

Cost Comparison

To “approximate” the content that we provide in our ControlLogix training program it would take 4 Rockwell classes and 10-days away from the work. And...

- There are no handbooks to take home—view only PDF
- There is no “rewind” button
- Is the instructor only reading the PowerPoints?
- Is the instructor a good “instructor”?
- How do you know that your technicians learned?...is there proof? ...because there are no test or quizzes
- The cost is over \$9,500—for one person and 10-days out
- How can your technicians review/refresh the materials? ...Are they going to forget the class in two weeks?

STUDIO 5000™ LOGIX DESIGNER LEVEL 1: CONTROLLOGIX SYSTEM FUNDAMENTALS

Programmable Controllers | In Person | Multiple Sessions Available

The course will assist you in developing and building a solid foundation with a fundamental knowledge of ControlLogix and other Logix5000™ systems. You will be introduced to basic Logix5000 concepts and terminology, and you will be exposed to Logix5000 system...

COURSE #:
CCP146

\$2,200.00

2-day

STUDIO 5000™ LOGIX DESIGNER LEVEL 2: BASIC LADDER LOGIC PROGRAMMING

Programmable Controllers | In Person | Multiple Sessions Available

After completing this course, you should be able to program basic ladder logic instructions for Logix5000™ controllers. This is a skill-building course that provides you with the resources required to complete this objective.

COURSE #:
CCP151

\$2,200.00

2-day

STUDIO 5000™ LOGIX DESIGNER LEVEL 3: BASIC LADDER LOGIC INTERPRETATION

Programmable Controllers | In Person | Multiple Sessions Available

This course is a skill-building course that provides you with a more detailed understanding of Studio 5000 Logix Designer® ladder logic instructions and terminology. This course also provides you with the resources and hands-on practice required to interpret ladder logic...

COURSE #:
CCL21

\$2,200.00

2-day

STUDIO 5000™ LOGIX DESIGNER LEVEL 2: CONTROLLOGIX MAINTENANCE AND TROUBLESHOOTING

Programmable Controllers | In Person | Multiple Sessions Available

Upon completion of this course, you will be able to troubleshoot a previously operational ControlLogix® system and restore normal operation. This course adds to your skill set by introducing new tasks such as connecting to a network, interpreting project execution,...

COURSE #:
CCP153

\$2,990.00

4-day

[GO To Product Site](#)

Understanding, Maintaining and Troubleshooting ControlLogix Systems

—with The Studio 5000 Designer Software—

Training Program

Module 1 Introduction to the ControlLogix—General Structure, Number Systems, and Basics of Boolean Logic

Module 2 ControlLogix Hardware Composition, I/O Structure and Architecture—Introduction to Tags

Module 3 Navigating the Studio 5000 Software and Creating, Opening and Understanding Projects

Module 4 Connecting to the Controller—Establishing RSLinx Connection to the Network

Module 5 ControlLogix Project Organization and Frequently Used Tag Structures

Module 6 Troubleshooting Ladder Diagram Logic in the ControlLogix System

Module 7 Creating and Editing Tags and Code—Documenting Troubleshooting Changes

Module 8 Troubleshooting Using the Studio 5000 Software—Using I/O Forcing and Toggling Functions

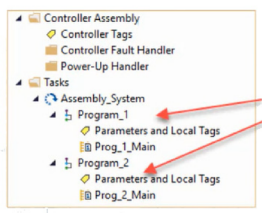
Module 9 Troubleshooting ControlLogix Hardware—Discrete and Analog I/O

Module 10 Troubleshooting Remote I/O, Controller and Power Supply—Using the Trend and Compare Tools to Troubleshoot

[GO To Product Site](#)

Understanding, Maintaining and Troubleshooting ControlLogix Systems—With Studio 5000 Designer Software

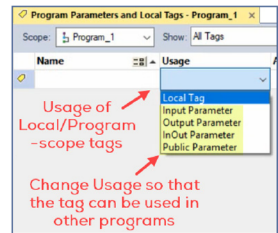
Local tags use what is referred to as Program “Parameters” to share data between Programs (Figure 10). In other words, if it is required to use a Local tag from one Program to another Program, then the programmer would change the Parameter in what is called the “Usage” of the tag from being a simple Local tag. When the Parameter is changed in the Usage, then the tag can be used in other Programs.



Parameters are used to share data between programs

Figure 10. Local tag from one program can be used in another if the “Parameters” are changed.

There are four types (Figure 11) of Program Parameters that can be used to share tag information between Programs, and these are Input, Output, InOut, and Public. These Program Parameters can share the data through what is called “Connections” (Figure 12), for instance to connect or relate to a base tag which has I/O devices wired to the module. We’ll see this connection feature shortly in an example.

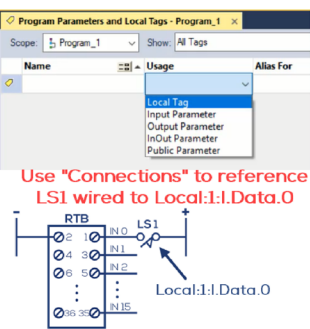


Usage of Local/Program -scope tags

Change Usage so that the tag can be used in other programs

Figure 11. Types or Usage for Local tags.

NOTES

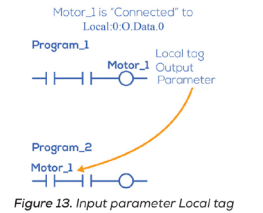


Use “Connections” to reference LS1 wired to Local:1:1.Data.0

Figure 12. Use of Connections to pass parameter data.

Module 7—Creating and Editing Tags and Code—Documenting Troubleshooting Changes

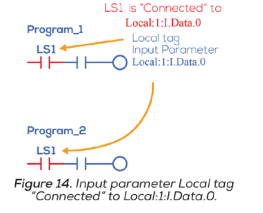
The Input Parameter usage passes data, for example tag status ON or OFF, by value from one Program to another (Figure 13). The value passed does not change during the execution of that Program scan. When the tag is “Connected”, or like-aliased, to an input from a module base tag the value of the tag will not change during execution of the Program due to the asynchronous I/O operation. This avoids for the programmer having to “freeze” the inputs, or buffer inputs, before the Program scan.



Motor_1 is “Connected” to Local:0:0.Data.0

Figure 13. Input parameter Local tag

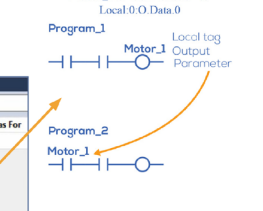
For instance, if LS1 is Connected, which is like a pseudo-alias, to Local:1:1.Data.0 (Figure 14), and the value of LS1 at the beginning of the program scan is 1, the tag value will not change during the program scan. The value of LS1 is sort of frozen from the beginning to the end of the program scan—any asynchronous change, for instance from logic 1 to 0, will not affect the LS1 status in the code in that scan.



LS1 is “Connected” to Local:1:1.Data.0

Figure 14. Input parameter Local tag “Connected” to Local:1:1.Data.0.

The Output Parameter defines tag data that is a result of the Program execution, basically producing an output (Figure 15). When a tag Usage is changed to Output Parameter the value is passed and does not change until the end of the Program scan. Again, like in the Input Parameter, when a Local output tag Usage is changed to Output Parameter and the tag is “Connected” to an output from a base tag, it allows the programmer to avoid implementing output buffering to freeze the outputs until the end of the Program scan.



Motor_1 is “Connected” to Local:0:0.Data.0

Figure 15. Output parameter Local tag “Connected” to Local:0:0.Data.0.

10
Copyrighted Material. Duplication is prohibited under the law.

11
Copyrighted Material. Duplication is prohibited under the law.