since this is a tag name, there can be no spaces or special characters in the name.

This is a new tag for this program, so we have to define it. Right-click on the blue tag name field.

From the dropdown menu, choose “New System Enable”.




The “New Tag” dialog box appears.



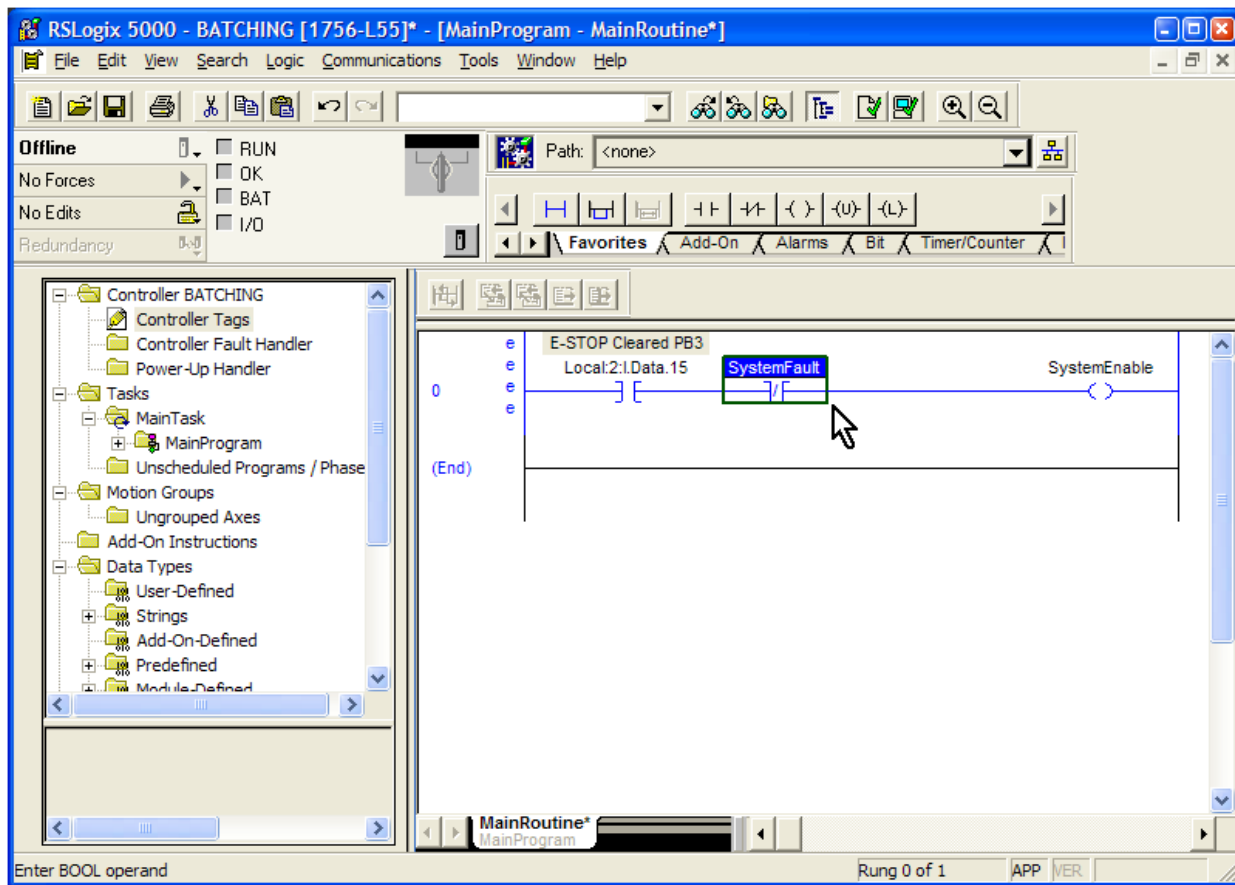
The tag we are defining here is simply a bit, so we can accept the default values in the dialog box. In fact, we really don't need to add a description, as the tag name itself is pretty self-explanatory.

Click “OK”.

We know from the Project Scope that the system must stop if there is a fault. We are not sure of the details of all those faults yet, but we do know that we will summarize those faults somewhere in the program. It will result in a bit. We will use the address tag “SystemFault” for that bit. We also know that we want the “SystemEnable” to be on if we do *not* have a fault.

Bear with me here and it will make sense. Click and drag the XIO (examine if open) tool button  from the User menu down to the new rung. Place it just to the right of the E-stop input. Double-click on the tag filed and type “SystemFault” in the box.

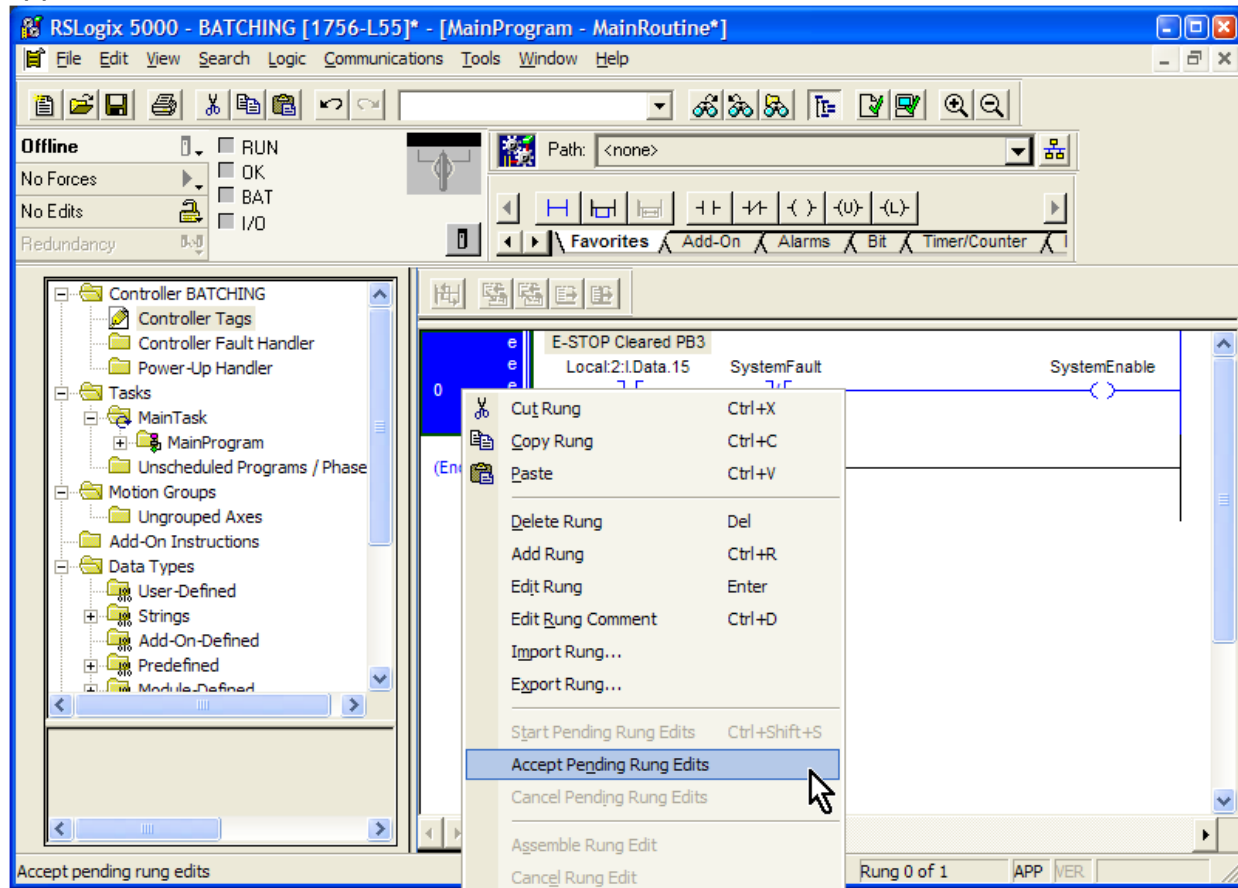
It should look like this:



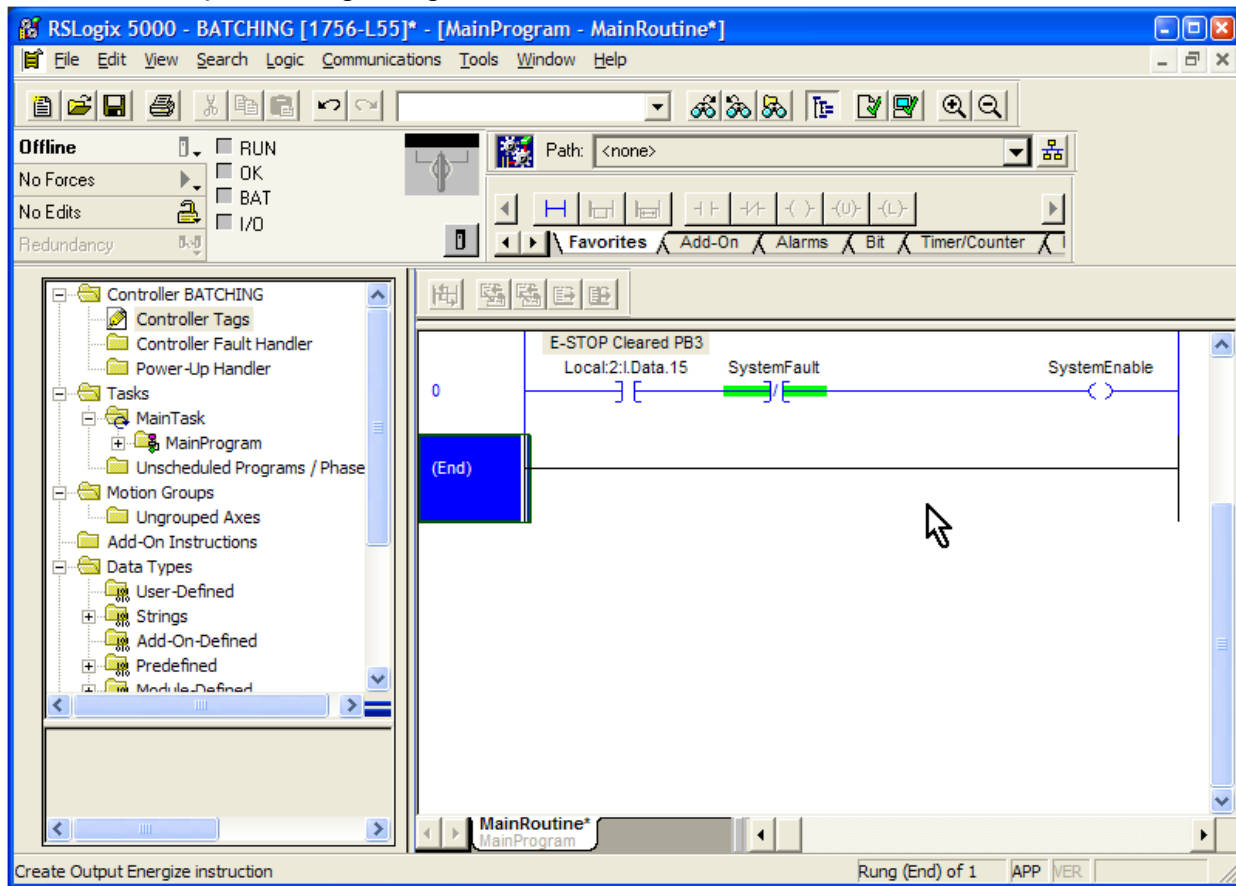
Right-click on the tag name and define the tag (you may also press CTRL-W to get to the “New Tag” dialog box). By the way, if you forget to define that tag, RSLogix 5000 will remind you when you accept the rung edits.

Let’s see what we have. The logic of the rung works just like an electrical circuit. If the E-Stop is cleared and there is not a System Fault, the System Enable bit will be on. That is exactly what we want. We will work out the fault logic later.

We need to accept the current rung. Right-click on the rung number (0) and this dropdown appears.



Choose “Accept Pending Rung Edits”.




We have completed the first rung in the program.

You may wonder why the SystemFault instruction is highlighted green. This is because the value of the SystemFault tag is 0. Since the instruction is an XIC, or Examine If Closed, the instruction is true. Therefore, the instruction is highlighted.


You should note that these colors are highly configurable. In fact, I have seen many different color schemes. Just keep in mind that you may look at someone else's laptop and find blue, for example, has been configured to highlight a bit that is true.


Starting a Batch Cycle


The Project Scope said that the operator may start a batch by pressing the “Start Batch” pushbutton on his console. Let's start with that input.

Right-click on the last rung and choose “Add Rung”. Click and drag the XIC (examine if closed) tool button  from the User menu to the left side of the new rung.

Click on the tag name field above the instruction and navigate through the tag groups until you find the input Local:2:I.Data.0. This is the “Start Batch PB1” pushbutton.

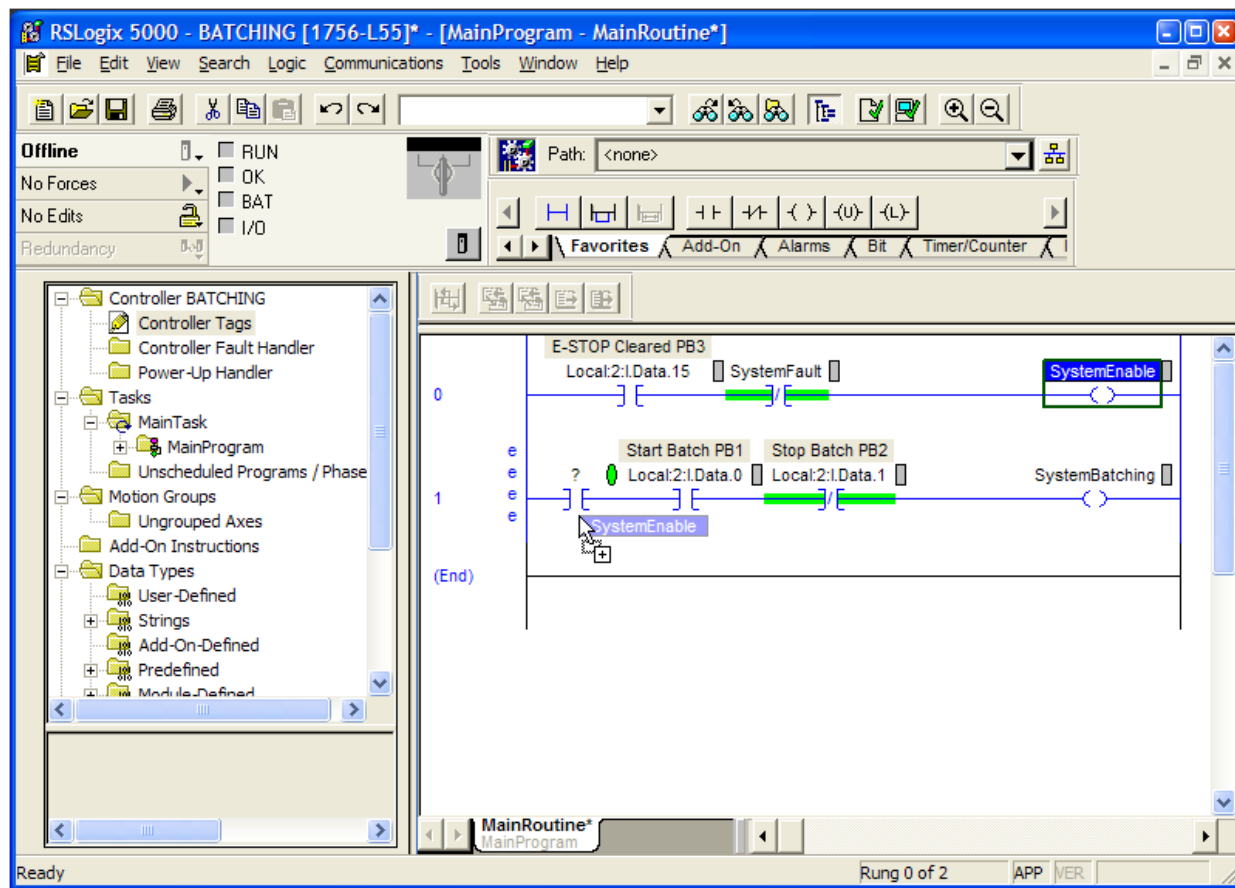
Click and drag the OTE (output energize) tool button  from the User menu down to the new rung. Place it on the marker at the right. We are creating a new tag that indicates the system is currently batching. Label this tag “SystemBatching”. Right-click on the tag, select “New SystemBatching” and accept the defaults.

If the operator chooses, he may stop the batch. We will make use of the “Stop Batch” pushbutton. Click and drag the XIO (examine if open) tool button  from the User menu down to the new rung. Navigate through the tag groups until you find the input Local:2:I.Data.1. This is the “Stop Batch PB2” pushbutton.

We don’t want the operator to be able to start a batch if the System Enable bit is not on. We will add that by dragging the XIC (examine if closed) tool button  to the left side of the new rung.

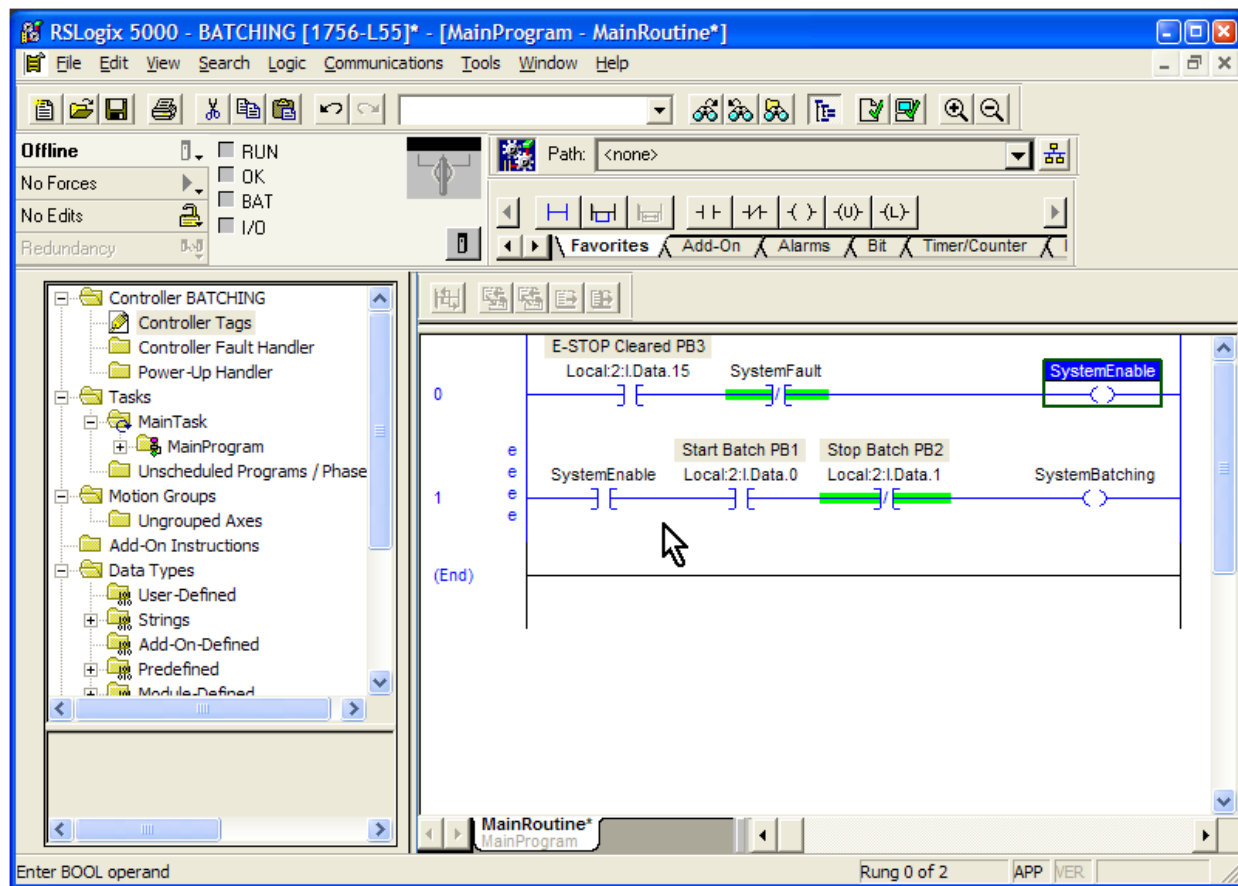
RSLogix 5000 lets us quickly assign a tag to this bit by dragging the bit name from another rung.

Click, hold and drag the tag name “SystemEnable” from the OTE in Rung 0. As you drag the tag name box, you will see gray rectangles appear, indicating that these are potential places to assign the tag. As the cursor gets nearer to a target, that target icon changes to a green oval.




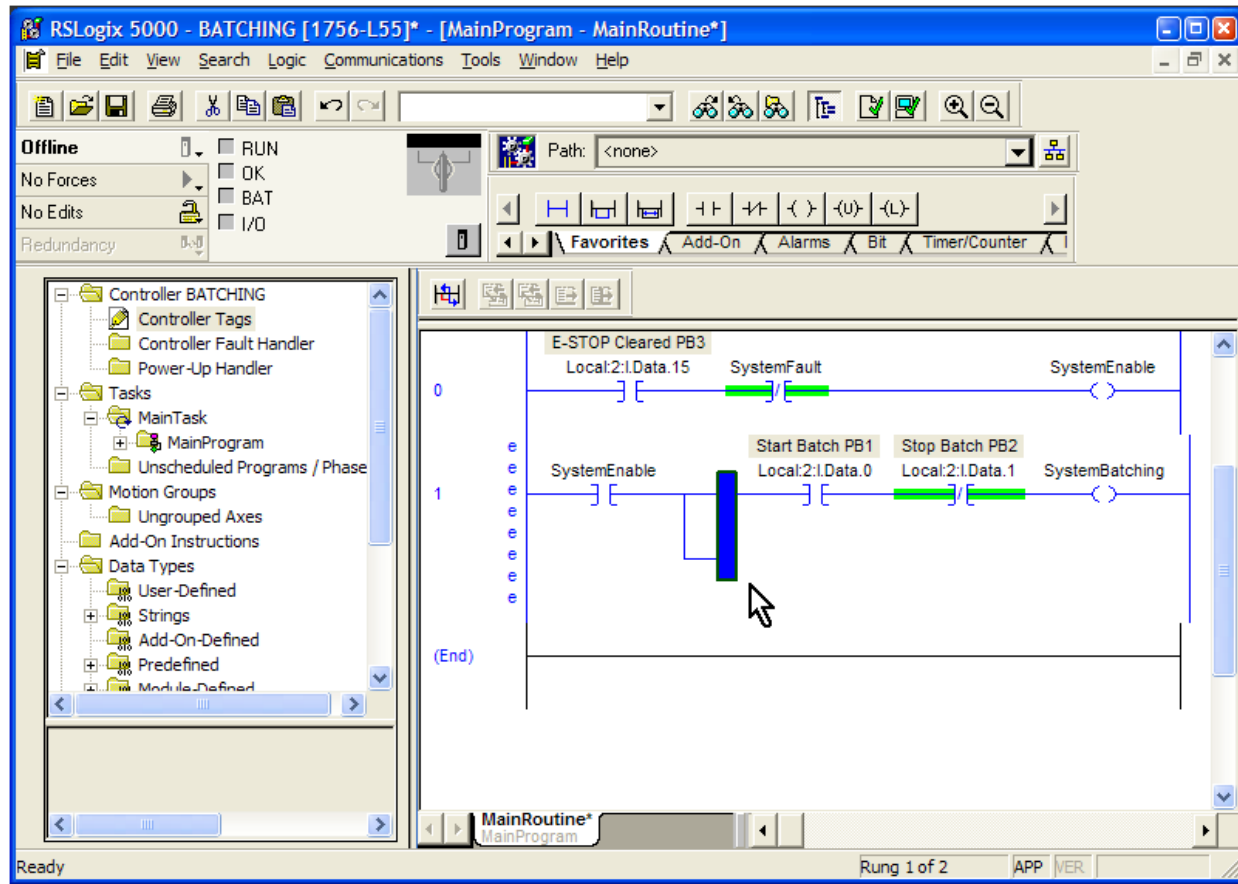
Release the mouse button when the cursor is near the first instruction in the rung.

Here is what we have so far:



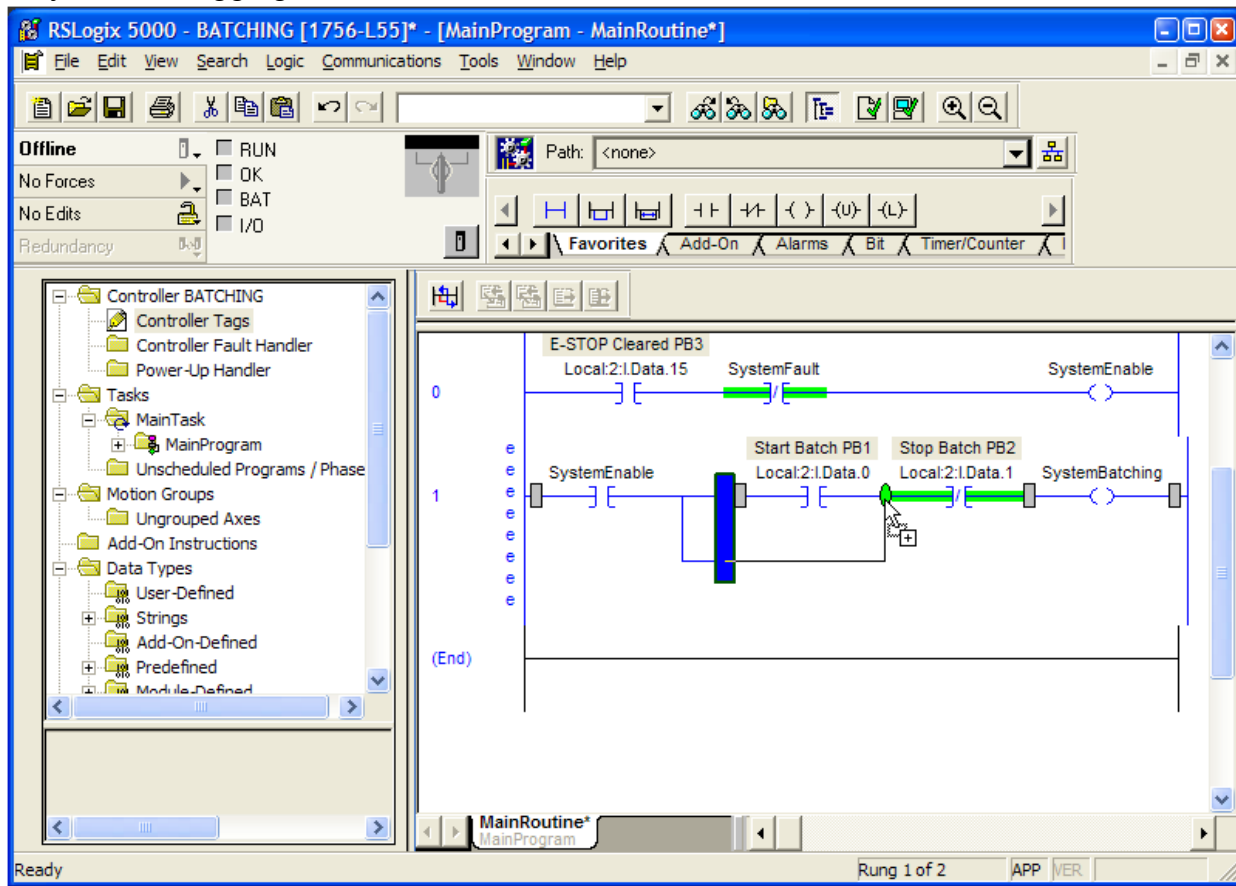
This rung we are creating will work much like a traditional motor starter circuit that uses a contact from the motor starter wired in parallel with the start button to hold in the coil. In the PLC, the “contact” is an XIO with the same tag as the “coil”, which is “SystemEnable”.

We need to “wire the contact” in parallel with the start button. We do this with a branch instruction. Drag the Branch tool button  and place it on the marker between the System Enable bit and the Start Batch bit.

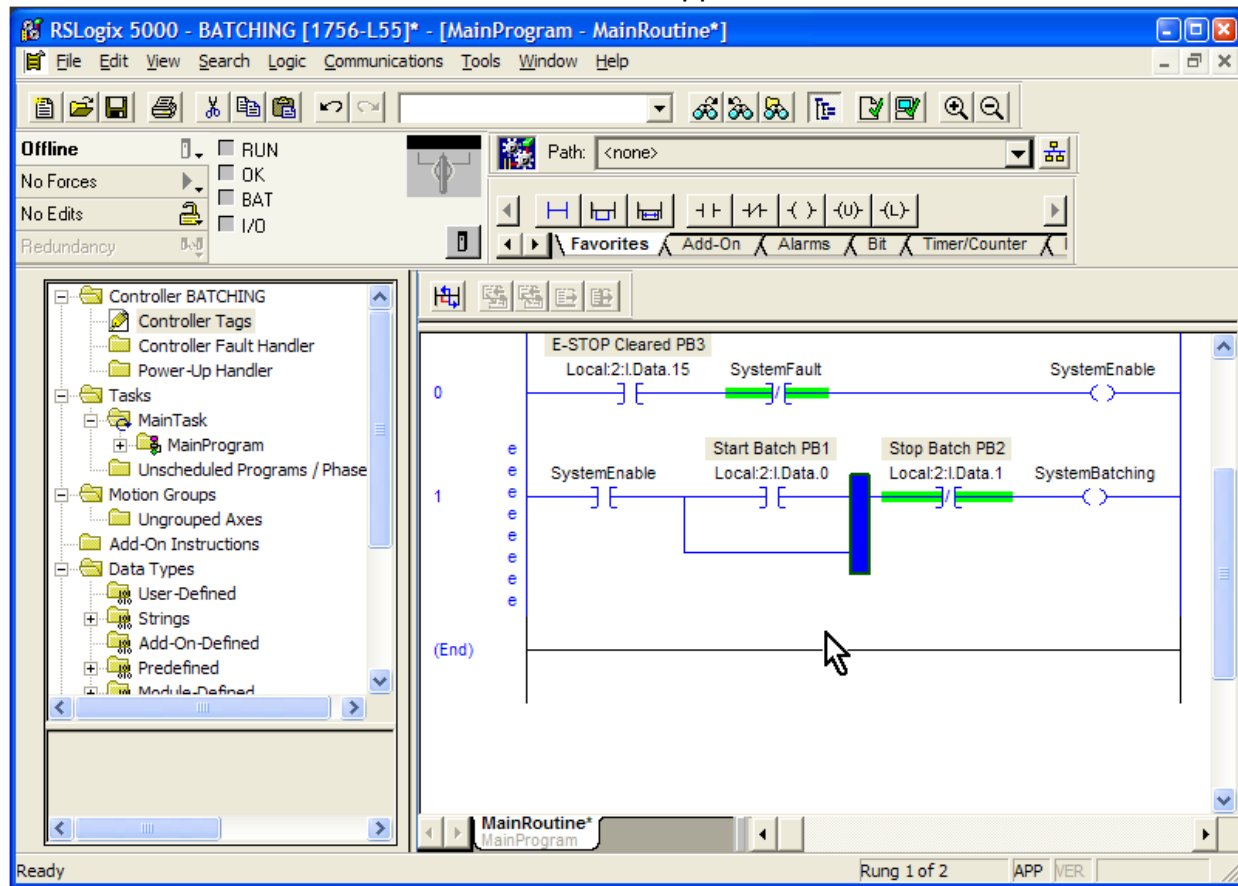



Click on the blue section of the branch and drag it to the target to the right of the Start Batch PB1 instruction.

As you are dragging, it looks like this.



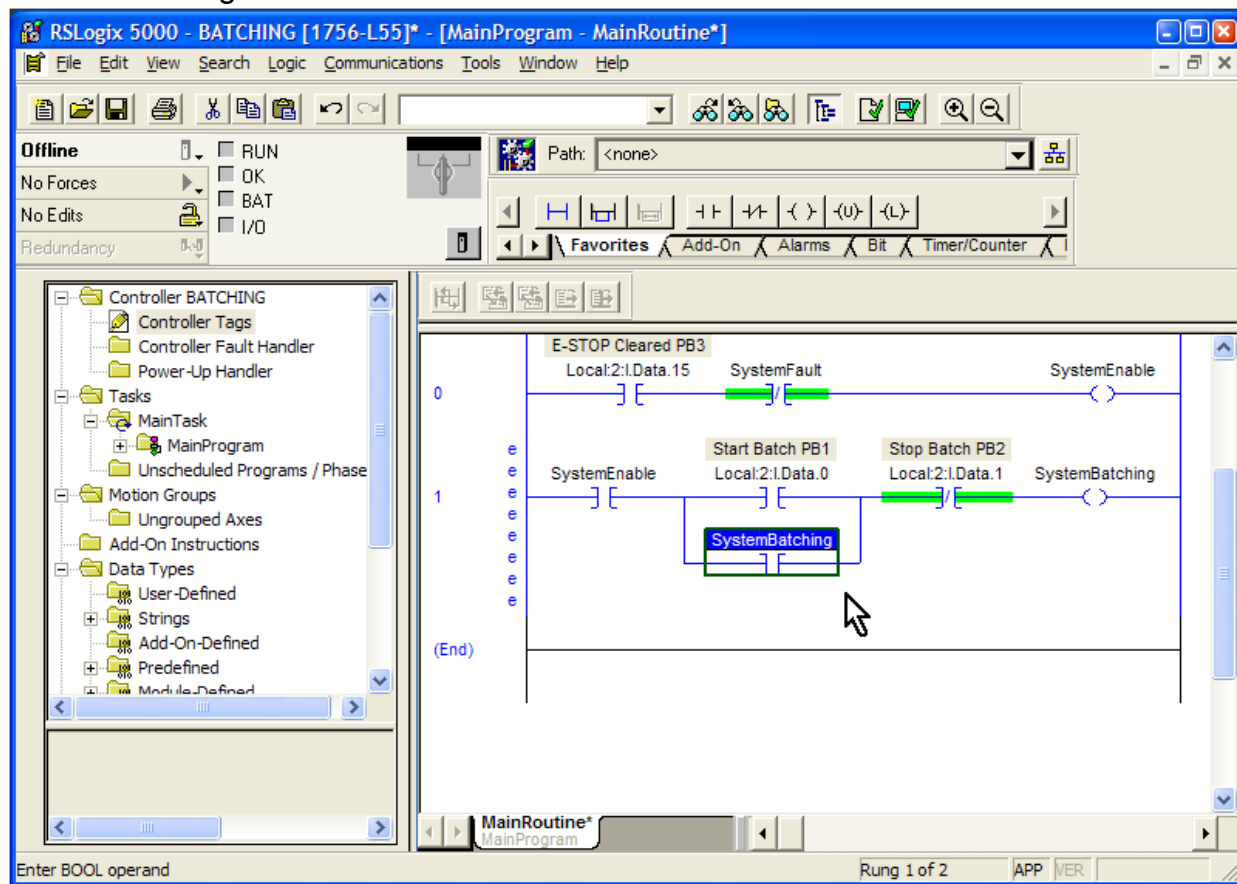
Release the mouse button and the branch will appear around the Start Batch bit.



Click and drag the XIC (examine if closed) tool button  from the User menu to the left side of the new branch.


We want this instruction to be tagged “SystemBatching”, just like the OTE in this rung.

There is a quick way to do that. Just click and drag the “SystemBatching” tag name from the OTE in this rung to the new instruction.

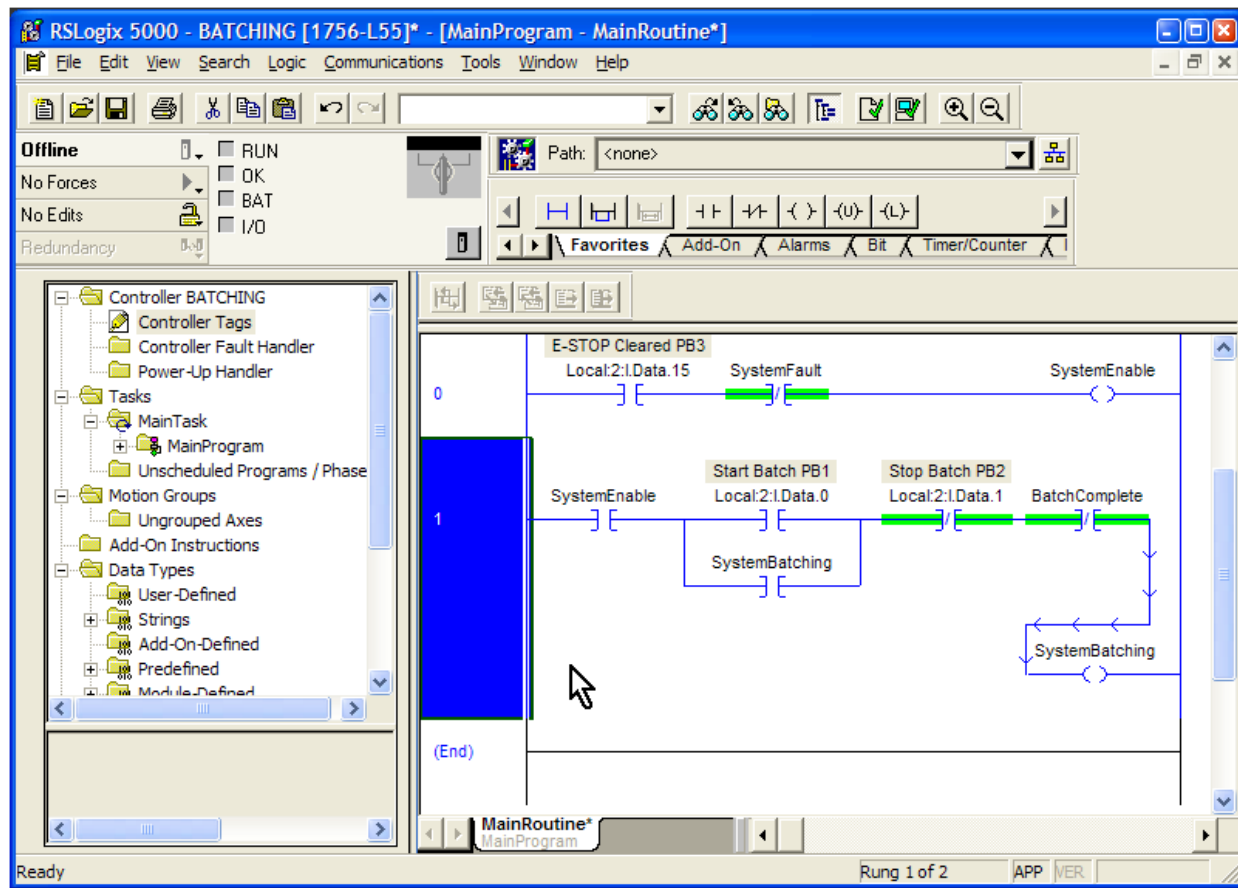


This is called a latching rung. If the SystemEnable bit is on, the SystemBatching bit can be latched by momentarily pressing the Start Batch pushbutton. The SystemBatching bit will stay on and the rung will remain latched until the Stop Batch pushbutton is pressed or the SystemEnable bit goes off.

Thinking ahead, though, we know that the system will stop the batch automatically after it has pumped all the finished product to the filling lines. We are not sure how we will know that yet, but we know we need a bit to unlatch the rung.

Click and drag the XIO (examine if open) tool button  from the User menu down to the marker just in front of the SystemBatching OTE instruction. Type in the tag “BatchComplete” and define the tag (CTRL-W).

Right-click on the rung number and verify the rung. It should appear like this.



Notice how the rung has become too long to be contained on one line, so RSLogix is putting the OTE instruction below and re-routing the connecting line. It is accurate, but a little confusing.

You can get around this by a couple of ways. You can set the “Zoom” factor under the “View” menu to get the rung to appear on one line.

You can also hide the Controller Organizer by pressing “ALT-0”. That is what we will do here. We don’t have a real need to see the Controller Organizer right now, anyway.

